© Copyright 2026

Sicen Liu

LLM-Driven NPCs for Realistic Behavior and Collaboration in Games

Sicen Liu

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2026

Committee:

Kelvin Sung

Arkady Retik

Wooyoung Kim

Program Authorized to Offer Degree:

Computing & Software Systems

University of Washington

Abstract

LLM-Driven NPCs for Realistic Behavior and Collaboration in Games

Sicen Liu

Chair of the Supervisory Committee:
Kelvin Sung
Computing & Software Systems

Realistic behaviors of non-player characters (NPCs) are important in interactive games and simulations because NPCs influence how players understand and interact with virtual worlds. Large language models (LLMs) provide new opportunities for developing believable and socially responsive NPCs. A review of recent work shows that two areas have received limited attention in existing LLM-based approaches: NPC decision-making grounded in a continuously updated environment and inter-NPC interactions based on observable agent states.

This thesis proposes an LLM-driven NPC framework to address these issues. The framework maintains a centralized Global World State that records environment states, player states, and observable states of the NPCs, providing NPC reasoning with explicit information about available entities, their locations, and current conditions. An LLM-based decision-making pipeline uses these structured representations to generate actions, update world states, and maintain behavioral continuity.

The framework is implemented as a turn-based simulation in which time advances through discrete ticks. Each tick processes the Environment, Player, and NPC Turns in a fixed order, allowing all three to update the shared world state. Simulation results show that NPC behavior remained consistent with current object availability, object states, locations, and environmental changes while adapting to other agents' requests and ongoing actions. These capabilities enabled coherent multi-step collaboration among multiple NPCs in a dynamic environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Kelvin Sung, for his guidance and support throughout my graduate study and the development of this thesis. His careful attention to detail, rigorous approach to research, and high standards for both technical work and academic writing have shaped not only this thesis but also the way I approach problems and evaluate my own work. I am especially grateful for the time and patience he devoted to repeatedly reviewing my work, discussing ideas with me, and helping me improve them. The lessons I learned from this process will continue to influence me beyond this thesis and throughout my future work.

I would also like to thank my committee members, Professor Arkady Retik and Professor Wooyoung Kim, for their time, valuable feedback, and suggestions during the development and examination of this thesis.

This work was supported in part by graduate student funding from the operating budgets of the CSSE/CSE graduate programs at the University of Washington for thesis and project work. I gratefully acknowledge this support, which helped cover the computational costs of the experiments conducted in this research.

Finally, I would like to express my heartfelt gratitude to my family for their unwavering support, which gave me the confidence and stability to pursue my graduate education abroad. I am also deeply grateful to my partner, Dingyuan, with whom I have shared this journey from the beginning, studied together, and supported each other through both difficult and rewarding moments. I am also thankful to his family for their generous support and encouragement throughout our studies.

Chapter 1. INTRODUCTION

Realistic behaviors of Non-Player Characters (NPCs) are an important component of interactive games [1] and simulations [2]. In digital games, NPCs are not only opponents or background entities but also social actors, collaborators, guides, and sources of narrative interaction [3]. The believability of NPC behavior can affect how players understand and interact with the game world, directly affecting the overall immersive experience [4], [5]. Beyond entertainment games, realistic NPCs are also relevant to simulation systems, where artificial agents may be used to reproduce human-like activity patterns or support training and scenario analysis [2]. For these reasons, both academic [3], [6] and industrial research [7], [8] have continued to explore the development of believable NPC behaviors that are adaptive and responsive to player actions and changing environments.

Large language models (LLMs) offer new opportunities to develop more flexible, context-sensitive NPCs [9]. Unlike traditional rule-based dialogue or behavior systems [10], LLMs can generate open-ended natural language responses conditioned on a given interaction context, making them suitable for NPCs that need to respond dynamically to player input [11]. Recent work has shown that LLMs can be extended beyond dialogue generation to support agent-like functions such as recalling from memory, planning, reflection, and action selection [12]. These capabilities are especially relevant to the design of NPCs because they often are required to maintain a role, interpret the current situation, choose an appropriate response, and interact with other agents or players over time [12]. In this way, LLMs offer a promising technical foundation for NPC systems, enabling a shift beyond fixed scripts toward more adaptive, socially responsive behaviors.

Several recent works have explored LLM-based architectures for believable agents and NPC-like systems. For example, generative-agent-style systems [12], [13] demonstrate that LLMs can be structured with memory, reflection, planning, and perception to support daily plans and emergent social behaviors in a multi-agent sandbox. In another example, Wang et al. place LLM-driven agents in interactive games such as Minecraft [14], where agents support long-horizon behavior based on environmental feedback, predefined executable skills, and iterative prompting with an LLM. Together, these studies have demonstrated that LLMs can be integrated into agent architectures that perceive, remember, reason, plan, and act [9]. In this context, there are two main

areas that warrant further investigation. First, NPC decisions grounded in an explicit environment that is continuously and dynamically updated. Second, multi-NPC interactions that are based on observable states of other agents.

These two are important areas because believable NPC behavior depends on the coherence between an agent's internal reasoning and the external simulation context. When NPC decisions are grounded in explicit environment states, their actions can be derived based on concrete objects, resources, and ongoing processes that already exist in the scene. For example, an NPC can respond to a functioning toaster, available bread, or an occupied table as part of the current situation. Multi-NPC interaction is essential because, in a shared environment, an NPC should respond based on the visible actions of other characters. With agent states represented explicitly, the system can allow an NPC to observe other characters' current actions, locations, and objects that they are handling, and then interpret this information as part of the interaction context. For example, when Alice observed that Bob was handling a pot, she could respond by waiting for the pot or offering assistance as her next step.

To support environment-grounded decision-making and multi-NPC responsiveness, this thesis proposes an LLM-driven NPC framework that maintains consistency, explicitness, and inferable observability through three main components: a centralized Global World State, explicitly maintained NPC observable states, and an LLM-based decision-making pipeline. The Global World State maintains consistency by providing a shared and continuously updated representation of the environment, player, and object states, and the locations of NPCs. This structured representation also supports explicitness by directly providing object availability, states, and location information during NPC reasoning. The observable-state mechanism supports inferable observability by representing externally visible NPC conditions, such as current actions, facial expressions, and objects that they are handling, while excluding private intentions and hidden reasoning. The LLM-based decision-making pipeline refers to this information during NPC's planning, action generation, interaction handling, and world-state updating to control the behaviors of NPCs. Together, these components allow NPC behaviors to remain grounded in the current environment while also responding to other agents in a turn-based multi-NPC simulation.

To evaluate the proposed framework, a controlled research prototype is implemented to simulate a world with discrete ticks of constant duration. During each tick, the system processes the Environment Turn, Player Turn, and NPC Turn in a fixed order, allowing environmental

changes, player actions, and NPC reasoning to update the shared world state. The details of the system state and NPC behaviors are tracked and recorded. Within the NPC Turn, each NPC executes a reasoning pipeline that includes memory retrieval, observable-state generation, action planning and generation, interaction handling, and world-state updating. Additional tools are implemented to support interactive analysis of simulation traces, behavioral statistics, and execution flows. The implemented system is not intended to be a complete game or game-engine integration. The evaluation is also not intended to establish general applicability or robustness across arbitrary game genres or environments; instead, it examines whether the targeted behaviors enabled by the proposed mechanisms can be observed in a controlled multi-NPC simulation.

The simulation results demonstrated that the proposed framework can support environment-grounded decision-making and inter-NPC interactions. For example, during lunch preparation, NPCs acted around actual objects in the kitchen, such as eggs, bread, tools, and shared furniture. They refined their behavior in response to one another's requests and ongoing object use, such as waiting for or coordinating in handling shared items. These observations demonstrated that explicit environment states and observable agent states are taken into consideration and can result in improved multi-NPC coordination in a simulation.

The contributions of this thesis are twofold. First, the proposed framework supports environment-grounded NPC decision-making by incorporating explicit and continuously updated world-state information into NPC planning and action generation. Second, the framework supports multi-NPC responsiveness by enabling NPCs to respond to externally visible conditions of other agents during decision-making and interaction. These capabilities are implemented through continuously updated structured world states and explicit NPC observable-states representations, and are evaluated in a controlled multi-NPC simulation.

The remainder of this thesis is organized as follows. Chapter 2 reviews related work on NPC believability and LLM-based game agents. Chapter 3 presents the design of the proposed framework. Chapter 4 describes the implementation of the prototype system and the supporting tools. Chapter 5 presents the simulation results and discusses how the system supports environment-grounded decision-making, multi-NPC coordination, and player interaction. Chapter 6 concludes the thesis and discusses limitations and future work.

Chapter 2. RELATED WORK

This chapter reviews prior work on simulating believable NPC behavior and LLM-based agents in games. The chapter first discusses NPC believability and the limitations of traditional approaches and then reviews how improved LLM-based agents have been applied in game environments. The chapter then focuses on generative-agent-style systems, with particular attention to how existing work supports character behavior and multi-agent interaction. The chapter concludes by identifying opportunities to complement existing work with explicit, continuously updated world states, observable NPC states for multi-agent interaction, and player-aware state and action updates.

2.1 NPC Believability and LLM-Based Agents

Non-Player Characters (NPCs) are important in many digital games because they mediate player interactions with game worlds and narrative situations [15]. Because NPCs are presented to players through roles, behavior, dialogue, and social interaction, their believability can shape how players interpret the coherency and responsiveness of the game world [4], [5], [15], [16]. Traditional NPC behavior has been implemented through manually authored control methods, such as scripted dialogue systems and rule-based behavior control [17]. These methods are useful for producing stable and predictable behavior. However, they also depend on anticipated contexts and can be challenging to scale when NPCs must respond to variable player input, changing world states, unforeseen actions, or social interaction with other characters [16], [18], [19].

Outside game applications, recent advances in large language models (LLMs) have enabled flexible, context-sensitive behaviors across a range of tasks. Early models such as GPT-3 [20] demonstrated that LLMs can perform many tasks based on text instructions and in-context examples without task-specific parameters. Later, instruction-tuned models [21] further improved the ability of LLMs to follow user intents across a range of tasks. These capabilities have supported the development of LLM-based autonomous agents to interpret context, generate plans or responses, and produce outputs that can affect an external system [22]–[24].

Recent work, often presented as LLM-based game agents [9], has begun to apply LLM-based autonomous-agent results in interactive gaming environments. As Hu et al. proposed [9], LLM-based game agents can be classified according to a variety of types, such as task-execution agents, exploration agents, strategy agents, interactive characters, and social simulation agents. Among

these, interactive characters and social simulation agents are often implemented as NPCs in games [9], [12], [14], [25]–[28]. These systems have demonstrated that LLMs can support context-sensitive decision-making, action generation, character interaction, and multi-agent behavior in gaming environments.

2.2 LLM-Based Game Agents

Hu et al. [9] describe LLM-based game agents as systems that connect LLMs to gaming environments via memory, reasoning, and perception-action interfaces. These systems typically include additional communication and organizational mechanisms in multi-agent settings. As detailed in the following, this thesis identified two areas that are particularly relevant to NPC believability: goal-directed agents for task execution in game worlds, and LLM-based interactive characters that support NPC interaction and social behavior.

Goal-directed LLM-based game agents are agents that employ LLMs to plan, decide, or act toward explicit objectives in simulated worlds [14], [25], [26]. It is important to note that these agents may not be designed as NPC-like characters. For example, DEPS [25] supports LLM-based interactive planning, where plans for open-world tasks are revised based on execution descriptions and failure feedback. In gaming contexts, JARVIS-1 [26] emphasizes perception-action coupling by mapping multimodal observations and human instructions to planning, recalling from memory, and embodied control in Minecraft. Voyager [14] extends task-oriented game agents toward open-ended lifelong exploration by leveraging an automatic curriculum, an executable-code skill library, and environmental feedback to acquire reusable skills over time. These systems demonstrate how LLM-generated plans or decisions can be linked to executable actions and updated through feedback to support long-term progress in game worlds.

LLM-based interactive characters are agents presented as NPC-like entities with roles, dialogue, memory, and social behavior. Volum et al. [27] employ a code-oriented language model to generate interactive Minecraft characters that can produce both natural-language responses and executable actions. Rao et al. [28] study player-NPC collaboration in an in-game quest and identify the importance of proper game-state information and visual understanding. Park et al. [12] present a persistent multi-agent social-simulation approach in which LLM-based characters remember events, form plans, initiate conversations, and exhibit emergent social behavior in a shared

sandbox environment. These systems focus on character presentation, player or agent interaction, and persistent social context.

Together, these two categories of results connect LLM-based game agents to NPC believability in complementary ways. Goal-directed task-execution systems demonstrate how LLM-generated decisions can be translated into executable actions, evaluated against environmental feedback, and revised to support long-term task progress. Interactive character and social simulation systems demonstrate how LLMs can support character roles, dialogue, memory, collaboration, and persistent social behavior. Taken together, these examples provide two useful perspectives on how LLM-based agents can contribute to believable NPC behavior in interactive game worlds.

2.3 Generative Agents

Generative Agents [12] provides the central example of LLM-based persistent social character simulation. It supports simulating interactions among LLM-based characters in a shared social environment, where multiple characters repeatedly observe the environment, update their internal context, and affect one another over time.

The architecture of the Generative Agents system is illustrated in Figure 2.1. The Perceive component sends natural-language observations from the sandbox environment into the Memory Stream module, where the agent's experiences are stored. The Retrieve module then selects relevant entries from the Memory Stream based on recency, importance, and relevance, producing the Retrieved Memories shown on the right side of the generative-agent memory structure. These retrieved memories support the Reflect and Plan components: Reflect synthesizes higher-level inferences from past experiences, while Plan produces daily schedules and decomposes them into more specific actions. The Act component then generates actions and conversations, which create new observations that are again stored in the Memory Stream and influence later behavior. This system demonstrates how memory, retrieval, planning, and reflection can be organized to produce believable daily plans, conversations, and emergent social behaviors among multiple agents.

