UNIVERSITY OF SOUTHERN DENMARK

MASTER'S THESIS

Conversational AI in Games: An Empirical Study of LLM-Powered NPC Dialogue Systems

Author:
Anthon K. S. PETERSEN (494680),
Emil RIMER (481870),
Rasmus PLOUG (497505)

Supervisor:
Marco SCIREA

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in Engineering

in the

SDU Metaverse Lab
The Maersk Mc-Kinney Moller Institute

Character Count: ~160.488

July 19, 2025

UNIVERSITY OF SOUTHERN DENMARK

Abstract

The Faculty of Engineering
The Maersk Mc-Kinney Moller Institute

Master of Science in Engineering

Conversational AI in Games: An Empirical Study of LLM-Powered NPC Dialogue Systems

by Anthon K. S. PETERSEN, Emil RIMER, Rasmus PLOUG

Non-player character (NPC) dialogue plays a crucial role in narrative-driven video games. NPCs help shape the player experience by contributing to narrative, immersion, and agency. Large language models (LLMs) are currently being explored for game development, but despite recent advancements in dynamic LLM-generated dialogue for NPCs, few empirical studies have compared how they impact player experience across different dialogue system designs.

This study explores how LLM-driven dialogue systems affect the player experience. A prototype game was developed in the Unity Game Engine as a custom test environment. It is a role-playing game (RPG) focused on gameplay through dialogue. The game was made in four different versions, each featuring a different dialogue system design. These versions are: *Control Version* (no LLM), *Version A* (Dialogue rephrasing), *Version B* (Fixed open-ended dialogue options), and *Version C* (Fully open-ended dialogue).

A user study was conducted with a total of 64 participants, each playing one version of the game. Behavioral data was collected, and all participants completed a post-game questionnaire. The results indicate that *Version C* featured significantly longer dialogue interactions and higher overall engagement compared to the other versions. Participants rated the NPCs in *Version C* as the most interesting to talk to. Casual players in particular highlighted *Version C* and *Version B* as contributing more to immersion, agency, and flexibility. However, participants also noted that the dialogue could occasionally be inconsistent or lacking in relevant information.

The findings suggest that a fully open LLM-driven dialogue system has potential to enhance the player experience by improving narrative, immersion, and overall involvement in NPC interactions.

Contents

List of Figures

List of Tables

List of Abbreviations

NPC	**N**on-**P**layer **C**haracter
LLM	**L**arge **L**anguage **M**odel
AI	**A**rtificial **I**ntelligence
RPG	**R**ole-**P**laying **G**ame
TTS	**T**ext-**T**o-**S**peech
STT	**S**peech-**T**o-**T**ext
UI	**U**ser **I**nterface
API	**A**pplication **P**rogramming **I**nterface

Chapter 1

Introduction

The use of large language models (LLMs), with the introduction of OpenAI's ChatGPT [1], [2], has made its way into the world of video game development. The possibilities for applying LLMs to games are almost limitless, making them incredibly versatile. This has created a domain that is not only novel, but also rapidly expanding due to the pace and advancement of these technologies. The role of artificial intelligence (AI) in video games is still unclear, and the current focus is primarily on its ability to generate content. However, less focus is being given to research approaches that explore not only how it can be applied, but also how it can enhance the player's experience. LLMs naturally falls into to the domain of dialogue, making them especially relevant for interactions between players and AI through in-game conversations.

This project is motivated by the growing interest in implementing LLMs in video games, particularly to enhance dialogue systems. While LLMs are increasingly being explored in this context, gaps can still be found in the literature regarding how their features can be effectively applied. The concept of using LLMs in games is currently a popular research domain, making it relevant to explore this area and contribute new findings. Although this work is academic in nature, its exploratory approach may help identify principles for designing NPC dialogue that meaningfully influences player experience.

This thesis contributes to ongoing research on LLMs in games by introducing a testbed for systematically comparing dialogue systems and their impact on player interaction. The testbed is implemented as a game prototype, featuring familiar elements such as NPCs, dialogue, and narrative progression. Four distinct dialogue systems are embedded within the same game environment and tested on participants with prior gaming experience. The resulting data are analyzed and discussed to evaluate differences in player engagement and system performance.

1.1 Research Questions

This study adopts an exploratory approach, and to guide the project, the following research questions were developed:

- RQ1: How does the integration of different LLM-driven features in NPC dialogues influence player interaction in video games?
- RQ2: Did any LLM-driven feature enhance player interaction compared to the Control Version?

The questions focus on various ways to integrate LLM-driven features into NPC dialogues. They also consider the impact of LLM-driven player interaction features

on the overall player experience. Finally, the study examines how these LLM-based dialogue systems influence player experience within the developed test environment and how their effects may differ across other types of games and genres.

1.2 Thesis Structure

This thesis is divided into several chapters. Below is an overview of each chapter and its content:

- Chapter 2: Related Works – Reviews existing literature on traditional dialogue systems in games, procedural dialogue generation with LLMs, and NPCs in video games. It also identifies gaps in the current literature.
- Chapter 3: Methodology – Describes the approach used for prototype design, outlines the methods for different dialogue system variants, and explains how these systems are implemented.
- Chapter 4: User Study – Describes the design of the user study, including data collection and analysis methods.
- Chapter 5: Results – Presents descriptive statistics, quantitative and qualitative results, and in-game player responses.
- Chapter 6: Discussion – Summarizes findings, interprets results, discusses study limitations, and examines LLM behavior and variability.
- Chapter 7: Conclusions – Summarizes key findings, answers the research questions, and offers suggestions for future work and recommendations.