Later systems extend the direction of Generative Agents into broader simulation or virtual-environment settings. AgentSims [29] provides a sandbox for evaluating LLM agents with configurable agents, environments, memory, planning, and tool-use mechanisms. Humanoid Agents [13] extends the Generative Agents model by incorporating basic needs, emotions, and

relationship closeness to shape agents' daily activities and conversations. Lyfe Agents [30] further explores real-time social interaction among generative agents in a 3D virtual environment.

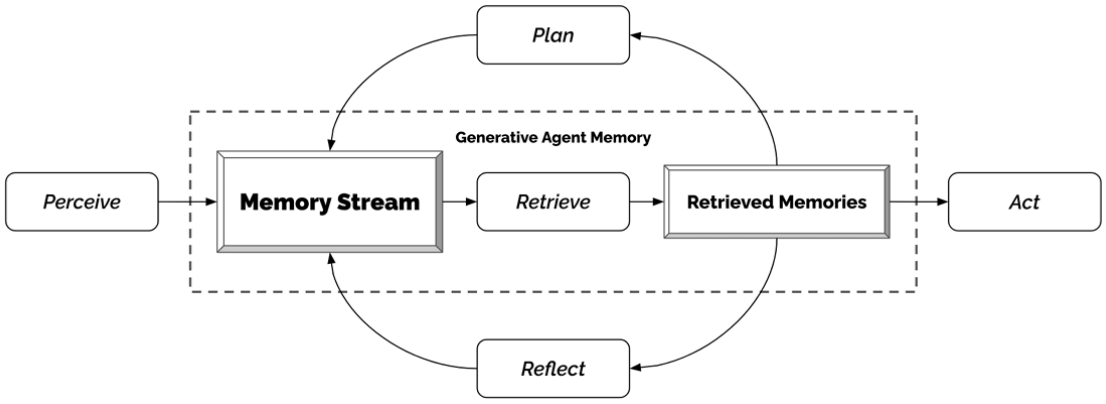

Figure 2.1. The Architecture of Generative Agents System [12].

2.4 SUMMARY

Based on the results discussed, there are opportunities for improvement in the context of believable NPC behavior in persistent gaming environments.

First, existing results are organized around memory-based experience recording rather than an explicit and continuously updated representation of the current world state. As a result, ordinary NPC decisions and actions are not consistently grounded in the current availability, condition, or recent change of objects in the environment.

Second, these systems do not explicitly represent the current observable states of other agents, such as what another NPC is doing, holding, or visibly expressing. As such, it can be challenging to support non-verbal interaction and immediate cooperation between NPCs.

Third, although existing work may include players or users in the interaction loop, they are often treated as external dialogue partners [27], task providers [28], or instruction sources [29] rather than as world-embedded entities whose actions become part of the shared simulation context. The existing systems do not support player-NPC encounters as intrinsic and regular interactions in a persistent environment.

The next chapter presents a framework designed to support the identified functionality through a Global World State, observable NPC states, NPC planning and action generation, interaction resolution, and turn-based world-state updates.

Chapter 3. DESIGN

Figure 3.1 presents the overall architecture of the proposed LLM-driven NPC framework summarizing how an NPC connects character information, memory, action generation, and structured world states during simulation.

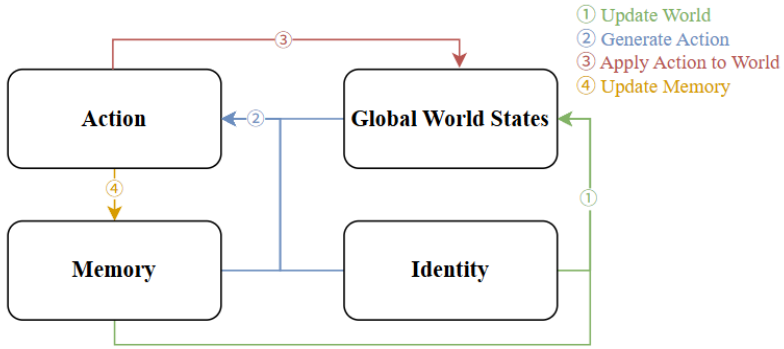

Figure 3.1. NPC Architecture.

Similar to the Generative Agents architecture [12], the architecture contains four main modules: Action, Global World States, Memory, and Identity. Among them, Global World States is a shared and detailed form of spatial information when compared to Spatial Memory.

The upper-left module of Figure 3.1 shows the Action module, which represents the process that generates an NPC's behavior. The upper-right shows the Global World States module, which stores the shared structured state of the simulation world. The lower-left shows the Memory module, which stores the NPC's accumulated experiences during the simulation. The lower-right shows the Identity module, which stores the stable character profile of the NPC.

The colored arrows in Figure 3.1 represent the information flows between these modules during each simulation cycle. In Step 1, the Update World green arrow shows that Identity and Memory provide information for updating the NPC-related state in Global World States. In Step 2, the Generate Action blue arrow shows that Identity, Memory, and Global World States are used as inputs to the Action module. The Apply Action to World red arrow in Step 3 shows that the result of an action is written back into the Global World States. The Update Memory orange arrow in Step 4 shows that relevant action information and resulting state changes are recorded into Memory.

Together, these modules describe the high-level structure of the framework. The following sections expand this overview by describing the details of each module and the simulation loop.

3.1 Action Module Components and Flow

Figure 3.2 expands the Action module introduced in Figure 3.1 and shows the action loop of an NPC. The step labels on the top-right of this figure are consistent with those of the overview in Figure 3.1: Steps 2a, 2b, and 2c are sub-steps of Step 2, Generate Action, while Steps 3 and 4 correspond to applying action effects and updating memory after action generation. In this action loop, the NPC first forms a short-term intention, then selects a concrete action, then produces world-state effects, and finally updates the shared simulation state and memory records.

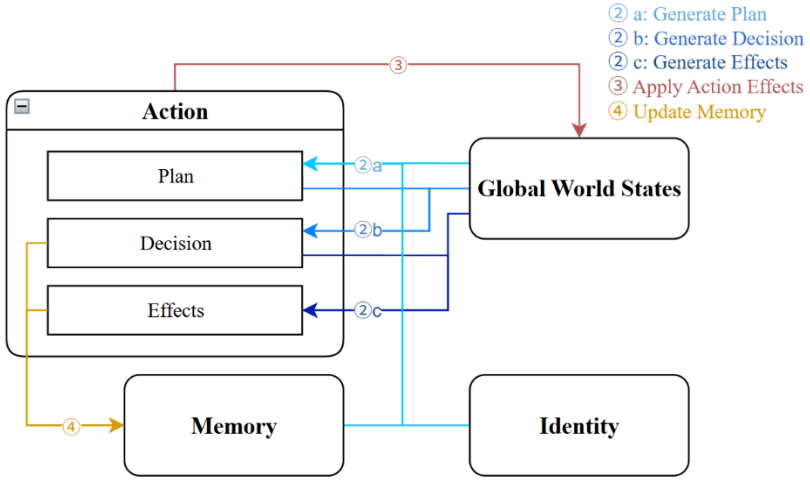

Figure 3.2. Internal Flow of the Action Module.

The upper-left part of Figure 3.2 shows that the Action module consists of three components: Plan, Decision, and Effects. At the very top of the Action Module, the Plan component represents the NPC's short-term intention. In Step 2a, an LLM generates the Plan based on the Identity of the NPC, Memory, and the current Global World States. The output describes what the NPC intends to do over the next short period and provides the basis for the following decision. For example, "Walk to the workshop and open the laptop for a short, quiet reset after confirming Bob has finished the kitchen cleanup".

In the center of the Action Module, the Decision component converts the Plan into a concrete action. In Step 2b, an LLM generates the Decision based on the current Plan and Global World States. The Decision is generated as an open-ended action description rather than a choice from a fixed set of candidate actions. This decision is constrained by the current simulation state: the actor, action, target, and reason must remain consistent with the NPC's context and with the objects,

agents, and conditions represented in the Global World States. For example, based on the above plan, the Decision may be “Alice switches on the laptop”.

At the bottom of the Action Module, the Effects component represents the direct consequences of the Decision. In Step 2c, an LLM generates the Effects by considering the Decision and the current Global World States. The output describes how the selected action changes the simulation state. For example, if the Decision is that Alice switches on the laptop, the Effects may update the laptop state to indicate that it is on, and update Alice’s observable action to show that she is using the laptop.

Step 3 updates the Global World States according to the generated Effects. This is a structured state-update operation rather than an LLM generation process. This step ensures that the action results are explicit and persistent parts of the simulation state. This allows later NPC decisions to be grounded in what has actually been changed in the world. Step 4 then summarizes the results from the Decision and Effects, recording the relevant action and consequences in Memory.

3.2 Global World States Components and Flow

Figure 3.3 examines the components of the Global World States and their interactions with the other modules.

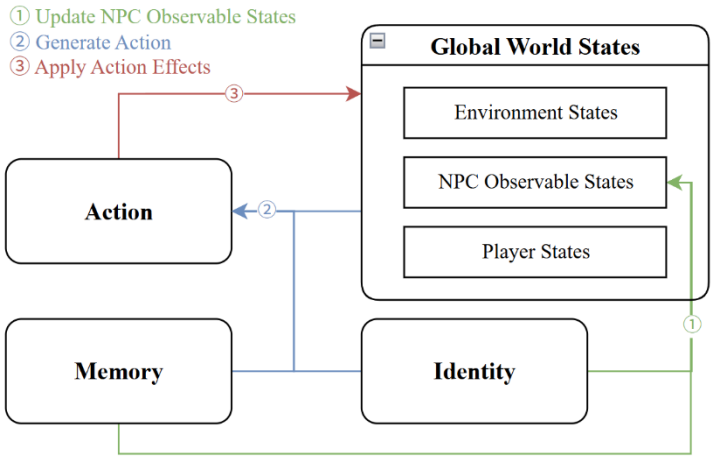

Figure 3.3. Internal Flow of the Global World States.

The Global World States module is the shared, structured representation of the simulation world. It records the external environment, NPC and player state information that can be referenced during action generation.

The Global World States module is organized as a hierarchical tree. As illustrated in the upper-right of Figure 3.3, the root node of the hierarchical tree contains three major components: Environment States, NPC Observable States, and Player States. Each component contains nested nodes that represent rooms, objects, object attributes, agents, agent attributes, and so on. This tree structure represents the existence of interactive objects, the spatial organization among objects, and the object states within the same framework. This hierarchical design also allows the Global World States to be extended by adding new branches or fields while keeping the same overall organization. The following subsections detail these three components.

3.2.1 *Environment States Components*

Figure 3.4 shows the structure of Environment States. Environment States represent the physical and interactive world of simulation. This state information is organized as a hierarchical structure of at least three levels, where the first level encodes logical locations, e.g., rooms. The objects that can be found at a location are always stored in the second level, e.g., a room contains objects. Finally, the third and subsequent levels record items or states of each object, e.g., an object can store its current attributes.

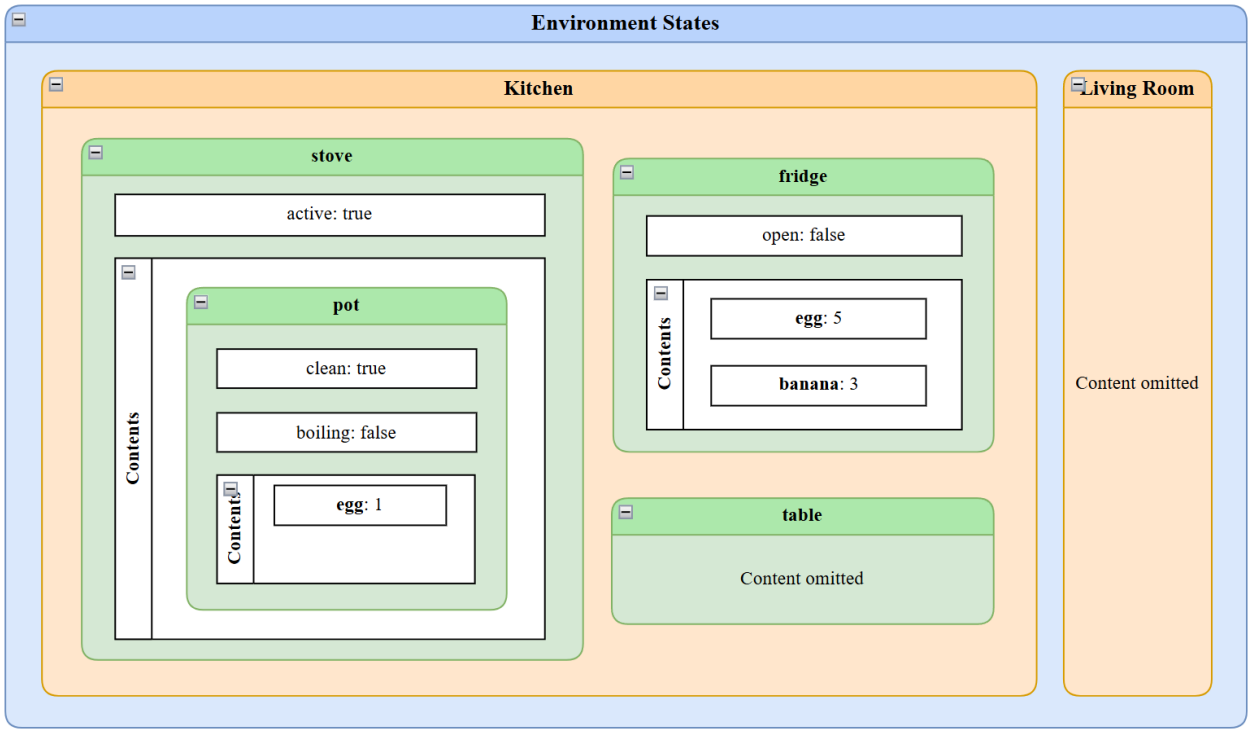

Figure 3.4. Environment States Components.

As illustrated in Figure 3.4, at the root, the Environment States is divided into logical locations, such as Kitchen and Living Room. Within each location, at the second level, the system stores all interactive objects such as the stove, fridge, and table. At the third level, each object stores its observable state attributes, such as whether a stove is active, and whether a pot is clean and boiling. Finally, some objects may function as containers and therefore may include a contents field. For example, a stove may have a pot as its contents, and a pot may have an egg inside.

This structure encodes objects, their spatial organization and state information simultaneously. NPC actions can be based on concrete objects and their current conditions in the environment.

3.2.2 *NPC Observable States*

Figure 3.5 shows the structure of NPC Observable States. This component records the externally visible state of each NPC that can be observed by other agents.

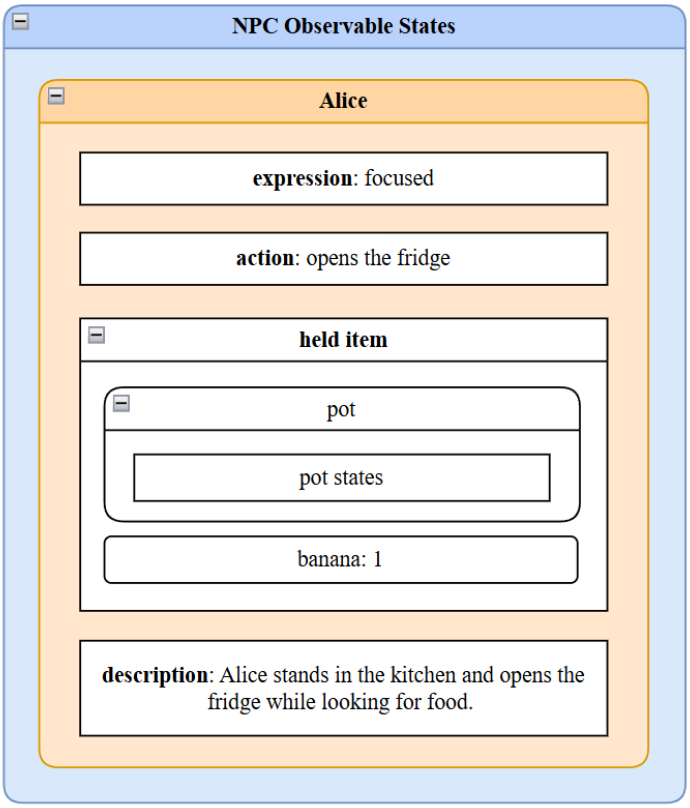

Figure 3.5. NPC Observable States Components.

As depicted in Figure 3.5, each NPC Observable State is organized as a hierarchical structure containing, at the first level, categories of observable states such as expression, current action, held item, and description. The expression field represents the NPC's visible emotional or facial state.

The action field records what the NPC is currently doing. The held item field records objects currently carried by the NPC. As illustrated in Figure 3.5, a held item such as a pot, can serve as a container and thus can be nested to contain other items. The description field provides a short visible description of the NPC's current situation.

The NPC Observable States module is essential for multi-agent interaction because it allows other NPCs and Players to respond to visible agent conditions. For example, an NPC can offer assistance when it notices that another NPC is in pain. An NPC can choose to join in if it notices another NPC is engaged in a certain task. An NPC can request a certain item if it notices another NPC is carrying that item.

3.2.3 *Player States*

Figure 3.6 shows the structure of Player States. Player States represent the human-controlled player within the same structured world model.

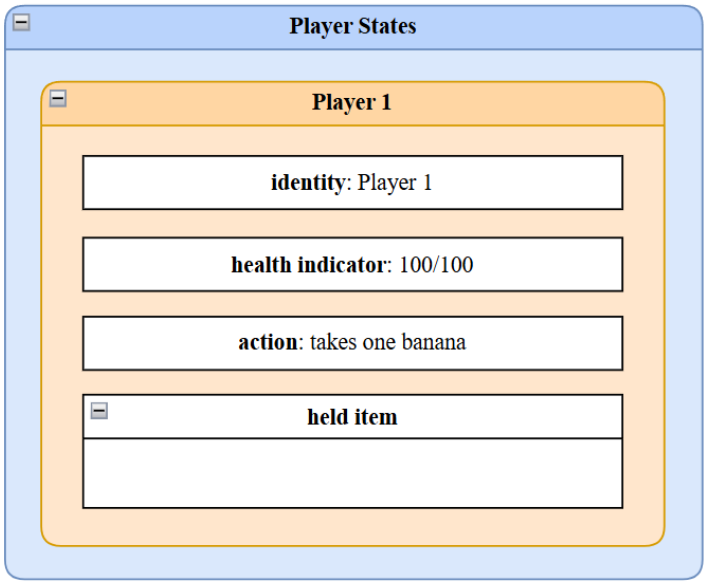

Figure 3.6. Player States Components.

Player States contain explicit player-related fields, such as identity, health indicator, action, and held item. These fields are updated from player input, system rules, or direct action effects. Unlike NPC Observable States, the Player States information is not inferred by an LLM. The framework does not infer the player's hidden intention, emotion, or internal condition.

By including Player States in Global World States, the system can connect player interaction with NPC reasoning. When the player performs actions in the world, the resulting state changes can be considered by NPCs in later action generation.

3.3 MEMORY MODULE

The Memory module stores each NPC's accumulated experience during the simulation. It provides historical context for later observable state and action generation.

In the proposed framework, Memory is NPC-specific, and the records are arranged according to simulation steps. A memory entry records the previous actions and consequences of the NPC, the other NPCs, and those of the player. Relevant environment changes during each simulation step are also recorded.

Memory entries are stored as compact natural language summaries rather than full copies of replay logs or world states. During subsequent behavior generation, Memory provides context about recent events and previous interactions. For example, an NPC may remember previously discussed topics, completed actions by other NPCs, and the ongoing changes in the environment. Such information supports NPCs to carry out coherent actions and the generation of plans based on what has already happened.

3.4 IDENTITY MODULE

The Identity module stores the stable character profile of each NPC and remains unchanged during the simulation. The module provides the character-level context for the generation of NPC behavior and observable states.

Similar to the Memory module, the proposed framework organizes the Identity independently for each NPC. Each NPC has an identity record that includes name, personality such as "social and optimistic", capabilities such as "outgoing, good at sports and cooking", family role such as "Alice's husband", occupation such as "PE teacher", and a brief description to explain their preferences or living habits.

The Identity module uses a compact task-oriented character schema rather than a complete psychological profile. This design is loosely grounded in McAdams and Pals' integrative personality framework, which describes personality as a combination of dispositional traits,

characteristic adaptations, and self-defining life narratives situated in culture and social context [30]. In this framework, personality corresponds to stable dispositional tendencies, capabilities correspond to task-related adaptations and abilities, family role and occupation correspond to the social roles most relevant to the household simulation, where the description provides a concise character summary comparable to a simplified narrative identity. Other dimensions in a full personality model, such as detailed psychometric trait scales, broader cultural background, or complete life history, are not included because of the lack of direct observability in the prototype environment. Extending the Identity module to include these richer psychological and cultural dimensions could be an interesting direction for future work.

During behavior generation, the Identity module provides a stable character context. For example, two NPCs may observe the same environment state but choose different actions because of their respective roles, personalities, or preferences. In this way, the Identity module maintains the consistency and uniqueness of a character.

3.5 Simulation Model

The simulation model defines the execution of the proposed framework over time. Instead of generating all NPC behaviors in a single continuous process, the simulation is divided into discrete ticks. Each tick represents a fixed interval of simulated time, and the system updates all the modules once during each tick.

Within each tick, the simulation follows a fixed turn order: Environment Turn, Player Turn, and NPC Turn. This order separates autonomous world changes, player actions, and NPC behavior generation into different phases. Each phase reads from and may potentially update the current Global World States. Therefore, subsequent phases in the same tick can be based on the state changes from prior phases.

The Environment Turn is executed first. It handles autonomous changes in the environment that are not directly caused by immediate agent actions. Autonomous environment changes are ongoing object processes that may continue or complete over time. For example, the boiling attribute of a teapot may change from false to true in a tick. This turn is necessary for environment-grounded decision making, because NPCs can only make reliable decisions based on the world state if the environment state itself is updated in a consistent and realistic way. The result of the Environment Turn is updated into the Environment States.

The Player Turn is executed next. Its purpose is to allow the human player to participate in the simulation as an in-world agent rather than only as an external source of instructions. If a player action is present, the system converts it into an action that follows the simulation's state-update process. Similar to NPC actions, the player action must operate within the existing world state and may produce direct effects on Player States and Environment States. The Player Turn allows explicit player input to affect the shared Global World States before NPCs generate their own actions. This is a simplified player-in-the-loop interaction model for testing how NPCs respond to player behavior within the same structured simulation world.

The NPC Turn is executed last in each tick. During this phase, each NPC generates actions and behaviors based on the described architecture. The NPC uses its Identity, Memory, and the current Global World States to update its own observable state, form a plan, generate an action and effects, and apply those effects back to the world state. Relevant action results are then recorded into the NPC's Memory module.

3.6 SUMMARY

The proposed framework is inspired by Generative Agents [12] and shares several of its architectural features. Both frameworks use LLM-driven characters that maintain character information and past experiences, form plans, generate actions, and update their context over time in a shared multi-agent simulation. The first distinctive extension of the proposed framework is a centralized and continuously updated Global World States representation, which provides current environment information throughout action generation and records the effects of generated actions as persistent state changes. The second is the explicit representation of NPC Observable States, which provides each NPC with the current visible conditions of other characters, allowing immediate non-verbal responsiveness and coordination around ongoing activities and shared objects.

In this thesis, a believable NPC is operationally defined as an NPC whose behavior appears human-like, coherent, character-consistent, and responsive to the surrounding simulation context. The NPC should behave in a way that is understandable to humans: its actions should correspond to its character identity, based on its previous experiences, respond to the current world state, and interact with other agents or the player in a contextually appropriate way.

The purpose of the proposed design is to produce believable NPCs in a shared world. It should enable different NPCs to interact with each other and the environment in a reasonable manner based on their own personality. The assessment criteria for believability will be detailed in the Results section.

Chapter 4. IMPLEMENTATION

This chapter describes the implementation of the proposed LLM-driven NPC framework as a local turn-based prototype. The chapter first introduces the development environment and project structure, then explains the tick-based simulation loop and the execution order of the Environment, Player, and NPC turns. The chapter then describes the NPC reasoning pipeline, the JSON data records used to store world states and intermediate execution results, the prompt and output-handling mechanisms used for LLM interaction, and the replay and analysis tools for inspecting generated behaviors.

4.1 Development Environment and Project Structure

4.1.1 *Development Environment*

The proposed framework repeatedly communicates between symbolic world states and LLM-generated decisions. The implementation required interfacing with the LLM through an API, preserving structured state across turns, constraining model outputs, and recording the resulting state transitions for subsequent inspection.

The prototype was implemented as a stand-alone Python-based system rather than as a real-time game engine. The Python system controls LLM interactions: calls with world-state contexts, validates returned results, and updates the simulation state. This fulfills the objective of examining the reasoning pipeline, not of creating a complete playable game. GPT-5.4 [31] was selected because of its reliability, stability, and run-time efficiency. Runtime data and replay snapshots were stored as local files such that the prototype could execute the simulation sequentially and continuously. Finally, a web-based replay interface was implemented for examining the recorded simulation.

4.1.2 *Project Structure*

The project structure is organized into data artifacts and executable code. The data artifacts store the simulation state in JSON-formatted files representing the current mutable state of the simulation, including the global world state, NPC-related records, player actions, environment

changes, and reflection records. These files are organized by tick to support time-sequenced visualization and post-run analysis.

The executable code defines the simulation controlling the per-tick execution order, including time advancement, environment updates, player updates, NPC reasoning, world-state updates, and replay export. The NPC reasoning is supported by a pipeline of I/O handling, model access, prompt construction, output standardization, and patch validation. The visualization and analysis tools operate on the exported replay data to support human inspection, behavior review, and consistency review.

4.2 TICK: SIMULATION TIME UNIT

A tick is the simulation time unit of the system. The system advances the simulated world by a tick, a fixed time interval, and executes the ordered update process. The tick therefore represents a discrete simulation step similar to that of a fixed-update frame in a real-time game engine.

The tick duration affects the granularity of generated actions. Too short a duration results in detailed physical sub-actions, such as raising a hand, placing it on the refrigerator handle, and pulling the door open. This level of motion decomposition is outside the intended scope of the prototype. An extended tick duration results in behaviors with an excessively low content on the time scale, such as spending three minutes to open the fridge. This will reduce the rationality of the behaviors of NPCs.

4.3 SIMULATION LOOP

The simulation is executed by a Python-controlled tick loop. The loop continuously advances the game-world time by tick duration, executes the Environment Turn, Player Turn, and NPC Turn in sequence, and stores the simulated results in a per-tick replay snapshot. Code Listing 4.1 shows the core control flow of the main loop.

Code Listing 4.1. Main Simulation Loop

```
Initialize runtime JSON files from initial data
Load data and NPC list
Export replay snapshot for Tick 0

For each tick until max_ticks:
    Advance tick number
    Advance simulated time
    Synchronize player presence setting
```

```
    Execute Environment Turn
    Execute Player Turn
    Execute NPC Turn
    Export replay snapshot for current tick
    If player mode is enabled:
        collect player input for the next tick
```

4.3.1 *Environment Turn*

The Environment Turn handles autonomous world changes before any player or NPC action is applied. The system passes recent Environment States history to the LLM, validates and updates the Global World States accordingly. Code Listing 4.2 shows the logic of the Environment Turn. A patch is a structured update instruction for JSON that specifies an existing state path and the new value for that path. The LLM first decides whether an environment-only change is required. If 'should_change' is false, no environment update is applied and the patch must be empty. Otherwise, the output must contain exactly one replace-only patch under 'Environment States'.

Code Listing 4.2. Environment Turn

```
Load recent Environment States history
Build environment auto-progress prompt
Send prompt to LLM
Parse output as JSON with:
    should_change, reason, patch

If should_change is false:
    require patch to be empty // Nothing needs to change
else:
    require one replace patch under Environment States
    validate patch on a copied Global World States
    apply validated patch to Global World States

Store result in EnvironmentAutoChanges.json
```

4.3.2 *Player Turn*

The Player Turn applies the structured player action generated and validated by LLM from the user's previous input to the Global World States. It will be enabled when the 'player mode' config is enabled. Code Listing 4.3 shows the core logic. 'result patch' exists means a valid user action exists in this turn, and it should have an impact on the Global World States.