1.2.1 Use of AI in the Writing Process

LLMs, such as ChatGPT, were used throughout the writing process to help refine wording and improve flow. The structure, content, sources, claims and academic decisions were made by the authors. The LLM was used purely as a writing support tool, not as a co-author.

Chapter 2

Related Works

2.1 Large Language Models

LLMs have rapidly advanced in recent years, with examples such as OpenAI's ChatGPT [1], [2], Google's Gemini [3], Meta's LLaMA [4], and Microsoft's Phi-3 [5]. As these models became accessible to the public, it was only a matter of time before developers began exploring the vast possibilities of integrating LLMs into their respective fields.

LLMs have the ability to comprehend and generate natural human language [6], [7]. This opens up a lot of opportunities for creating software that works with such data. As a result, LLMs can excel in a wide range of environments, such as education [8], customer support [9] and game development [10]. Their versatility makes them a powerful tool for developers looking to build more natural and intelligent user experiences.

2.1.1 Limitations and Mitigation Strategies

Even though LLMs have passed the Turing test, demonstrating that humans often cannot distinguish their responses from human-written text [11], [12], they are not without flaws. LLMs tend to struggle with more complex tasks, which can result in incorrect answers or hallucinations [13], [14]. Hallucinations are when LLMs generate false information that sounds plausible. This is, of course, a significant issue when technology relies on these responses.

However, multiple countermeasures have since been developed to reduce the likelihood of this happening [15]. These countermeasures include both pre- and post-mitigation strategies, all aimed at reducing the likelihood and impact of potential hallucinations.

Pre-mitigation strategies often revolve around the prompting style. Several prompting techniques are commonly used to reduce hallucinations and ensure more consistent output from the LLM. One such technique is few-shot prompting [1], where the prompt includes a few examples of the desired output. This leaves less room for the LLM to guess the format or intent, leading to more accurate responses. Another approach is Chain-of-Thought (CoT) prompting [16], which encourages the LLM to *"think out loud"* by generating its reasoning step by step. This process helps the model reflect on its reasoning and often leads to higher-quality output.

Post-mitigation strategies include ways to process the generated output and filter out hallucinations or incorrect responses. One example is the use of a second LLM that evaluates the initial prompt and the output to ensure it aligns with the intended instructions. This technique was used by Bai et al. [17], where a separate model was employed for self-evaluation and reinforcement learning to improve output reliability.

Hallucinations and incorrect outputs can also often be the result of tasks that are too complex. When requests become overly complicated, it is common to begin modularizing the LLM system. This involves splitting the task into smaller components, each handled by a dedicated LLM module. By doing so, each module can focus on a specific subtask and ignore irrelevant parts of the prompt, which helps reduce confusion and improve the overall reliability of the output [18].

While these strategies have shown promise, they are not without faults. Detecting and eliminating hallucinations remains an open research challenge, especially in complex or high-stakes domains [14], [15].

2.1.2 LLMs in Games

Since OpenAI and other companies released their services through application programming interfaces (API), many game developers have sought to integrate LLMs into games. The use of LLMs in games introduces new design opportunities and possibilities for player interaction. This relatively untapped domain quickly gained popularity among indie developers and researchers, driven by the unique creative potential that emerges from combining LLMs with game mechanics [10].

A natural connection for LLMs in games is their ability to create dynamic dialogue. Developers can leverage LLMs to generate NPC responses and interpret player input. Combining these capabilities enables the simulation of believable, responsive NPCs in video games.

Several tools and games have already been developed to experiment with LLMs. *AI Dungeon*, created by Nick Walton, allows players to generate open-ended text adventures using GPT-2 and later GPT-3, offering a dynamic storytelling experience [19]. *CALYPSO* is a tool designed to assist Dungeon Masters in Dungeons & Dragons by providing descriptive content and encounter ideas [20]. The commercial game *1001 Nights* (Ada Eden, 2024) also leverages LLM-driven dialogue, using AI to support co-creative storytelling between the player and the game [21]. Community mods have also been created for popular games like *Skyrim* (Bethesda Game Studios, 2011), where the *Mantella* mod enables traditional NPCs to use LLMs and respond freely.[1]

LLMs can offer several benefits for games, such as flexibility in content generation [10]. A single LLM system can be applied across multiple NPCs or datasets, enabling diverse and scalable applications. Furthermore, LLMs can personalize playthroughs by adapting to the player's tone and playstyle [22]. Lastly, LLM integration enhances replayability, as content is typically generated at runtime, allowing for unique experiences in each session.

However, relying on LLMs in games also raises concerns regarding predictability and ethical considerations. If a game depends heavily on the LLM's responses for progression, the system must be highly reliable. Hallucinations can confuse players or break immersion, potentially affecting the game's overall reception [14]. In addition, integrating LLMs into real-time systems can result in challenges related to latency and cost [10]. Furthermore, delegating dialogue generation to LLMs may diminish the creative role of game designers and writers, raising questions about authorship and narrative control [10].

These strengths and challenges highlight the growing role of LLMs in games, particularly as developers explore new ways to balance creativity, control, and player experience.

[1] https://www.nexusmods.com/skyrimspecialedition/mods/98631

2.2 Non-Player Characters in Games

NPCs play a vital role in video games. They are defined as characters controlled by the game. Their role and purpose can vary, and they can have almost endless functionalities in a virtual world [23]. Examples of well-known NPCs from video games include Cortana from *Halo* (Bungie, 2001; 343 Industries, 2012–present), GLaDOS from *Portal* (Valve, 2007), and Claptrap from *Borderlands* (Gearbox Software/2K Games, 2009). However, they go all the way back to the enemies from *Space Invaders* (Taito, 1978), *Donkey Kong* (Nintendo, 1981), or the ghosts from the original *Pac-Man* (Namco, 1980).