Code Listing 4.3. Player Turn

```
Load current Player Actions
For each player action:
    If result patch exists:
        apply the patch to Global World States
```

```
After replay export:
    If player mode is enabled:  // Player participation is optional
        collect user input
        LLM generate current Player Actions
        validate actor, target, patch format, and allowed patch paths
        Store validated result in PlayerActions.json
```

4.3.3 *NPC Turn*

The NPC Turn executes a step-by-step LLM pipeline across the NPC set, including updating NPC memories; generating NPC observable states, plans, actions, and action results; resolving responses and conflicts; and finally updating the Global World States accordingly. Code Listing 4.4 shows the implemented order of the NPC pipeline. If the periodic reflection trigger condition is met, the IF branch under block A generates a reflected plan, and the IF branch under block B then skips normal plan generation. The detailed implementation of these pipeline steps is described in Section 4.4.

Code Listing 4.4. NPC Turn

```
// A: decide on reflection
If current tick reaches reflection interval:
    generate reflected plans
Update NPC memories
Generate NPC observable states
// B: generate or skip plans
If reflected plans were not generated:
    Generate NPC plans
Generate NPC actions
Resolve passive interactions
Generate NPC action results
Resolve action-result patch conflicts
Apply final NPC patches to Global World States
```

4.4 DETAILS OF NPC TURN

4.4.1 *Reflection*

Reflection is implemented as a periodic plan-revision support step in the NPC Turn. It asks the LLM to judge whether an NPC should continue or change its current direction, considering the NPC's identity, recent memory records, and visible world-state context. The reflection results will replace the NPC plan of this tick and be recorded in the NPC memory. The frequency was selected empirically during prototype testing. Experiments showed that different tick durations require different reflection intervals, and the practical range used in the prototype was approximately every

10 to 20 minutes of simulated time, which aligns with the intuition that humans do not constantly reflect on their own actions.

The purpose of reflection is to give NPCs an explicit opportunity to review their recent behavior and decide on a possible new direction. Although the normal planning step already uses the current world state, repeated LLM-based generation can still produce locally coherent but low-progress behavior patterns, such as repeatedly continuing the same conversation topic or repeatedly making small adjustments to the same object. The mechanism is inspired by the reflection component of Generative Agents [12] but serves only as a lightweight, periodic correction to generate short-term behavior.

4.4.2 *Details of NPC Turn Stages and LLM Interface*

This section focuses on the NPC reasoning and state-update process. Two implementation mechanisms are central to this process: the NPC turn stages and the LLM interface.

Code Listing 4.5 is a detailed breakdown of Listing 4.4 that summarizes the implemented stages of the NPC Turn. Stage A is the reflection generation stage introduced in Section 4.4.1. Stage B implements Step 4 in Figure 3.1 in the Design section, where recent action information and state changes are summarized into Memory. Stage C implements Step 1 in Figure 3.1, where Identity and Memory are used to update the NPC observable state in the Global World States. Stage D implements Step 2 in Figure 3.1, the Action module. Within Stage D, plan generation corresponds to Step 2a in Figure 3.2 in the Design Section. Action generation corresponds to Step 2b in Figure 3.2. Interaction handling allows NPCs to consider the interaction requests proposed by other NPCs and regenerate their actions. Action-result generation corresponds to Step 2c in Figure 3.2. Conflict handling checks incompatible result patches and replaces conflicting results with a non-conflicting waiting result when needed. Stage E implements Step 3 in Figure 3.1, where the final accepted action effects are applied to the Global World States.

Code Listing 4.5. NPC Turn Pipeline Stages

```
A: Reflection Generation if triggered
B: Memory Summarization
C: Observable-state Generation
D: Action Processing
      Plan generation:
         Generate and write plan and reason to NPCPlans.json
         If reflection is triggered, skip this step and use reflected plan
      Action Generation:
         Generate and write action, actor, target and reason to
```

```
            NPCCurrentActions.json
      Interaction Handling:
         Revise action if the NPC is targeted by another agent
      Action-result Generation:
         Generate, validate and write result.patch to NPCCurrentActions.json
      Conflict Handling:
         Replace conflicting action results with a non-conflicting waiting result
E: Update Global World States
```

Code Listing 4.6 abstracts the shared implementation pattern of the interface to the LLM in the NPC pipeline. In each NPC pipeline stage, the system first sends the required runtime records to the LLM via a step-specific prompt, then stores the parsed, checked, and validated JSON output in the corresponding runtime record. It applies to reflection generation, memory summarization, observable-state generation, plan generation, action generation, interaction handling, and action-result generation. These steps differ in their input records, prompt templates, required output fields, validators, and target JSON files, but they follow the same general execution structure.

Code Listing 4.6. LLM Interface of NPC Pipeline Stages

```
For each generation target:
    Load required records from StateStore
    Build a step-specific prompt:
        system instruction + JSON-formatted context from records
    Send the prompt to the LLM
    Parse the response as a JSON object
    Check required output fields
    Run a step-specific validator if needed
    Store the validated output in the target runtime JSON record
```

4.5 DATA RECORDS

The implementation uses multiple JSON files to separate long-lived simulation state from tick-level execution records. The persistent state of the simulation consists of three core files corresponding to the three main modules of the NPC architecture described in the Design section.

- ‘GlobalWorldStates.json’ is the general world-state file and stores the current time, environment states, observable agent states, and location records.
- ‘NPCMemories.json’ stores accumulated memory entries and provides historical context for later NPC reasoning.
- ‘NPCIdentities.json’ stores stable character definitions used as LLM context and remains unchanged throughout the simulation.

Unlike these three modules, the Action module does not correspond to a single persistent JSON file; instead, its generated outputs participate in the continuing simulation by being applied

to ‘GlobalWorldStates.json’ through Step 3 in Figure 3.1 and summarized into ‘NPCMemories.json’ through Step 4 in Figure 3.1.

Other files are independent of world models. They are execution records that either provide input to later pipeline steps or are folded back into the core files.

- ‘EnvironmentAutoChanges.json’ stores autonomous environment changes generated during the Environment Turn;
- ‘PlayerActions.json’ stores the structured player action generated from user input during the Player Turn. Result patches in these two files are applied to ‘GlobalWorldStates.json’, and the events are summarized into NPC memory.
- ‘NPCPlans.json’ stores the current short-term plan for each NPC and is used by the action-generation step in the same tick.
- ‘NPCCurrentActions.json’ stores the current NPC actions and their generated results; these results provide both world-state patches and memory content.
- ‘NPCReflections.json’ stores periodic reflection outputs, which may update the current NPC plan and later be integrated into memory.

Replay records are stored separately from the active runtime files. At the end of each tick, the system exports a replay snapshot that preserves the current Global World States together with the related NPC, player, environment, plan, action, memory, and reflection records. These replay JSON files are used by the HTML replay viewer and analysis scripts to inspect the simulation after execution.

4.6 LLM Interaction and Output Handling

4.6.1 *Prompt Construction*

Existing LLM-based game agent systems [12], [14] always use task-specific prompts rather than a single general prompt. Common prompt design patterns can be summarized from the prompts used by these systems: an effective prompt should contain the task to be performed, the constraints to be followed, the expected output form, and the context to be used.

The prototype implements this prompt design pattern by constructing each LLM call from two parts: a task-oriented system instruction and a JSON-formatted user context assembled from the runtime JSON records. The system instruction functions as the prompt template for different

LLM generation tasks. It defines the LLM's task in the current step, the constraints to be followed, and the expected output format. The JSON-formatted user context provides the step-specific simulation data to the LLM generation task.

4.6.2 *Output Handling*

LLM outputs may fail to return valid JSON, omit required fields, use unauthorized state modifications, refer to an object that does not exist in the current world state, or generate a patch that cannot be safely applied. These problems cannot be fully eliminated through prompting alone. Therefore, the implementation handles LLM outputs through JSON enforcement, task-specific validation, and retry. These mechanisms do not guarantee perfect semantic correctness, but they reduce structural errors and prevent many invalid outputs from entering the persistent simulation state.

Each LLM response is first parsed as a JSON object and checked against the required fields of the current prompt category, such as 'plan' and 'reason' for plan generation. When appropriate, additional validators are applied, such as checking whether a non-empty target exists in the global world state; and ensuring that generated patches use supported replace-only operations, point to valid existing paths, and preserve compatible held-item structures. If parsing or validation fails, the system retries by adding an error explanation to the prompt; after five failed attempts, an error is raised, and invalid output is not written into the runtime state.

4.7 Replay Visualization and Analysis Tools

To support inspection of multi-tick NPC behaviors and global world state changes over time, the project uses four complementary review methods: an HTML replay viewer for visual replay, Python analysis scripts for rule-based consistency verification, LLM-assisted semantic review for higher-level behavior interpretation, and manual verification. The inspection requirements, analysis targets, and interface design were specified by the author, while the code for the visualization and analysis tools was generated and iteratively revised with AI assistance.

The HTML replay viewer loads replay snapshots that contain runtime records that are exported at the end of each tick, and allows the user to move across ticks without rerunning the simulation. A screenshot of the replay viewer is shown in Figure 4.1. It provides a map-based view of the rooms, objects, agent positions, highlights changed environment objects, and displays panels

for Player State, Environment Auto Change, NPC Observable States, and selected JSON sections such as Global World States, NPC Current Actions, NPC Memories, and NPC Plans. This viewer supports direct inspection of whether the generated actions match the changing world state.

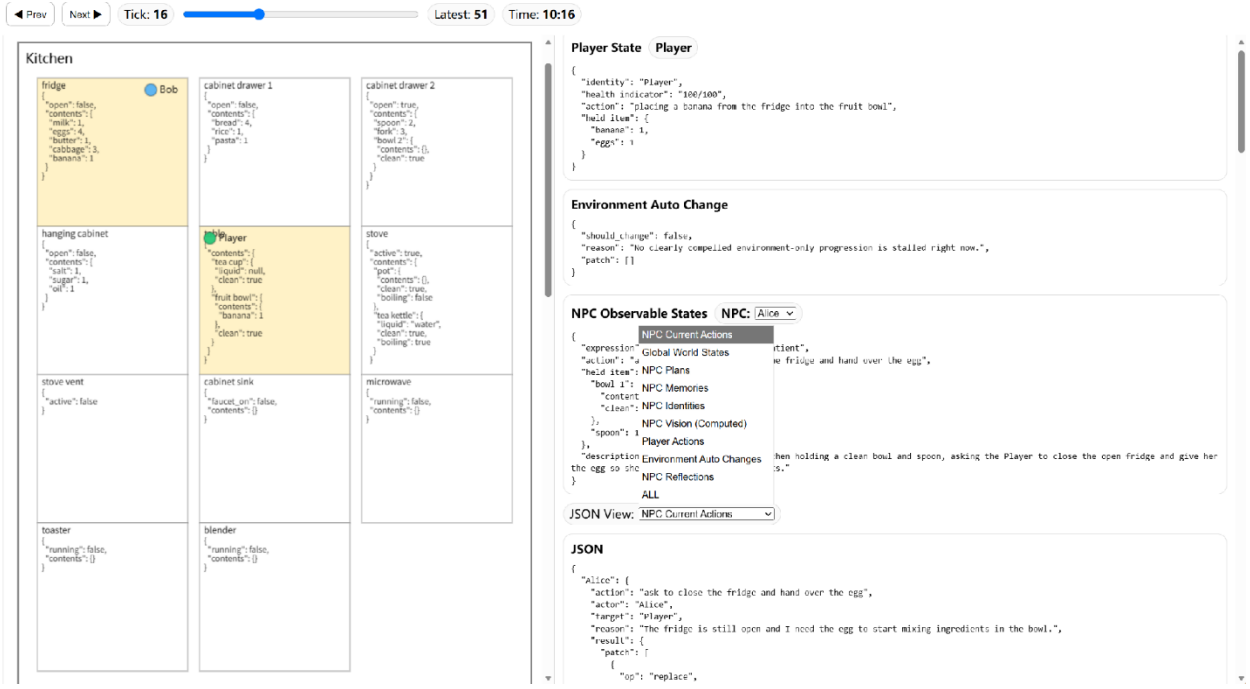

Figure 4.1. A screenshot of the HTML replay viewer.

The project also includes Python analysis scripts that read the exported replay JSON files and generate Markdown reports for later review. The replay audit script checks structural and behavioral signals, including item quantity and location changes, whole-container content replacements, patch conflicts, unauthorized NPC-state writes, clock mismatches, environment auto-change risks, NPC location changes, held-item changes, repeated behavior signals, and wait or pause statistics. The storyline report script produces a more readable timeline in which each row represents a tick, and each agent column shows that agent's current action. It also highlights position changes, environment keywords, and repeated actions, allowing an easier review of the broad storyline. A screenshot of the storyline report is shown in Figure 4.2. These scripts do not replace human judgment, but they reduce the amount of replay data that must be reviewed manually.

Tick	Time	Alice	Bob	Cathy
0000	10:00	room: Kitchen —	room: Workshop —	room: Living Room —
0001	10:01	stove vent stove Turn on the stove vent.	room: Workshop → Kitchen kitchen alice Walk to the Kitchen to join Alice and begin breakfast prep.	room: Living Room → Kitchen kitchen tea cup table walk to the kitchen table and pick up the tea cup
0002	10:02	fridge bob waits briefly because Bob is handling fridge	fridge Open the fridge door.	tea cup table Place the tea cup on the table.
0003	10:03	cabinet drawer 1 bread Open cabinet drawer 1 to get the bread.	kitchen fridge alice eggs Take the eggs out of the fridge and place them on the kitchen counter beside Alice.	cabinet drawer 1 alice waits briefly because Alice is handling cabinet drawer 1
0004	10:04	toaster bread put the bread into the toaster	fridge close the fridge door	fridge bob waits briefly because Bob is handling fridge
0005	10:05	cabinet drawer 1 cathy waits briefly because Cathy is handling cabinet drawer 1	stove table eggs pot Pick up the eggs from the table and crack them into the pot on the stove.	close the open cabinet drawer

Figure 4.2. A screenshot of the storyline report.

For behavior patterns that require semantic judgment, the replay viewer, audit reports, and storyline reports were also used as inputs for LLM-assisted semantic review. This review helped identify higher-level patterns such as collaboration between NPCs, task switching, routine continuation, and possible repetitive behavior. However, LLM-assisted review was used only as a supplementary inspection method. The final interpretation is the result of manual verification, because an LLM may misread the replay context or infer behavior that is not directly supported by the recorded simulation data.

Chapter 5. RESULTS

This chapter evaluates whether the proposed framework can produce believable NPC behavior in the simulated domestic scenario. Section 5.1 defines the evaluation criteria based on the design objectives introduced in Chapter 3, and Section 5.2 presents the scenario and simulation configuration used for the experiments. The detailed evaluation is then organized into four main parts. Section 5.3 examines coarse-grained human-like behavior over an extended simulation period, focusing on whether the NPCs form understandable daily routines and transition reasonably between activities. Section 5.4 evaluates four specific framework functionalities in fine-grained experiments: environment-grounded decision-making, coherent multi-step tasks, multi-NPC interaction, and collaborative behavior. Section 5.5 evaluates character consistency from multiple simulations. Section 5.6 evaluates whether the Player can participate in the shared world through ordinary interaction and collaboration with NPCs, and whether NPCs can respond appropriately to unreasonable Player actions. Finally, Section 5.7 summarizes the findings and discusses the limitations of the evaluation.

5.1 Evaluation Criteria

As defined in Chapter 3, the goal of the proposed framework is to produce believable NPCs in a simulated world. In this thesis, a believable NPC is defined as an NPC whose behavior appears human-like, coherent, character-consistent, and responsive to the surrounding simulation context. Therefore, the evaluation focuses on whether the implemented system can generate NPC behavior that is grounded in the current world state, remains coherent over time, is consistent with the character's identity, and is responsive to other NPCs and the Player.

The evaluation criteria were derived from two complementary sources. First, prior NPC believability studies were used to identify recurring behavioral dimensions. These studies associate believable NPCs with personality and emotion, understandable goals, social relationships, appropriate behavior within the game world, adaptation to changing circumstances, perception of other characters' actions, and participation in a changing social context [4], [32], [33]. Second, relevant LLM-driven simulation systems were examined to identify approaches for manifesting abstract dimensions as observable behavior. Results from Generative Agents [12] demonstrated

daily routines, planning and reflection across time, conversations between agents, and coordination around shared social events, while those from Humanoid Agents [13] showed that needs, emotions, and relationship closeness can produce differences in agents' daily activities and conversations.

Based on these results, the following behavior objectives and evaluation criteria are derived.

- Coarse-grained human-like behavior evaluates whether individual actions form a plausible daily pattern;
- NPC decisions based on environment states evaluate whether behavior is situated in and responsive to the current world;
- Coherent multi-step tasks evaluate whether goals and previous actions are maintained over time;
- Multi-NPC interaction evaluates responses to other agents' observable behavior;
- Collaborative behavior evaluates coordination toward a shared activity; and
- Character consistency evaluates whether behavioral and emotional tendencies remain aligned with identity.

Together, these criteria cover the human-like, coherent, character-consistent, environment-responsive, and socially responsive behavior specified in Chapter 3. The next step in this study is to create appropriate scenarios to observe the behavior objectives above and verify the criteria.

5.2 Scenario Design and Simulation Configuration

The evaluation is based on simulating a domestic scenario in which three NPCs lead their daily activities. This setting was selected because ordinary household life naturally includes independent activities, object usage, multi-step tasks, conversations, and collaborations. The resulting behavior is straightforward to examine and interpret, where activities such as cooking, working, resting, and interacting with family members can be readily judged for reasonableness and realism.

5.2.1 *Simulation Configuration*

Each experiment begins at 10:00 AM on a Sunday. Sunday morning provides a reasonable context for all three NPCs to remain at home and freely engage in household, work, leisure, and social activities.

The tick duration was configured according to the purpose of each simulation run. For coarse-grained experiments that simulate a longer period of daily activity, each tick represented three

minutes of simulated time. In short-term experiments that required more fine-grained behavior inspection, each tick represented one minute of simulated time. These values were selected empirically based on experiences from prototype experimentation. Simulation runs were fixed at a 200-tick duration, limited by the amount of data to be generated and analyzed. The selected duration yields sufficient data to observe and verify multi-step behaviors and daily routines, while keeping the volume of collected data manageable for analysis.

5.2.2 *Environment Configuration*

The simulated world consists of a Kitchen, a Workshop, and a Living Room. The Kitchen allows opportunities for food preparation, eating, cleaning, and shared household tasks. The Workshop offers opportunities for work and practical activities with objects such as a laptop, notebook, books, a whiteboard, and a toolbox. The Living Room supports resting, entertaining, exercising, hobby-based activities, and informal social interactions.

Each room contains relevant ordinary objects. For example, the Kitchen contains food, containers, utensils, appliances, a stove, a sink, and a table, while the Living Room contains a sofa, a television, a guitar, and exercise equipment. Together, the three rooms provide sufficiently diverse activity contexts to evaluate the targeted behaviors while maintaining a compact, manageable environment.

5.2.3 *NPC Configuration*

The scenario consists of three typical adult family-member NPCs representing a typical household setting. Alice and Bob are married, and Cathy is Alice's mother. Alice is a systematic thinker and a software engineer, Bob is a social and optimistic physical education teacher, and Cathy is an easygoing supermarket employee who prefers quiet hobbies. Figure 5.1 shows the identities defined for the three NPCs.

The scenario begins with Alice in the Kitchen, Bob in the Workshop, and Cathy in the Living Room. Note that in the following experiments, the NPCs changed their locations early in the simulations. Three NPCs are sufficient for observing individual behavior, one-to-one interactions, and group collaborations.

```
"Alice": {
  "name": "Alice",
  "personality": "Highly curious and systematic",
  "capabilities": "Smart, good at music and computer",
  "family role": "Bob's wife",
  "occupation": "Software Engineer",
  "description": "Alice loves planning and debugging, but can become impatient with vague statements. She prefers
    use tools to think. She likes to keep the environment organized, and help others with practical advice."
},
"Bob": {
  "name": "Bob",
  "personality": "Social and optimistic",
  "capabilities": "Outgoing, good at sports and cooking",
  "family role": "Alice's husband",
  "occupation": "PE Teacher",
  "description": "Bob likes to exercise lightly to stay energetic. He gets distracted easily by comfort, but
   responds well to collaboration."
},
"Cathy": {
  "name": "Cathy",
  "personality": "Easygoing and quietly creative",
  "capabilities": "Patient, enjoys crafts and small creative hobbies",
  "family role": "Alice's mother",
  "occupation": "Supermarket staff",
  "description": "Cathy has a relaxed personality and enjoys spending time on small hobbies such as knitting,
   cooking new dishes, arranging flowers, or listening to music. She likes to stay ina peaceful environment
   rather than getting involved in what others are doing."
}
```


Figure 5.1. NPC Identities Used in the Experiments.

5.3 Coarse-Grained Experiments

Coarse-grained experiments examine whether NPCs can follow an understandable, reasonable daily behavioral pattern over a meaningful period. The simulation consists of 200 consecutive ticks, with each tick representing three minutes. The complete run therefore covers ten simulated hours, from 10:00 to 20:00.

Table 5.1 summarizes the full run by grouping the 200 ticks into nine major activity periods. The simulation begins with Alice and Bob preparing tea and bread while Cathy plays the guitar independently. Cathy later joins them, and the three NPCs move to the Living Room for a shared morning break. Around midday, their attention shifts to lunch preparation, followed by the meal and cleanup. During the afternoon, the NPCs alternate between resting in the Living Room and working together in the Workshop. In the early evening, they return to the Kitchen to prepare for dinner, dining together, and clean the remaining items. Table A.1 in Appendix A presents part of the original NPC actions from the same run.

Table 5.1. Summary of the 200-Tick Coarse-Grained Behavior Timeline

Timeline	Activity
10:00-10:54	**Morning tea preparation**: Alice and Bob prepared tea and bread in the kitchen, while Cathy moved from playing guitar alone to joining them.
10:54-12:21	**Morning break**: The three NPCs gathered in the living room for tea, bread, conversation, guitar playing, and TV watching, completed a stretching session, and shifted their attention toward lunch.
12:21-12:54	**Lunch preparation**: The three NPCs cooperated in the kitchen to prepare eggs and toast for lunch.
12:54-13:48	**Lunch and after lunch cleanup**: The three NPCs ate lunch together, cleaned up after lunch at the kitchen table while maintaining a casual conversation.
13:48-14:39	**Early-afternoon break**: The group rested in the living room, watched television, played guitar, and held a short sing-along.
14:39-15:42	**Workshop collaboration**: The three NPCs held a collaborative brainstorming and note-taking session in the workshop.
15:42-17:45	**Extended afternoon break**: The group returned to the living room for an extended period of TV watching, conversation, guitar playing, and light exercise.
17:45-18:39	**Dinner preparation**: The NPCs moved to the kitchen and cooperated in preparing a simple dinner of eggs, cabbage, and bread.
18:39-20:00	**Dinner and after dinner cleanup**: The three NPCs ate dinner, cleaned the dinner items together and continued with a conversation at the kitchen table.

The summarized timeline indicates that the generated activities are plausible, coherent, and consistent with typical daily routines. The NPCs engaged in ordinary activities, including food preparation, eating, cleaning, work, rest, entertainment, exercise, and conversation. Their activities also changed over time: individual morning behavior developed into shared activity, meals were followed by cleanup and rest, and the group later resumed work before preparing dinner. Although some activities continued for relatively long periods, the NPCs did move on and proceed with subsequent tasks. The run includes several transitions between rooms, activity types, and levels of social participation. These observations verify that the framework supports coarse-grained behavior that resembles an understandable day at home.

5.4 Fine-Grained Experiments

These experiments are designed to verify that the framework supports environment-grounded decisions, coherent multi-step tasks, multi-NPC interactions, and collaborative behaviors. The results were observed and verified from the same simulation, where each tick represented one minute.

Table 5.2 summarizes the NPC actions generated during the first 15 ticks of a simulation. In the beginning, the three NPCs gather in the Kitchen preparing food. Alice turns on the stove vent,

retrieves bread from the cabinet drawer, and places it in the toaster. She later prepares a spoon, receives the toasted bread from Bob, places it on the table, and retrieves salt from the hanging cabinet. Bob joins Alice, takes eggs from the fridge, and cracks the eggs into the pot. While the eggs are cooking, he removes the toasted bread, hands it to Alice, checks the eggs, turns off the burner, and moves the pot to the table. Cathy takes the teacup, closes the drawer after Alice removes the bread, retrieves a bowl from the cabinet drawer, and uses the spoon to transfer a boiled egg from the pot into the bowl. By 10:15, the three NPCs had prepared toasts and boiled eggs and set out the basic tableware for the meal. The following subsections examine the different functionalities demonstrated in this segment.

Table 5.2. NPC Actions During the First 15 Minutes (15 ticks) of a Simulation

Tick	Time	Alice	Bob	Cathy
0001	10:01	Turn on the stove vent.	Walk to the Kitchen to join Alice and begin breakfast prep.	walk to the kitchen table and pick up the tea cup
0002	10:02	waits briefly because Bob is handling fridge	Open the fridge door.	Place the tea cup on the table.
0003	10:03	Open cabinet drawer 1 to get the bread.	Take the eggs out of the fridge and place them on the kitchen counter beside Alice.	waits briefly because Alice is handling cabinet drawer 1
0004	10:04	put the bread into the toaster	close the fridge door	waits briefly because Bob is handling fridge
0005	10:05	waits briefly because Cathy is handling cabinet drawer 1	Pick up the eggs from the table and crack them into the pot on the stove.	close the open cabinet drawer
0006	10:06	waits briefly because Bob is handling toaster	pushes the toaster lever down to start the bread toasting	open cabinet drawer 2
0007	10:07	take a spoon from the open cabinet drawer 2	turns the stove burner under the pot up to start boiling the eggs	waits briefly because Alice is handling cabinet drawer 2
0008	10:08	place the spoon on the table	check the boiling eggs in the pot	take bowl 1 from the open drawer
0009	10:09	close the drawer	remove the toast as it pops up	place the clean bowl on the table
0010	10:10	take the toast from Bob	hands the toast to Alice	pick up the spoon from the table
0011	10:11	waits briefly because Cathy is handling table	waits briefly because Cathy is handling stove	Scoop one boiled egg from the pot on the stove into bowl 1 with the spoon.
0012	10:12	waits briefly because Cathy is handling table	wait beside the stove for Cathy to finish scooping the eggs	places the spoon on the kitchen table
0013	10:13	place the toast on the kitchen table	turns off the burner and lifts the pot of eggs off the stove	step away from the table to clear space for the toast
0014	10:14	Open the hanging cabinet to get the salt.	set the pot of eggs on the kitchen table beside bowl 1	waits briefly because Bob is handling table
0015	10:15	take the salt from the hanging cabinet and set it on the table beside bowl 1	waits briefly because Alice is handling table	close the hanging cabinet door

5.4.1 *NPC Decisions Based on Environment States*

Table 5.3 shows the detailed states of relevant objects in the simulated world: fridge, table, and stove from 10:01 to 10:05. Yellow highlights indicate that an object state has changed. The

sequence shows the fridge door opening and closing, and the eggs moving from the fridge to the table and then into the pot.

Table 5.2 includes several decisions that are based directly on the states of these objects. Bob's handling of the eggs provides a clear example. At 10:02, the eggs were stored in the fridge with the door closed, so Bob first opened the fridge door. At 10:03, Bob removed the eggs from the fridge and placed them on the Kitchen table. Because the fridge remained open after the transfer, Bob closed it at 10:04. At 10:05, the eggs were located on the table, so Bob moved them from the table into the pot on the stove. In these ways, Bob's decisions and actions are based on and consistent with the current state of the fridge and the eggs. It is verified that NPC decisions were based on the current Global World States and that the direct effects of their actions were written back into the shared state, consistent with Steps 2 and 3 in Figure 3.1 of the Design section.