NPCs can take on different roles, such as enemies, allies, or companions, etc. They can also provide specific functions such as buying and selling, providing services, guarding places, getting killed for loot, giving quests, offering information, performing general services, or even functioning as background characters to make a place seem busier and more alive [23] [24].

2.2.1 Player-NPC Relationship

A crucial NPC role is the ability to provide an active connection to the narrative by being interactable actors for the player [24]. The NPC has a certain level of affordance that can give a visual indicator of what the NPC can provide, how it behaves, and how it acts within its game world [24]. An NPC would therefore be designed in accordance with its intended purpose within the virtual world. However, affordances can be controlled to a certain degree by the design choices made by the developers, though the impression is still dependent on the player's perception of the NPC [24]. This also shapes the relationship between the player and the NPCs and creates believability. This believability has the potential to enhance the immersion of the interaction between the player and the NPCs [25]. The effect of this, and the way NPCs perform their role as part of the narrative, affects how the player experiences the story of the game [25], [26].

In addition to fulfilling different roles, NPCs can be designed based on principles that provide them with a level of intelligence, personality, and agency [27]. These principles enable NPCs to support emergent gameplay in their interactions with the player.

NPCs are therefore vital for games and can be added with many different roles and purposes. How an NPC is implemented should be carefully considered, as both their affordance and believability can affect the relationship between the player and the NPC and thereby also the player's experience.

2.3 Dialogue Systems in Games

Dialogue systems in games are often one of the core pillars that shape the gameplay experience, particularly in role-playing games (RPGs). These systems provide players with essential context, whether through interactions with NPCs or internal monologues. They also enable players to explore different conversational routes, often allowing them to actively shape their own narrative path through dialogue choices. Through dialogue options, players can also often witness the development of the player-character, gaining insight into their evolving personality, motivations, or relationships over time [28]–[30].

Dialogue systems have evolved significantly throughout the history of video games. One of the earliest forms relied on text parsing, where players would input

full sentences and the game would extract key words to determine a response. This approach was used in games like *Zork* (Infocom, 1977). In contrast, modern games have begun integrating LLMs to procedurally generate dialogue that responds dynamically to environmental events and player interactions [31], [32].

The following section will explore the space of traditional and procedural dialogue systems, with the aim of understanding existing approaches and identifying areas for potential improvement.

2.3.1 Traditional Dialogue Systems

Traditional dialogue systems often rely on branching decision trees. When interacting with an NPC, the player is presented with a line of dialogue and a set of pre-written responses. By selecting from these options, the player moves through a decision tree that unfolds into new dialogue paths. This system is typically deterministic, meaning that choosing the same options will always lead to the same outcome [33]. As a result, interactions between the player and NPCs are easy to replicate, allowing players to skip familiar dialogue during replays.

The implementation of dialogue trees can vary in complexity, depending on how much the player's choices influence the rest of the game. In simpler systems, such as *Pokémon Red and Blue* (Game Freak, 1996), dialogue is mostly linear and non-interactive, with NPCs delivering fixed lines and little to no variation based on player input. In contrast, more complex systems like those in *Baldur's Gate 3* (Larian Studios, 2023) allow dialogue choices to shape the narrative in meaningful ways, influencing character relationships, quest outcomes, and even the game's ending [34].

Disregarding the complexity of individual implementations, the traditional static dialogue tree has remained a staple in game design for over two decades. This enduring relevance is unsurprising, as the method offers developers reliable control over tone, pacing, and overall narrative quality [34], [35]. A deterministic structure ensures that the story unfolds exactly as intended. However, this same rigidity can limit the sense of player agency and diminish the illusion of choice, as outcomes often remain fixed regardless of the player's dialogue selections [33].

Modern games such as *Baldur's Gate 3* (Larian Studios, 2023) and *Disco Elysium* (ZA/UM, 2019) have reached a narrative depth that offers players a genuine sense of agency. Their complex dialogue systems, where player responses meaningfully influence the unfolding story, create the impression that choices truly matter. However, achieving this level of narrative interactivity is often unrealistic for most developers aiming to create dialogue-driven games. One of the main challenges of using traditional dialogue systems today is the sheer volume of writing required, along with the need to carefully weave dialogue into the game world so that it reflects player actions and maintains narrative coherence. [33], [36]

To address these challenges, researchers have explored the use of artificial intelligence in interactive narratives. For instance, Riedl and Bulitko discuss how intelligent systems can balance player agency with authorial control, potentially reducing the authorial burden [37].

In summary, large and complex dialogue systems that shape the game's narrative can significantly enhance player immersion [34]. When a game offers a wide range of meaningful choices, it encourages replayability by allowing players to explore different narrative paths [38], [39]. On the other hand, less complex traditional dialogue systems often struggle to provide a unique experience across multiple playthroughs. Players may begin to anticipate NPC responses, which reduces the sense of novelty that narrative-driven games aim to deliver [37]. One possible solution to this issue

is to procedurally generate dialogue, allowing each playthrough to feel fresh and dynamic by introducing variation and unpredictability in NPC interactions [10].

2.3.2 Procedural Dialogue Systems

Procedural dialogue systems, in contrast to traditional dialogue systems, involve interactions and dialogues that are constructed dynamically at or during the game's runtime. For the purposes of this thesis, we define procedural dialogue as dialogue that is not fully pre-authored. As a result, procedurally generated dialogue has the potential to increase variety, enhance responsiveness, and improve player agency by reducing reliance on static dialogue trees [10], [37].