Table 5.3. Environment State Transitions During 10:01-10:05

	10:01	**10:02**	**10:03**	**10:04**	**10:05**
fridge	{ "open": false, "contents": { "milk": 1, "eggs": 6, "butter": 1, "cabbage": 3, "banana": 3 } }	{ "open": true, "contents": { "milk": 1, "eggs": 6, "butter": 1, "cabbage": 3, "banana": 3 } }	{ "open": true, "contents": { "milk": 1, "butter": 1, "cabbage": 3, "banana": 3 } }	{ "open": false, "contents": { "milk": 1, "butter": 1, "cabbage": 3, "banana": 3 } }	{ "open": false, "contents": { "milk": 1, "butter": 1, "cabbage": 3, "banana": 3 } }
table	{ "contents": { "fruit bowl": { "contents": {}, "clean": true } } }	{ "contents": { "fruit bowl": { "contents": {}, "clean": true }, "tea cup": { "liquid": null, "clean": true } } }	{ "contents": { "fruit bowl": { "contents": {}, "clean": true }, "tea cup": { "liquid": null, "clean": true }, "eggs": 6 } }	{ "contents": { "fruit bowl": { "contents": {}, "clean": true }, "tea cup": { "liquid": null, "clean": true }, "eggs": 6 } }	{ "contents": { "fruit bowl": { "contents": {}, "clean": true }, "tea cup": { "liquid": null, "clean": true } } }
stove	{ "active": true, "contents": { "pot": { "contents": {}, "clean": true, "boiling": false }, "tea kettle": { "liquid": "water", "clean": true, "boiling": false } } }	{ "active": true, "contents": { "pot": { "contents": {}, "clean": true, "boiling": false }, "tea kettle": { "liquid": "water", "clean": true, "boiling": false } } }	{ "active": true, "contents": { "pot": { "contents": {}, "clean": true, "boiling": false }, "tea kettle": { "liquid": "water", "clean": true, "boiling": false } } }	{ "active": true, "contents": { "pot": { "contents": {}, "clean": true, "boiling": false }, "tea kettle": { "liquid": "water", "clean": true, "boiling": false } } }	{ "active": true, "contents": { "pot": { "contents": { "eggs": 6 }, "clean": true, "boiling": false }, "tea kettle": { "liquid": "water", "clean": true, "boiling": false } } }

5.4.2 *Coherent Multi-Step Tasks*

The storyline from Table 5.2 shows several activities developed through successive steps. For example, Bob's egg-preparation sequence began at 10:01 with opening the fridge, removing the eggs, placing them on the table, closing the fridge, and transferring them into the pot. He then turned on the stove burner, checked the eggs, waited while Cathy was using the stove area, turned off the burner, lifted the pot, and finally placed it on the table at 10:14. Each of these actions built on the results of earlier actions and advanced the task.

Additionally, Bob interleaved this sequence with toast preparation. At 10:06, he started the toaster after Alice had inserted the bread. He later removed and handed the toasted bread to Alice, and returned his attention back to the eggs. The two activities progressed in parallel with coherent contexts that are consistent over time.

These examples show that the NPCs could preserve task progress over time and generate coherent subsequent follow-up actions. This supports the intended use of Memory to maintain the recent context during simulation cycles, which corresponds to Step 4 in Figure 3.1 of the Design section.

5.4.3 *Multi-NPC Interactions*

The details from Table 5.2 include both direct exchanges and action adjustments caused by another NPC's current activity. For example, Bob went to the kitchen at 10:01, aiming to join Alice's ongoing activity. At 10:02, Alice waited because Bob was already handling the fridge. At 10:10, Bob handed the toast to Alice. At 10:13, Cathy stepped aside to allow Alice to place the toast on the table. Table 5.4 presents the NPC Observable States during this time period, demonstrating that NPCs maintain awareness of and respond appropriately to the behaviors of other NPCs.

These examples show that an NPC could respond to the expression, current action, and the held item of other NPCs. This supports the intended flow in which NPC Observable States are updated into the Global World States and being considered during subsequent action and interaction processing, corresponding to Step 1 in Figure 3.1 of the Design section.

Table 5.4. NPC Observable State Changes

Time	NPC Observable States

10:01	"Alice": {
	"expression": "Attentive and composed",
	"action": "Turning on the stove vent",
	"held item": {},
	"description": "Alice appears neatly put-together, scanning her surroundings with a focused, curious look."
	},
	"Bob": {
	"expression": "Warm smile",
	"action": "Walking to the Kitchen to join Alice and begin breakfast prep.",
	"held item": {},
	"description": "Bob appears relaxed and upbeat, standing with easy energy and an open posture."
	},

10:02	"Alice": {
	"expression": "Focused and mildly intent",
	"action": "waits briefly",
	"held item": {},
	"description": "Alice originally intended to Open the fridge to get the breakfast ingredients., but paused because Bob was already handling fridge."
	},
	"Bob": {
	"expression": "cheerful and attentive",
	"action": "Opening the fridge door.",
	"held item": {},
	"description": "Bob stands in the kitchen by the fridge with its door open, ready to get the eggs for breakfast prep."
	},

10:09	"Alice": {
	"expression": "attentive, slightly impatient",
	"action": "closes cabinet drawer 2",
	"held item": {},
	"description": "Alice stands in the kitchen with empty hands after closing cabinet drawer 2, looking toward the stove and toaster."
	},
	"Bob": {
	"expression": "Focused and upbeat",
	"action": "removes the toast from the toaster",
	"held item": {
	"toast": 1
	},
	"description": "Bob stands in the kitchen by the toaster, having just removed the toast while keeping an eye on the boiling eggs."
	},

10:10	"Alice": {
	"expression": "Attentive and slightly impatient",
	"action": "takes the toast from Bob",
	"held item": {
	"toast": 1
	},
	"description": "Alice stands in the kitchen holding the finished toast after taking it from Bob, ready to place it on the table next."
	},
	"Bob": {
	"expression": "Cheerful and focused",
	"action": "hands the toast to Alice",
	"held item": {},
	"description": "Bob stands in the kitchen by the toaster and stove with empty hands after handing the toast to Alice, still alert and engaged in breakfast prep."
	},

10:13	"Alice": {
	"expression": "slightly impatient but composed",
	"action": "places the toast on the kitchen table",
	"held item": {},
	"description": "Alice stands in the kitchen after placing the toast on the table, with empty hands and ready to get the salt."
	},
	"Cathy": {
	"expression": "Calm and focused",
	"action": "steps away from the table to clear space for the toast",
	"held item": {},
	"description": "Cathy stands in the kitchen away from the table with empty hands, calmly making room for Alice to set down the toast."
	}

5.4.4 *Collaborative Behaviors*

Table 5.2 shows that in these 15 minutes, Alice handled the bread, toast, table items, and salt. Bob retrieved and cooked the eggs, operated the toaster, and moved the pot. Cathy prepared the cup,

bowl, and spoon and transferred an egg into the bowl. These actions together contribute to the food-preparation activity.

Note that the sequence includes dependencies of actions in a collaboration. From 10:03 to 10:13, Alice took the bread from the cabinet and inserted it into the toaster before Bob started it. Bob removed the toast and handed it back to Alice. Cathy moved away from the table so that Alice could place the toast.

Within the 15-minute segment, the three NPCs collectively prepared toast, eggs, utensils, and seasoning. This is a result of coordinated actions connected through shared Environment States, NPC Observable States, NPC memory, and decision making. This demonstrates that the framework is capable of supporting collaborative behavior that develops organically within a shared activity context.

5.5 Multi-Run Experiments: Character Consistency

Character consistency is the degree to which an NPC's behavioral tendencies remain stable and align with the Identity settings across simulation runs. The evaluation focuses on recurring patterns rather than isolated actions, because character consistency is more clearly reflected in how an NPC repeatedly approaches work, rest, interaction, and shared household tasks over time.

Activities consistent with the NPCs' defined identities can be observed across different simulation runs. For example, during activities such as food preparation, Alice and Bob frequently worked together. Alice often continued performing her own task while directing the subsequent activity or asking another NPC to handle a specific object. When her progress was delayed by waiting or unclear coordination, her observable expression was more likely to become impatient. These patterns are consistent with Alice's systematic, organized, and practically helpful identity, as well as her stated impatience with vague situations. Bob more often responded to Alice's requests and completed the requested tasks. He also regularly took on primary responsibility for cooking and was usually involved in the cleanup afterward. These are consistent with his cooking capability, collaborative tendency, and social personality. Cathy showed a different pattern. In some runs, she joined Alice and Bob after an activity had already begun and contributed to the shared task. While in other runs, she remained in the Living Room and continued with her personal activity. This reflects her easygoing personality and her preference for quiet hobbies. These divisions of labor also conform to the family structure composed of these three roles.

Additional experiments used deliberately exaggerated personality settings to determine whether changing Identity would produce clearly different long-term behavior patterns. A patriarchal husband rarely participated in washing dishes after meals, whereas Bob was usually involved in the subsequent cleanup. A traditional homemaker continued working for extended periods after her husband and father had begun resting, whereas Alice rarely remained alone in household work after the others had stopped. A religious father repeatedly prayed and praised the other characters but did not contribute to household tasks, whereas Cathy was generally quieter and frequently participated in practical work when she chose to join an activity.

These comparisons show that the NPCs followed behavioral patterns that are consistent with their defined Identities. The NPCs developed recurring tendencies that corresponded to their stated personalities, capabilities, family roles, and preferences. Significantly different identity settings produced visibly different patterns under the same type of domestic scenario. These results indicate that identity influences behavior not only at the level of individual actions, but also in repeated choices about leadership, cooperation, household work, rest, social interaction, and participation over time.

5.6 Player Interaction Experiments

The Player interaction experiments examine whether a human-controlled Player can participate in the same shared world as the NPCs and whether NPCs can respond to Player actions in a contextually appropriate manner. Player input was used to introduce situations that would be unlikely to occur naturally during an ordinary household simulation, allowing the framework to be tested under both routine and unusual conditions.

Table 5.5 presents NPC and Player actions from one simulation run containing several Player interventions. The table is divided into three segments: ordinary interaction between the Player and NPCs, collaboration involving the Player and multiple NPCs, and NPC responses to unreasonable Player actions.

5.6.1 *Ordinary Interaction Between NPCs and the Player*

In Table 5.5, ticks 3-7 show two ordinary requests made by the Player. The Player first asked Bob for a banana. Bob responded by opening the fridge, retrieving a banana, following the Player after the Player moved away, and finally handing the banana to the Player. The Player also asked Cathy

for the guitar. Because the guitar was already beside Cathy, she was able to hand it directly to the Player. Both NPCs correctly identified the requested object and completed the requested transfers. These examples show that the Player can initiate ordinary requests and receive responses in ways similar to those from other NPCs.

Table 5.5. NPC and Player Actions from a Player-Interaction Simulation

Tick	Time	Alice	Bob	Cathy	Player
Ordinary interaction between NPCs and the player					
0003	10:03	wait attentively by the stove	open the fridge	remain seated and rest quietly	ask Bob "can you take one banana for me?"
0004	10:04	wait by the stove	take a banana from the fridge	answers that she's just resting on the sofa	walk to the Living Room and say "Hello! what are you doing?" to Cathy
0005	10:05	close the fridge door	walk to the living room holding the banana	pick up the guitar from the sofa	ask Cathy "can you give me the guitar?"
0006	10:06	stand by quietly near the kitchen table	hand over the banana	hand over the guitar	wait
0007	10:07	remain standing quietly by the kitchen table	say a friendly follow-up to continue the conversation	sit down and lean back to rest	take banana from Bob
Collaboration between multiple NPCs and the player					
0015	10:15	take the bowl and spoon from Bob	hand the bowl and spoon to Alice	set the remote down and remain seated relaxing	ask Alice to take the egg from Player's hand
0016	10:16	ask to close the fridge and hand over the egg	close the fridge door	remain seated and rest quietly on the sofa	take a banana from the fridge
0017	10:17	take the banana from the Player while keeping her attention on the egg next	hold out his hands to receive the egg from the Player	pick up the remote from the coffee table	hand the banana to Alice
0018	10:18	reach out to take the egg from Bob into her free hand	pass the egg to Alice as soon as he receives it	sit quietly holding the remote	hand the egg to Bob
NPC responses to unreasonable Player Actions					
0024	10:24	tell Bob to stop and hold off on the egg while checking if he's okay	step back from the Player and say that punch was not okay and he needs a moment	make a brief concerned comment asking everyone to calm down while staying seated on the sofa	punch Bob in the face
0025	10:25	ask Bob if he wants to sit down for a minute and keep the cooking paused	stand beside Alice and tell her he just needs a minute to steady himself	firmly tell the Player to put the dumbbells down and take a breath	pick up the dumbbells
0026	10:26	tell the Player to take the dumbbell out of the microwave and not use it	step back from the microwave and tell Alice he needs another moment	call out more firmly for the Player to stop touching the microwave and calm down immediately	walk to the kitchen and place the dumbbell into the microwave
0027	10:27	open the microwave and pull the dumbbell out	tell the Player to stop the microwave and remove the dumbbell immediately	shout for the Player to stop the microwave and remove the dumbbell immediately	press the microwave start button
0028	10:28	Refuse to hand over the dumbbell and tell the Player to step back.	firmly tells the Player not to take the dumbbell and to back off	firmly tells the Player not to take the dumbbell and to leave it with Alice	ask Alice to hand over the dumbbell
0029	10:29	keep hold of the dumbbell and watch the Player from the kitchen doorway	stand beside Alice and keep watch toward the living room doorway	remain seated on the sofa and verbally warn the Player to keep their distance and stay calm	say "ok" to Alice and walk to the Living Room
0030	10:30	set the dumbbell down on the kitchen table	remain at the kitchen doorway and take a few slow breaths while keeping watch on the living room	warns the Player to stop there and keep a safe distance while staying seated on the sofa	—
0043	10:43	sets the dumbbell down securely on the Workshop floor away from the doorway and calls out that Bob can start helping reset the kitchen	quietly tell Cathy he is steady enough and that the kitchen is calm, while staying seated at the table.	quietly ask if he is steady and if the kitchen remains calm	—
0045	10:45	Alice asks Bob to clear bowl 1 into cabinet drawer 2 while she answers Cathy that the reset is under control.	put bowl 1 away for Alice	quietly ask if she needs help finishing the kitchen reset while staying at the doorway	—

5.6.2 *Collaboration Between Multiple NPCs and the Player*

In Table 5.5, ticks 15-18 show the Player participating in a shared task involving Alice and Bob. The Player repeatedly asked Alice to take the eggs and banana currently held by the Player. Alice

responded to the request while also reminding the Player that the fridge had been left open. Bob then closed the fridge, received the eggs from the Player, and immediately passed them to Alice.

This sequence shows that Player participation can become part of an ongoing multi-agent activity. The NPCs did not treat the Player's actions as an isolated external command. Instead, they incorporated the requests and the items held by the Player into the shared task, responded to an environment-state issue caused by the Player, and continued with the object handoff across multiple agents.

5.6.3 *NPC Responses to Unreasonable Player Actions*

The final segment in Table 5.5 introduces a sequence of unreasonable Player actions. At Tick 24, the Player suddenly punched Bob. All three NPCs immediately responded to the event: Alice showed concern for Bob, Bob pushed the Player away, and Cathy asked everyone to remain calm.

At Tick 25, the Player picked up a dumbbell in the Living Room, and Cathy attempted to stop the Player verbally. At Tick 26, the Player placed the dumbbell inside the Kitchen microwave. Alice and Cathy warned the Player, while Bob moved away from the appliance. At Tick 27, the Player started the microwave. Bob and Cathy demanded that the player stop while Alice turned off the microwave and removed the dumbbell. At Tick 28, the Player asked Alice to return the dumbbell. Alice refused, and Bob and Cathy also responded in support of the refusal.

After Tick 30, the Player left the scene and the NPCs began dealing with the aftermath. They remained cautious for approximately ten minutes. After approximately thirteen minutes, they confirmed the situation with one another and moved the dumbbell out of the Kitchen. Approximately fifteen minutes after the Player left, the NPCs returned to normal activity and began working on a Kitchen project.

These responses were both contextually appropriate and temporally coherent. The NPCs did not treat each disruptive Player action as an isolated event. Their reactions escalated as the Player's behavior became more dangerous, remained consistent across the three NPCs, and continued after the Player had left. Subsequent caution, discussion, removal of the dumbbell, and gradual return to normal activity show that the consequences of the event persisted over multiple ticks rather than being reduced to a single immediate response.

5.7 SUMMARY

The experiments demonstrate that the proposed framework can produce the behavioral properties included in the operational definition of believable NPCs under the tested scenario. In the ten-hour coarse-grained simulation, the NPCs formed a recognizable daily routine that includes food preparation, meals, cleanup, work, rest, and social activity, with comprehensible transitions between the events. The fine-grained and Player interaction experiments showed that the NPCs could maintain connected tasks across ticks, responding and collaborating with other NPCs and players based on the environment and the other agents' states while aligning with their own identities. Together, these results demonstrate human-like, temporally coherent, and character-consistent behaviors within the scope of a domestic life simulation.

The results also provided direct evidence supporting the main design features of the framework.

- The explicit and continuously updated Environment States grounded NPC decisions in existing objects and their current conditions. Other existing systems rely on memory-based experience records and therefore lack explicit mechanisms for NPCs to consistently determine the current availability, condition, or recent changes of objects within a dynamically changing environment [12], [13], [29]. In the proposed framework, the current Environment States are directly provided during NPC reasoning and are continuously updated by action effects. In contrast, as detailed in Section 5.4.1, the NPCs opened containers before retrieving objects, and their subsequent actions were informed by the recorded movement and state changes of those objects.
- The explicit NPC Observable States supported social responsiveness. Other existing systems do not explicitly represent another NPC’s current action, held objects, or visible expressions, limiting their ability to support immediate, non-verbal responses based on these observable conditions. In contrast, as detailed in Sections 5.4.3 and 5.4.4, the NPCs waited for occupied objects, responded to expressions and ongoing actions, transferred held items, shared tasks, and completed a shared activity.
- The Memory supported continuity across multiple ticks, while Identity enabled persistent differences in leadership, cooking, cleanup, rest, and social participation. Other existing systems do not combine memory and character information with the continuously updated

Environment States and explicit NPC Observable States. In contrast, as detailed in Sections 5.4.2, 5.4.3, 5.4.4, and 5.5, the NPCs continued their activities over multiple ticks and adjusted their subsequent actions in response to environmental changes and the observable conditions of other NPCs, while maintaining identity-consistent behavior.

Finally, the results presented in Section 5.6 showed that Player States and Player Actions could be integrated into the same shared world model. These observations are made within a carefully composed environment and are insufficient to support the conclusion that the framework provides a general solution for believable NPC behavior. However, they provide evidence of the proposed framework's ability to integrate identity, memory, environmental information, observable agent states, and structured state updates, as described in Chapter 3, to enable the targeted behaviors.

However, the evaluation remains limited in scale and scope. The framework is not limited to the three NPCs, three rooms, and objects used in the experiments. Currently, expansions require explicit element design and manual additions. Increasing the number of NPCs would also increase the number of LLM calls, prompt size, execution time, and experimental cost. The current results demonstrate and verify the proposed functionality only under the evaluated domestic configuration, rather than establishing its behavior across a broad range of scenarios. In addition, the framework depends on LLM-generated observable states, plans, actions, and effects, and thus also suffers from the occasional LLM glitches such as hallucinations.

Future evaluation should apply the same framework to larger NPC groups, more varied identities and environments, longer simulation runs, and should also include user studies or formal measurements of behavioral quality.

Chapter 6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This thesis investigated how LLM-driven NPCs can generate believable behavior that remains grounded in a changing simulation environment and responsive to other agents. It focuses on two directions: grounding ordinary NPC decisions in an explicit and continuously updated world representation and supporting multi-NPC interaction through the observable states of other agents. It also represents the Player as part of the shared simulation context.

To address these issues, this thesis proposed an LLM-driven NPC framework organized around four modules: Action, Global World States, Memory, and Identity. The Global World States module represents the shared simulation world with a hierarchical tree. Within this structure, Environment States simultaneously encode the existence of interactable objects, their spatial organization, and their current states. NPC Observable States record NPC's visible conditions explicitly, allowing other NPCs to observe these conditions and reason about appropriate responses and interactions. Player States represent the human-controlled Player within the same structured world model and connect the player interaction with NPC reasoning.

The Action module connects these state representations to NPC behavior through Plan, Decision, and Effects. An NPC first forms a short-term plan based on its Identity, Memory, and world state information. The plan is then converted into a concrete action that refers to existing agents or objects in the simulation. Direct changes resulting from the action are then applied to the Global World States via structured state-update operations. Relevant actions and consequences are subsequently summarized into NPC Memory. This closed loop allows later decisions to be based on what has actually occurred in the simulated world.

The proposed framework was implemented as a stand-alone Python prototype with a tick-based simulation loop. Each tick executes the Environment Turn, Player Turn, and NPC Turn in a fixed order. The evaluation results demonstrated the main capabilities of the proposed framework in a domestic scenario. In the ten-hour coarse-grained simulation, the NPCs formed a comprehensible daily routine involving food preparation, meals, cleanup, work, rest, and social activity, with coherent transitions between the activities. The fine-grained experiments provided direct evidence for the targeted functionalities: NPC decisions followed the current states and

locations of objects, multi-step tasks remained connected across ticks, and NPCs responded to one another's observable actions, shared objects, and requests while coordinating toward common activities. Additional runs revealed recurring behavioral differences generally consistent with the defined Identities. The Player experiments further showed that NPCs could respond to ordinary Player requests, collaborate with the Player on shared tasks, and maintain coherent reactions to unreasonable Player actions over multiple ticks.

Overall, the results support the two main contributions of this thesis. First, the experiments show that incorporating explicit and continuously updated world-state information into NPC decision-making can support actions that remain grounded in the current availability, locations, and states of objects in the simulated environment. Second, the experiments show that explicitly representing NPC observable states can support multi-NPC responsiveness and coordination based on the visible conditions of other agents. These findings support the use of structured world states and explicit observable-agent representations as a practical foundation for believable LLM-driven NPC simulation.

6.2 Future Work

Future work can extend both the framework and its evaluation in several directions.

First, the Memory module should support summarization, selective retrieval, and forgetting or compression. The current chronological memory records will continue to grow during long simulations, while the lightweight reflection mechanism only revises short-term plans at fixed intervals. A more advanced design could consolidate older experiences, retrieve only contextually relevant memories, and adaptively trigger reflection in response to important events, repeated behaviors, or completed goals.

Second, the simulation should support more flexible temporal granularity. A fixed tick duration cannot represent fine-grained actions and long-duration activities equally well. Future systems could combine variable-duration actions, event-driven updates, or multiple temporal levels for physical actions, conversations, environmental processes, and long-term planning.

Third, the framework should introduce agent-specific perception. The current Global World States provide one shared observable state for all NPCs. As a result, all observers receive the same visible description. A future design could separate objective, observable cues from observer-

specific interpretations, allowing different NPCs to interpret the same expression or action differently according to their identities, memories, relationships, and current situations.

Fourth, the Identity module could be expanded beyond its current set of character attributes. Additional dimensions, such as detailed personality traits, values, preferences, cultural background, life history, needs, moods, and relationships, could support more nuanced and persistent differences in NPC behavior. A dynamic Identity model could also be incorporated into the framework, allowing selected character attributes to evolve gradually through long-term experiences, relationships, and significant events.

Fifth, future work should evaluate the framework under broader and more systematic experimental conditions. The current straightforward domestic environment does not establish that the framework will maintain the same level of robustness in substantially different scenarios, such as larger NPC populations or outdoor environments. The Player experiments are also limited and do not establish robustness under more diverse situations, such as sustained adversarial behavior. In addition, the current evaluation is entirely qualitative and subjective. The operational definition of believability used in this thesis does not constitute an empirical measurement of player-perceived believability. Future evaluation should include larger and more diverse scenarios, longer simulation runs, multiple independent evaluators or user studies, and human ratings of behavioral coherence and believability. These evaluations would provide stronger evidence regarding both the generalizability of the framework and the quality of the behaviors produced.

Finally, beyond the current game-oriented simulation, another future direction is to investigate whether the proposed framework can be adapted to other situated multi-agent domains in which autonomous agents must reason about a changing environment and respond to the observable conditions of other agents. This direction could be explored in other virtual multi-agent environments, embodied-agent simulations, and robotic systems involving human-robot or multi-robot interaction. Such extensions would require domain-specific mechanisms that are not addressed in the current work.

BIBLIOGRAPHY

[1] P. García-Sánchez, "Georgios N. Yannakakis and Julian Togelius: Artificial Intelligence and Games: Springer, 2018, Print ISBN: 978-3-319-63518-7, Online ISBN: 978-3-319-63519-4, https://doi.org/10.1007/978-3-319-63519-4," *Genet Program Evolvable Mach*, vol. 20, no. 1, pp. 143–145, Mar. 2019, doi: 10.1007/s10710-018-9337-0.

[2] "Using Machine Learning to Increase the Fidelity of Non-Player Characters in Training Simulations | CMU Software Engineering Institute." Accessed: May 22, 2026. [Online]. Available: https://www.sei.cmu.edu/blog/using-machine-learning-to-increase-the-fidelity-of-non-player-characters-in-training-simulations/

[3] J. Togelius and G. N. Yannakakis, "General general game AI," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, Santorini, Greece: IEEE, Sep. 2016, pp. 1–8. doi: 10.1109/CIG.2016.7860385.

[4] H. Warpefelt and H. Verhagen, "A model of non-player character believability," *Journal of Gaming & Virtual Worlds*, vol. 9, pp. 39–53, Mar. 2017, doi: 10.1386/jgvw.9.1.39_1.

[5] H. Warpefelt, *The non-player character: exploring the believability of NPC presentation and behavior*. in DSV Report Series, no. 16– 003. Stockholm: Department of Computer and Systems Sciences, Stockholm University, 2016.

[6] A. B. Loyall, "Believable agents: building interactive personalities," 1997. Accessed: Apr. 29, 2026. [Online]. Available: https://www.semanticscholar.org/paper/Believable-agents:-building-interactive-Loyall/570b0e89c4b33a2bf0101fe7397cf58e325f7890

[7] J. J. Kim, "Bringing Personality to Pixels, Inworld Levels Up Game Characters Using Generative AI," NVIDIA Blog. Accessed: May 23, 2026. [Online]. Available: https://blogs.nvidia.com/blog/generative-ai-npcs/

[8] "How Ubisoft's New Generative AI Prototype Changes the Narrative for NPCs." Accessed: May 23, 2026. [Online]. Available: https://news.ubisoft.com/en-gb/article/5qXdxhshJBXoanFZApdG3L/how-ubisofts-new-generative-ai-prototype-changes-the-narrative-for-npcs

[9] S. Hu *et al.*, "A Survey on Large Language Model-Based Game Agents," Jun. 08, 2026, *arXiv*: arXiv:2404.02039. doi: 10.48550/arXiv.2404.02039.

[10] I. Millington, *Artificial intelligence for games*. in The Morgan Kaufmann series in interactive 3D technology. Amsterdam Heidelberg: Elsevier Morgan Kaufmann, 2006.

[11] S. R. Cox and W. T. Ooi, "Conversational Interactions with NPCs in LLM-Driven Gaming: Guidelines from a Content Analysis of Player Feedback," in *Chatbot Research and Design*, vol. 14524, A. Følstad, T. Araujo, S. Papadopoulos, E. L.-C. Law, E. Luger, M. Goodwin, S. Hobert, and P. B. Brandtzaeg, Eds., in Lecture Notes in Computer Science, vol. 14524. , Cham: Springer Nature Switzerland, 2024, pp. 167–184. doi: 10.1007/978-3-031-54975-5_10.

[12] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, San Francisco CA USA: ACM, Oct. 2023, pp. 1–22. doi: 10.1145/3586183.3606763.