Even though previous researchers have attempted to simulate procedural dialogue systems through various AI frameworks, the space has remained relatively underexplored. However, with recent advancements and increased availability of LLMs, there has been a surge in research focused on integrating this technology into procedural dialogue systems [40], [41]. Thanks to LLMs' ability to generate dynamic responses based on user input, they have been effectively used to create chat-like dialogue systems in games. As a result, both researchers and players have begun developing frameworks and mods that incorporate LLMs into both new and existing game titles [10], [42].

In the player-made mod *Mantella* for *Skyrim* (Bethesda Game Studios, 2011), players can use speech-to-text (STT) and text-to-speech (TTS) technologies to engage in spoken conversations with NPCs throughout the game world. The system processes the player's voice input and runs it through an NPC interface designed to ensure that the response aligns with the character's personality and knowledge. This setup enables a near-natural conversational experience between the player and NPCs [10]. To initiate a dialogue, the player simply approaches an NPC and begins speaking. To end the conversation, the player must signal their intent verbally by saying "Goodbye" or a similar phrase, allowing for a more fluid and immersive interaction.

Researchers have also developed games from the ground up that utilize procedurally generated dialogue. *AI Dungeon* (Latitude, 2019) is an interactive, text-based adventure game that leverages LLMs to generate dynamic narratives in response to player input. The game begins with the player creating a character, after which they input actions or dialogue using natural language. This input is processed by an LLM, which generates a continuation of the ongoing story. The game's responses adapt to the player's choices, enabling an almost infinite number of possible story paths [19].

Another commercially available game utilizing procedurally generated dialogue is *1001 Nights* (Ada Eden, 2024), a 2D text-based narrative game with a strong emphasis on storytelling. In this game, the player assumes the role of Shahrzad, who must co-create a story with the AI-driven King. By guiding the King to mention specific weapons such as "sword" or "shield", the player can materialize these items within the game world. These weapons are then used in a climactic battle against the King. Upon defeating him, players can replay the game to explore alternative narrative paths and combat strategies [21]. This clever use of LLMs to bring stories to life offers a fresh and imaginative way of exploring what the technology can do.

More subtle integrations of LLMs have also been explored in research. In 2023, a team of researchers investigated player perceptions of LLM-generated dialogue in the commercial video game *Disco Elysium* (ZA/UM, 2019) [43]. In their study, an LLM was used to rephrase certain dialogue options while maintaining the original

tone and awareness of surrounding narrative elements. The researchers found that players generally preferred the human-written dialogue over the LLM-generated alternatives.

However, implementing procedurally generated dialogue can also introduce challenges for both players and developers. Increased freedom for players often means reduced control for developers. Allowing players to respond and act however they choose can lead to unexpected story developments and alter the intended gameplay experience. This issue was encountered by researchers in 2024 when they developed a text-based adventure game in which the player had to defuse a bomb threat in a town [44]. Players used text commands to gather clues and prevent the threat in time. If they failed, they would restart the game with the knowledge gained from previous runs. Unexpectedly, some participants chose to side with the antagonist, an outcome that was not anticipated by the developers.

2.4 Gaps in Existing Literature

Current research has seen numerous attempts to implement LLM-driven dialogue systems, resulting in valuable insights and data. Some studies have explored novel and creative implementations that point to new possibilities for future integration. However, most research tends to focus on isolated implementations and does not compare LLM-driven dialogue systems with traditional approaches or examine different strategies for LLM integration [19], [21], [43], [44]. Moreover, it is not feasible to compare dialogue systems in established games with those newly developed in prototype environments. This gap limits our understanding of how various LLM-based dialogue systems actually impact player experience in narrative-driven games.

Furthermore, many studies primarily focus on technical implementation or design frameworks, often at the expense of collecting empirical data on how LLM-driven dialogue systems influence player interaction and behavior. Since games are ultimately designed for human users, it is essential to understand how this emerging technology impacts the experiences of its target audience.

By combining these gaps, it becomes clear that future research must compare different dialogue systems and investigate how these technologies affect players. This study addresses the gap by directly comparing several versions of LLM-driven NPC dialogue against a traditional baseline, focusing on how different integration strategies influence player interaction. This approach responds to recent calls in the literature for more player-centered evaluations of LLMs in games [10], [40], and is guided by the research questions outlined earlier.

Chapter 3

Methodology

3.1 Game Prototype Design

This section will cover the game prototype design. It will focus on the methodological approach, how different elements of the game were designed, and how it relates to exploring the research questions. The prototype follows the principles of an ordinary video game [23] from a test participant's perspective, but it is designed to monitor and test various metrics of data related to the problem domain. It is therefore not only a game, but also a tool for testing.

The game serves as a test environment where all design choices, visual elements, mechanics, sounds, and more, are made with the research goals in mind. These elements are included not just to create an enjoyable experience, but to ensure the game functions as a high fidelity prototype. While the importance of each design choice may vary, all contribute meaningfully to the game's effectiveness as a test environment.

The game was designed with flexibility in mind to allow different dialogue systems to fit within the same game environment. Thus, all dialogue systems are implemented in the same game world without any world changes besides the dialogue interfaces. It was crucial to create a world that would be susceptible for change of dialogue systems, as a comparative study between the different versions would be conducted later. To address this, an analysis was conducted to determine which aspects of the game would be most sensitive to variations introduced by the different dialogue systems. This analysis identified three primary areas of impact:

- The Game World
- The Story and Narrative
- The NPCs and Challenges

A proper design of these three elements was crucial to create a game that could adapt to the different versions while maintaining consistency in player agency. Here, the focus was mainly on creating a flow in how they operate with each other.

First, the world is designed so that all the NPCs and the story fit the visual environment. The general style matches the theme, and all the sounds are consistent with what the player sees. All challenges and interactions remain the same across all versions.

The NPCs and their tasks are also designed so that the information they can provide matches the challenges in all versions. The output of information changes depending on which version is being played, but the content is the same. The personality, location and rhetoric are also the same for each NPC.

Lastly, the story was designed to match the world, the NPCs, and the challenges. The story is the same across versions and is automatically adaptable.

In summary, the prototype served a dual purpose. It functioned both as a traditional game and as a controlled, adaptable test environment. The design choices were implemented to create a sense of flow and cohesion between these two purposes.

3.1.1 Game Overview

The following is a description of the game. It covers the theme, the world, the NPCs, the challenges and the role of the player. The chosen theme for the game is a classical medieval fantasy setting, featuring adventure, quests, magic, and evil creatures. The fantasy genre was chosen as it is a relevant genre that supports a narrative-driven world in which dialogue plays a central role. This enables the implementation of NPCs with distinct characteristics and personalities, allowing for nuanced interactions between them and the player. Some typical RPG mechanics were not implemented. Such mechanics include combat, experience leveling, carried gear, etc. It was decided that an RPG is not defined by these elements, and including them would distract from the research focus and require unnecessary resources. Therefore, the main mechanic of the game is dialogue. Pictures of the game can be seen in Figures 3.1 and 3.2.

The premise of the game is as follows: Goblins, led by a clever Hobgoblin, have stolen all the food from the town. The player must embark on an adventure to retrieve the stolen food. To do so, they must solve various challenges. These challenges are solved through interaction and dialogue. Players can interact with NPCs, some of which have important information related to the different challenges. This information can be interpreted and applied to help the player complete the challenges. The player can also choose to try and complete the challenges without seeking help first.

The game world is divided into distinct areas. These are the town, goblin outpost, forest, mountain road, and the Hobgoblin's lair. Each area serves a purpose. The town is where the player begins their journey. The majority of the NPCs can also be found in the town. The town is composed of houses, some of which serve as locations for the important NPCs. The player starts at the inn, which functions both as the starting point and as the respawn point. Upon failing a challenge, the player is returned to the inn. The player can choose whether to gather information from the town's NPCs or proceed directly into the goblin outpost.

However, the player must speak to the Innkeeper before they can proceed to the forest. From there, each subsequent area presents a challenge. The goblin outpost contains a goblin camp and a road that splits into four paths. Only one of these paths is safe, and the player must determine which is correct.

In the forest, the challenge is to choose the correct color of berry. The purpose is to gain nutrition and choosing the wrong one results in fatigue and fainting, thus failing the challenge. Various types of berries in different colors are found here, but only one color is safe to eat.

Next is the mountain road, where the path is blocked by a magical barrier. The barrier can be removed only if the player says the correct magic words.

Finally, the player must convince the Hobgoblin to return the stolen food. He can be persuaded by participating in his game, which requires the player to speak in rhymes. Upon completing the final challenge, the player is teleported back to the town, where the stolen food has been returned and the NPCs are celebrating. This marks the end of the game.

FIGURE 3.1: Pictures from the game

The top picture shows a bird's-eye view of the town. The bottom picture shows the player character (left) and the Innkeeper (right) inside the town inn.

The Player Character

The player character is intentionally simple. He is a male figure with no detailed background or predetermined character traits. His mission is to help the village recover their stolen food. The primary purpose of the player character is to provide the player with an agent that gives them presence in the game world, enabling them to navigate the world, interact with NPCs, and complete the various challenges.

Story and Narrative

This chapter will elaborate on the design choices behind the story and how it benefits the game as a test environment. It was also a requirement that the story support all versions of dialogue systems. The story follows the premise, but its simplicity and the inclusion of NPCs allow the player to create an emergent narrative [45]. To summarize and elaborate on the story:

A clever Hobgoblin and his band of goblins have stolen the town's food supply. The town is in distress, as they risk starvation unless the food is returned. All the NPCs in the town are aware of what has happened, and they expect the player to help retrieve the stolen food. To recover the food, the player must embark on an adventure through the forest to the mountains, where the Hobgoblin resides in his lair. The journey poses many challenges, which can be solved either by pure luck or applied information.

This story was designed to support the implementation of unique NPCs and to allow players to interact with them, explore the story, and create their own narrative. Here, the narrative is defined as a sequence of incidents or events that share information with the player [46]. In this project, dialogue is used to guide the player

FIGURE 3.2: More pictures from the game

The top picture shows the goblins at their encampment. The bottom picture shows the entrance to the Hobgoblin's lair.

through the story while they engage with an emergent narrative [46]. For example, the player is not given any direct knowledge of the goblins' motive unless they find it through NPC dialogue.

The story was also designed to appeal to a wide range of players. This was done by prioritizing the simplicity of the story, combined with the emergent narrative. This approach was selected to reduce the cognitive load for testers and offer a less complicated playthrough, which benefits the test environment. It was designed with the intent of helping the player understand the story, since a complicated plot could overwhelm the participants, therefore negatively affecting their experience and complicate NPC dialogue.

3.1.2 Key NPCs and Challenges

The NPCs play a central role in this project, acting as the primary interface through which players engage with the various dialogue systems. The world is designed so that the player begins by gathering relevant knowledge through interactions with the NPCs. The game features many NPCs, some of which are designated as key NPCs.

The key NPCs are: the Blacksmith, the Herbalist, the Thief, the Huntsman, the Wizard, the Bard, the Innkeeper, the Magic Barrier, the Goblins, and the Hobgoblin.