[13] Z. Wang, Y. Y. Chiu, and Y. C. Chiu, "Humanoid Agents: Platform for Simulating Human-like Generative Agents," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Y. Feng and E. Lefever, Eds.,

Singapore: Association for Computational Linguistics, Dec. 2023, pp. 167–176. doi: 10.18653/v1/2023.emnlp-demo.15.
[14] G. Wang *et al.*, “Voyager: An Open-Ended Embodied Agent with Large Language Models,” Oct. 19, 2023, *arXiv*: arXiv:2305.16291. doi: 10.48550/arXiv.2305.16291.
[15] H. Warpefelt, *Towards an updated typology of non-player character roles*. 2015.
[16] M. Treanor, J. McCoy, and A. Sullivan, “Social Play in Non-Player Character Dialog,” *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 11, no. 4, pp. 99–101, Nov. 2015, doi: 10.1609/aiide.v11i4.12838.
[17] M. Dawe, S. Gargolinski, L. Dicken, T. Humphreys, and D. Mark, “Behavior Selection Algorithms,” in *Game AI Pro 360*, 1st ed., S. Rabin, Ed., CRC Press, 2019, pp. 1–14. doi: 10.1201/9780429055058-1.
[18] C. Guckelsberger, C. Salge, and J. Togelius, “New And Surprising Ways to Be Mean. Adversarial NPCs with Coupled Empowerment Minimisation,” Jun. 04, 2018, *arXiv*: arXiv:1806.01387. doi: 10.48550/arXiv.1806.01387.
[19] M. Guimarães, P. A. Santos, and A. Jhala, “Emergent social NPC interactions in the Social NPCs Skyrim mod and beyond,” Jan. 20, 2023, *arXiv*: arXiv:2207.13398. doi: 10.48550/arXiv.2207.13398.
[20] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” Jul. 22, 2020, *arXiv*: arXiv:2005.14165. doi: 10.48550/arXiv.2005.14165.
[21] L. Ouyang *et al.*, “Training language models to follow instructions with human feedback,” Mar. 04, 2022, *arXiv*: arXiv:2203.02155. doi: 10.48550/arXiv.2203.02155.
[22] L. Wang *et al.*, “A Survey on Large Language Model based Autonomous Agents,” *Front. Comput. Sci.*, vol. 18, no. 6, p. 186345, Dec. 2024, doi: 10.1007/s11704-024-40231-1.
[23] S. Yao *et al.*, “ReAct: Synergizing Reasoning and Acting in Language Models,” Mar. 10, 2023, *arXiv*: arXiv:2210.03629. doi: 10.48550/arXiv.2210.03629.
[24] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language Agents with Verbal Reinforcement Learning,” Oct. 10, 2023, *arXiv*: arXiv:2303.11366. doi: 10.48550/arXiv.2303.11366.
[25] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, “Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents,” Jul. 08, 2024, *arXiv*: arXiv:2302.01560. doi: 10.48550/arXiv.2302.01560.
[26] Z. Wang *et al.*, “JARVIS-1: Open-World Multi-task Agents with Memory-Augmented Multimodal Language Models,” Nov. 30, 2023, *arXiv*: arXiv:2311.05997. doi: 10.48550/arXiv.2311.05997.
[27] R. Volum *et al.*, “Craft an Iron Sword: Dynamically Generating Interactive Game Characters by Prompting Large Language Models Tuned on Code,” in *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, M.-A. Côté, X. Yuan, and P. Ammanabrolu, Eds., Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 25–43. doi: 10.18653/v1/2022.wordplay-1.3.
[28] S. Rao *et al.*, “Collaborative Quest Completion with LLM-driven Non-Player Characters in Minecraft,” Jul. 03, 2024, *arXiv*: arXiv:2407.03460. doi: 10.48550/arXiv.2407.03460.
[29] Z. Kaiya *et al.*, “Lyfe Agents: Generative agents for low-cost real-time social interactions,” Oct. 03, 2023, *arXiv*: arXiv:2310.02172. doi: 10.48550/arXiv.2310.02172.
[30] D. P. McAdams and J. L. Pals, “A new Big Five: Fundamental principles for an integrative science of personality,” *American Psychologist*, vol. 61, no. 3, pp. 204–217, 2006, doi: 10.1037/0003-066X.61.3.204.

[31] "GPT-5.4 Model | OpenAI API," OpenAI Developers. Accessed: Aug. 04, 2026. [Online]. Available: https://developers.openai.com/api/docs/models/gpt-5.4
[32] M. Lee and C. Heeter, "What do you mean by believable characters?: The effect of character rating and hostility on the perception of character believability," *Journal of Gaming & Virtual Worlds*, vol. 4, pp. 81–97, Mar. 2012, doi: 10.1386/jgvw.4.1.81_1.
[33] H. Warpefelt, M. Johansson, and H. Verhagen, "Analyzing the believability of game character behavior using the Game Agent Matrix," in *Proceedings of DiGRA 2013 Conference*, 2013. doi: 10.26503/dl.v2013i1.692.

APPENDIX A

Table A.1 presents part of the original NPC actions from the coarse-grained experiments in Section 5.3. The entries are organized by the activity periods in Table 5.1 and show representative actions at key transition points.

Table A.1. Tick-Level NPC Actions from the 200-Tick Coarse-Grained Simulation

Tick	Time	Alice	Bob	Cathy
10:00-10:54 Morning tea preparation				
0001	10:03	Tell Bob the stove is on needlessly and switch off the burner.	Walk to the kitchen and offer to help Alice make tea or a quick snack.	Sit down on the sofa and pick up the guitar to begin softly strumming.
0006	10:18	answers Cathy that she can share some and pauses stirring to acknowledge her	Ask if she'd like him to fetch cups or snacks for Cathy as she arrives.	Walk into the kitchen and quietly ask for a cup of tea or to share some.
0007	10:21	Ask Bob to fetch another cup and a spoon for Cathy.	Open cabinet drawer 2 and take 1 spoon for Cathy.	Respond softly to Alice and stay beside her while waiting for the tea to be handed over.
0011	10:33	puts the sugar into the hanging cabinet and tells Bob bananas or bread would be fine	Ask if they'd like a light snack such as bananas or bread with the tea.	Set the bowl and spoon down on the kitchen table.
0018	10:54	waits briefly while Bob handles table	pick up the sliced bread from the kitchen table to carry it to the living room	Carry her bowl and bread into the living room and sit down on the sofa for tea.
10:54-11:21 Morning break				
0019	10:57	walks to the living room to join the tea-and-bread break	Carry the sliced bread to the living room and place it on the Coffee Table.	Sip quietly from the bowl while seated on the sofa.
0027	11:21	pick up the guitar from the sofa and give it a light test strum while staying seated	keep chatting while seated and hold the bread after finishing the bite	continue chatting quietly while seated on the sofa
0032	11:36	set the guitar down on the Sofa and thank Bob as the TV comes on	waits briefly while Alice handles TV	Eat the small piece of bread while remaining seated on the sofa and chatting softly.
0040	12:00	Place the remote on the coffee table and suggest a short stretch or moving to the workshop desk.	stand up from the sofa, agree to Alice's stretch idea, and head toward the treadmill for a light stretch break	Make a brief warm comment about the show while staying seated on the sofa.
0046	12:18	agree with Bob and stand from the workshop desk to head with him toward the kitchen for lunch	Suggest heading to the kitchen together now for an easy lunch.	watches the TV from the sofa
12:21–12:54 Lunch preparation				
0049	12:27	Take the butter and milk out of the fridge for lunch prep.	ask Cathy to grab the bread from the living room coffee table for lunch	Head to the living room to take the bread from the coffee table for lunch.
0055	12:45	ask if the eggs are ready so they can plate lunch and eat	turns off the stove burner under the pot of eggs and tells Alice they are ready to plate.	pick up the spoon from the table to start setting the lunch items neatly
0056	12:48	waits briefly while Bob handles pot	carry the pot of eggs from the stove to the kitchen table	waits briefly while Bob handles table
0057	12:51	Tell Bob she will plate the egg first, then join them at the table.	Invite Alice and Cathy to sit at the table and start lunch.	set the spoon on the kitchen table beside the toast
0058	12:54	transfer the egg from the pot into bowl 2 and set the bowl on the table	sit down at the kitchen table to start lunch	sit down at the kitchen table and wait quietly for the egg to be plated
12:54–13:48 Lunch and Post-lunch cleanup				
0060	13:00	waits briefly while Cathy handles bowl 2	Invite Alice and Cathy to start eating with him while taking another bite of his toast.	stand up and carry the milk and bowl 2 from the table to the sink area
0063	13:09	Eat the eggs from bowl 2 while making light conversation.	Take another bite while chatting casually about lunch.	chat quietly with Alice and Bob while staying seated at the table
0070	13:30	wipe her hands on a kitchen towel and sit down at the table to join the conversation	Stand up from the table and open the fridge to take out a banana.	stand up and carry the milk and bowl 2 from the table to the sink area
0072	13:36	continue chatting from her seat at the table	Peels the banana and takes a few bites.	move the milk from beside the cabinet sink into the fridge
0075	13:45	stand up and tell Bob she can join him in the living room after a moment.	Stand up and suggest moving the conversation to the living room for a more relaxed break.	Stand up and carry bowl 2 from the cabinet sink to the table side neatly.

Table A.1. Tick-Level NPC Actions from the 200-Tick Coarse-Grained Simulation (continued)

Tick	Time	Alice	Bob	Cathy
13:48–14:39 Early-afternoon break				
0077	13:51	sit down on the sofa beside Cathy and continue the brief chat.	waits briefly while Alice handles Sofa	remain seated and watch the TV quietly
0080	14:00	pick up the guitar from the coffee table.	reply with a brief joking comment to Cathy while staying seated and watching TV.	make a brief light comment about the TV while staying seated on the sofa
0083	14:09	Softly strum the guitar and give Bob a brief amused sung reply from the sofa.	keep softly singing along while making a lighthearted comment to Alice about the tune	continue softly singing along while seated and relaxed on the sofa
0087	14:21	Suggest a short chat break while staying seated on the sofa.	leans back on the sofa and briefly chats with Alice while glancing at the TV	sit quietly and watch the screen as the singing trails off.
0091	14:33	turns off the TV with the remote and	stands up from the sofa and agrees to head	Keeps watching the TV from the sofa and
14:39–15:42 Workshop collaboration				
0094	14:42	uses the laptop at the desk to start the	Suggest Cathy jot down the first points as	Pick up a pen and jot down the first few
0095	14:45	types a few key points into the laptop to organize the group's notes.	Offer another concrete idea for Cathy to add to the notebook.	Write down Bob's suggested idea in the notebook.
0101	15:03	summarizes the captured notes aloud with Bob and Cathy	Discuss the notebook ideas with Alice and Cathy and agree on the strongest options as she summarizes.	quietly scan the notebook pages while listening to the review discussion.
0110	15:30	gesture toward the laptop and notebook and suggest wrapping up with a quick glance before taking a short break in the living room	Step to the desk and point at the notebook while agreeing to do a quick final review before the living room break.	sit down at the desk and rest her hands near the notebook while listening quietly
0112	15:36	keeps suggesting a short break in the living	agrees with Alice and suggests Cathy join	Write a brief follow-up note in the
15:42–17:45 Extended afternoon break				
0117	15:51	presses the remote to turn the TV on while staying seated on the sofa	continue chatting casually while staying seated on the sofa	continue chatting casually while seated on the sofa
0130	16:30	sets the guitar on the coffee table and leans back into the sofa to keep watching TV.	make a warm, easygoing comment about Alice's gentle playing while staying seated on the sofa	make a soft warm comment about how nice Alice’s guitar sounds while staying seated on the sofa
0140	17:00	stand up from the sofa and join Bob for a quick stretch break by the treadmill	waits briefly while Alice handles Alice	remain seated on the sofa and keep quietly watching the TV as the conversation eases off.
0145	17:15	waits briefly while Bob handles Bob	walks with Alice to the sofa and sits down, agreeing to relax with TV or quiet chat.	rolls up the yoga mat and sets it aside near the sofa.
0150	17:30	sets the remote back on the coffee table	make another light comment about the TV	stay seated on the sofa and keep watching
17:45–18:39 Dinner preparation				
0158	17:54	Tell Cathy to check the bread in cabinet drawer 1 and see what table items are useful for dinner.	Close the fridge and suggest making eggs with cabbage for dinner, asking Cathy to grab bread or utensils.	Open cabinet drawer 1 and check for bread to bring out for dinner.
0162	18:06	Ask Cathy to place the fork on the table and close cabinet drawer 2.	Ask if eggs-and-cabbage with bread sounds good for dinner.	sets the fork on the table, then closes cabinet drawer 2.
0165	18:15	waits briefly while Bob handles stove	Place the egg on the stove and turn it on to start cooking.	pick up the butter from the kitchen table and place it neatly with the dinner items Alice is setting out
0168	18:24	Ask Bob to turn on the stove vent.	turns on the stove vent while staying by the stove to keep tending the egg.	sets the cabbage on the kitchen table beside the bread.
0172	18:36	Transfer the cooked egg from the stove into	chat casually while waiting for the egg to	sit down at the kitchen table and wait
18:39–20:00 Dinner and Post-dinner cleanup				
0173	18:39	Sit down at the kitchen table and begin eating from bowl 2 with the spoon.	Sit down at the kitchen table and gesture for Alice to join dinner.	waits briefly while Alice handles bowl 2
0181	19:03	take a small piece of bread from the table and eat it	Take the bread from cabinet drawer 1 and begin eating at the table.	take another small bite of bread while speaking softly
0190	19:30	sets the spoon down in the cabinet sink	waits briefly while Cathy handles cabinet sink	Pick up bowl 2 and the fork from the kitchen table and carry them to the sink area.
0196	19:48	sit down at the kitchen table and start a brief chat	pulls out a chair and sits down at the kitchen table to continue chatting	Pick up bowl 2 and the fork from the kitchen table and carry them to the cabinet sink area.
0200	20:00	Place the fork into cabinet drawer 2 and close the drawer.	Stand up and carry bowl 2 from the cabinet sink to the kitchen table.	waits briefly while Bob handles bowl 2