These are defined as characters who are designed based on principles that allow them to exhibit intelligence and personality [27]. These principles help create emergent gameplay interactions between the player and the NPCs. Additionally, this

(A) Map for navigating the world

(B) Notebook for remembering vital information

FIGURE 3.3: In-game tools provided by the Innkeeper to assist the player

provides these NPCs with both personality and agency, which enhances their believability and communicates their capabilities more clearly [25]. This gives the NPCs the power to co-create narrative experiences with the player, helping the player better understand the story [26].

Some key NPCs serve a more guiding role. For instance, the Bard introduces the player to the story, while the Innkeeper provides useful tools to assist the player. These tools are a map and notebook. These can be seen in Figure 3.3a and 3.3b.

The personalities of the key NPCs were modeled to make them more dynamic and intelligent. This was done by giving them unique traits containing a set of organized characteristics that influence their interactions [26]. These traits were tailored to match the theme and role of each key NPC, making them distinct characters. The traits not only differentiate the NPCs but also shape the tone of conversation, potentially resulting in sarcastic rhetoric or angry discourse.

Table 3.1 is a simplified overview of the personalities that define the overall characteristics of the different key NPCs.

The Challenges

The player faces different challenges as they progress through the game. They can choose to approach the challenges blindly and attempt to solve them, or they can seek information by asking specific key NPCs. The challenges are completed

Key NPC	Description
The Blacksmith	A hardworking man with deep knowledge of metalwork. Tough, reliable, and respected in the town.
The Herbalist	A gentle and caring expert on nature. Knows which fruits, flowers, and berries are safe or dangerous.
The Thief	Charismatic, cunning, and resourceful. Shrouded in mystery and driven by self-interest.
The Huntsman	A quiet and capable pathfinder who supplies wild game. Deeply familiar with the surrounding forest.
The Wizard	Eccentric but brilliant. Lives alone in a tower and enjoys testing others with riddles before offering help.
The Bard	Overcome with sorrow after the food theft. He channels his emotions through storytelling and music at the inn.
The Innkeeper	Warm, social, and pragmatic. Runs the inn with a welcoming spirit and knows much about the town's people.
The Magic Barrier	It is a static object. A magic-infused gate that waits for either the correct words or a reason to punish intruders.
The Goblins	Sly and opportunistic. Willing to strike a deal, but just as likely to start a fight if provoked.
The Hobgoblin	A brutish figure with a surprising fondness for rhymes. Guards the stolen food deep in his lair.

TABLE 3.1: Descriptions of the key NPCs and their narrative roles.

Challenge	Description of Uniqueness
Find the Innkeeper	Serves as a tutorial to introduce the player to the dialogue system.
Find information from the NPCs in town (optional)	Encourages players to actively seek out knowledge through optional conversations.
Bribe Fizzby the goblin (optional)	The dialogue with Fizzby can turn hostile, leading to the player being knocked unconscious. Success requires careful phrasing to avoid provoking the goblins.
Choose the correct forest path	Requires players to synthesize clues from both the Huntsman and the goblins to identify the correct path forward.
Eat the correct colored berries	The berry challenge demands precise knowledge, as random guessing is unlikely to succeed given the number of options.
Break the Magic Barrier	Players must recall or record a specific phrase to deactivate the barrier, testing memory or note-taking.
Convince the Hobgoblin to return the food	This challenge demands creativity, as the Hobgoblin only responds to rhyming dialogue.

TABLE 3.2: Overview of game challenges and their unique dialogue-based mechanics.

through simple interactions or dialogue. The challenges are designed with a degree of variation to ensure the player remains engaged and the gameplay does not become trivial. To achieve this, some challenges include subtle, implied characteristics that influence how they can be approached. The goal is to allow a more nuanced level of engagement by varying how players process and apply the information they receive. Table 3.2 provides an overview of the unique characteristics of each challenge in chronological order:

Each challenge is linked to key NPCs who provide the necessary information to

Key NPC	Information	Challenge
The Bard	Initiates the story and guides the player to the innkeeper	Find the innkeeper
The Innkeeper	Knows everyone in town and provides a map and a notebook	Find Information from the NPCs in the town
The Blacksmith	Knows that goblins love silver	Bribe Fizzby to acquire knowledge about the forest
The Herbalist	Knows that only the red berries in the forest are edible	Eat the correct color berries
The Thief	Knows that hobgoblins love rhymes	Convince the Hobgoblin to return the food
The Huntsman	Knows key information about the forest	Choose the correct forest path
The Wizard	Knows the magic word to disable barriers	Breaking the Magic Barrier
The Goblins	Know key information about the forest	Choose the correct forest path

TABLE 3.3: Overview of key NPCs, their knowledge, and associated challenges

complete them. These NPCs are designed to align thematically with their respective challenges. They are also prompted to deliver their guidance in a tone that matches their character traits. Regardless of which dialogue system variant the player is experiencing, the challenges remain the same, as do the information and personalities of the key NPCs. Table 3.3 displays an overview of the key NPCs, their functions, and how their information relates to their respective challenges.

3.1.3 Core Gameplay Loop

The core gameplay loop remains the same across all dialogue system variants, as the challenges themselves do not change. At the start of the game, the player chooses whether to interact with NPCs. They then attempt a challenge, which they can either succeed at or fail. If they fail, they are returned to the inn, where they can choose to immediately retry the challenge or spend time interacting with NPCs. Successfully completing a challenge allows the player to progress further in the game. The player can also fail a challenge even after having interacted with the correct NPCs if the information they have gained is not applied correctly. Figure 3.4 outlines the chronological progression of the challenges and the information associated with each one.

Upon failing a challenge, players are encouraged to seek new information or clarify what they have already learned by interacting with NPCs. To reduce time spent running back after failing a challenge, a fast travel system allows them to quickly return to a milestone near the challenge they reached.

This is the gameplay loop until the player has completed all the challenges and thereby also the game.

3.1.4 Tools for Creating the Prototype

The game is a high-fidelity prototype developed in Unity [1]. All assets used for both the NPCs and the game world's environment are pre-made assets from the Unity

[1] https://unity.com/

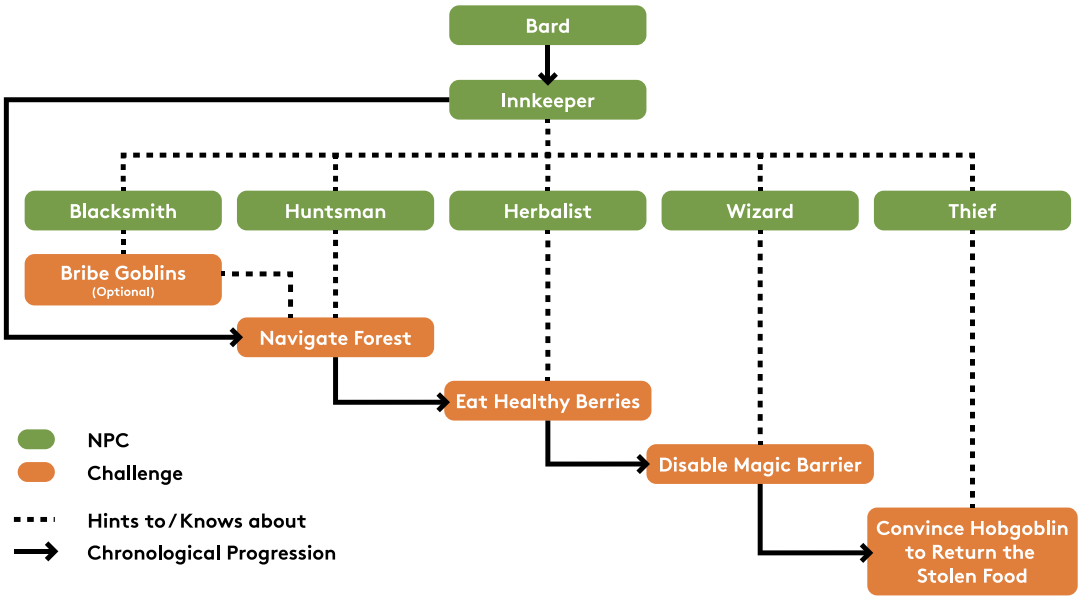


FIGURE 3.4: Challenge-Dependency Chart

The player begins with the Bard and is directed to the Innkeeper, who recommends speaking with five key NPCs. Each NPC provides information for solving the challenges.

Asset Store [2]. Premade assets were used as designing and creating everything from scratch would have required a significantly large amount of time. The assets come from multiple packages [3][4][5] that include a variety of characters and environments. All characters share a consistent visual style and align with the fantasy theme. The music and sound is royalty free and sampled from Pixabay [6].

3.1.5 Usability Tests

In this study, usability testing was conducted three times before finalizing the prototype. Usability testing was essential during the development phase of this technical prototype, especially given the use of an iterative design approach [47]. It helped identify technical issues, design flaws, and other limitations early in the process. Repeating tests across different iterations allowed developers to identify, track and verify if issues were recurring over time. Ensuring the overall quality of the game before conducting user studies was crucial. Exposing test participants to a low-quality version could bias their opinions about the key mechanics being evaluated [48]. Therefore, the game had to reach a certain standard of quality before the primary testing was conducted.

Three participants with general gaming experience were selected. All usability tests were conducted with the same participants. They were asked to play the game naturally while providing feedback during the testing. Their feedback helped identify critical errors, suboptimal mechanics, and UI issues that might negatively affect the user experience. Using the same participants in all tests allowed for a more consistent evaluation of changes made between iterations. This process aligns with commonly used practices in iterative usability testing [47].

[2] https://assetstore.unity.com/
[3] https://tinyurl.com/5t3jejzp
[4] https://tinyurl.com/4mmjhsdm
[5] https://tinyurl.com/chcfusk2
[6] https://tinyurl.com/3w6ec8jx

3.2 Dialogue System Variants

The core difference between the four versions of the game lies in the design of the dialogue systems. By altering how the dialogue system generates, rephrases, and varies its content, and by examining how players interact with the system in each version, a thorough investigation can be conducted. This investigation focuses on how players communicate, obtain information, and progress through the different game versions. These four versions range from fully pre-written dialogue with no

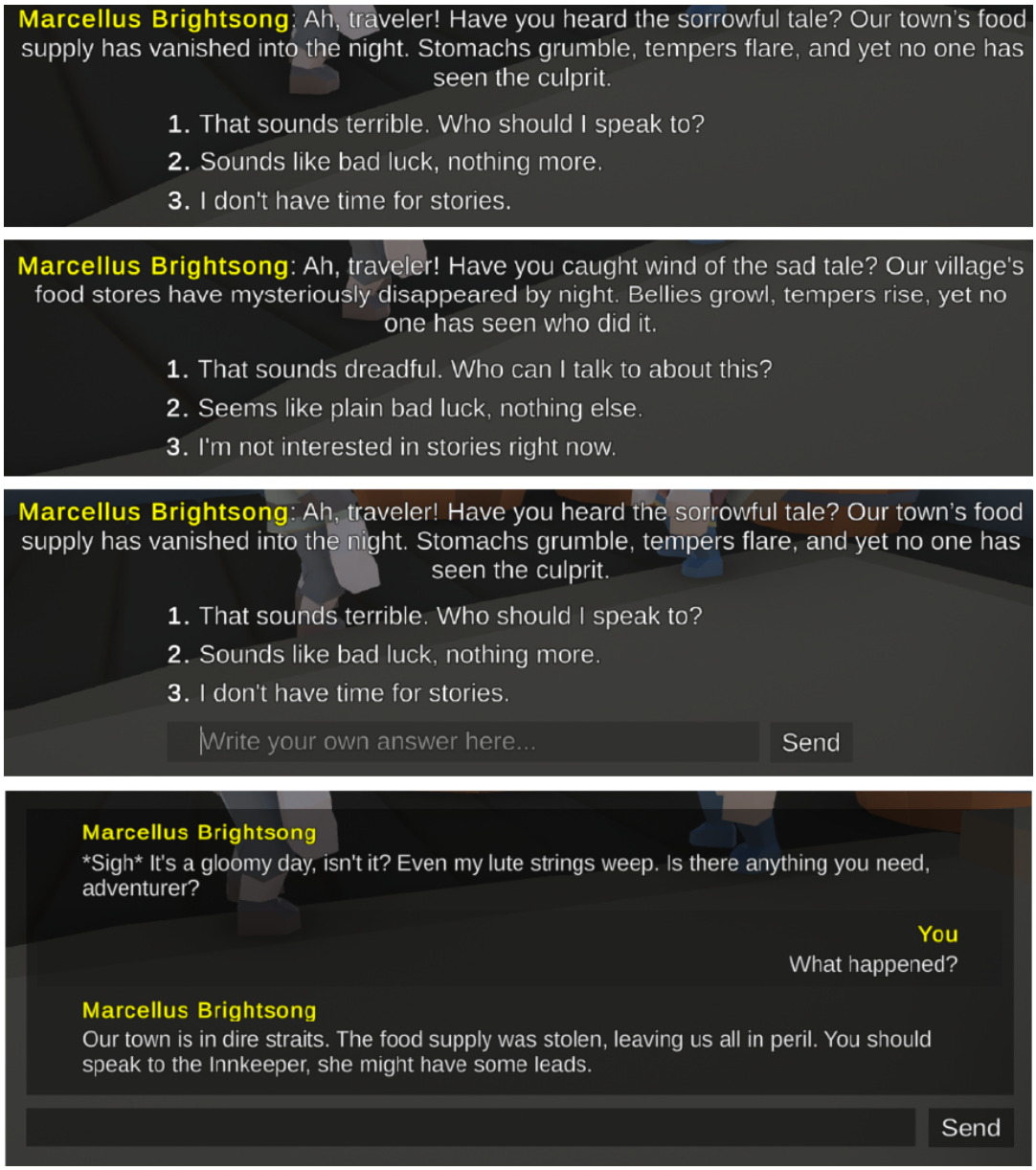


FIGURE 3.5: Comparison of dialogue system variants in the prototype. From top to bottom: *Control Version*, *Version A*, *Version B*, and *Version C*.

The *Control Version* uses fixed dialogue options. *Version A* introduces minor paraphrasing via LLM. *Version B* allows open-ended input that branches into new responses. *Version C* supports fully open-ended dialogue using LLMs for dynamic interaction.

LLM integration to an open-ended system entirely powered by an LLM. The versions are named *Control*, *A*, *B* and *C*, see Figure 3.5. The following section introduces the four different dialogue system variants implemented in the game. The specific technical implementation of LLM integration is described in detail in the section LLM Integration.

This project specifically uses OpenAI's API to **ChatGPT** as the underlying language model [49]. Accordingly, all references to "LLM" throughout this study refer to OpenAI's GPT-4o model.

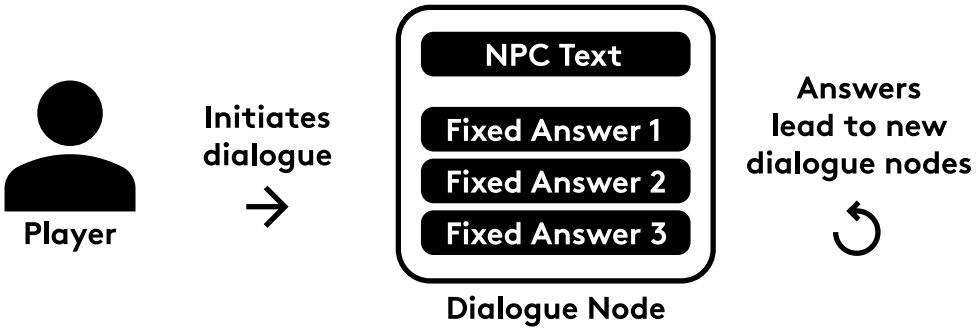


FIGURE 3.6: Control Version

Illustrates a traditional NPC interaction model. The player initiates a dialogue, triggering a dialogue node containing NPC text and a set of fixed player answers. The player chooses an answer, leading to a new dialogue node.

3.2.1 Control Version: Fixed Dialogue Choices

The *Control Version* represents a traditional dialogue system, commonly found in narrative-driven games. In this version, all NPC dialogue is fully scripted and prewritten, with no generative elements. This means that both the dialogue and the branching conversation paths are manually authored in advance, and no LLM is utilized at any point in this version.

This dialogue system is implemented using a traditional branching architecture, wherein dialogue nodes are organized in a hierarchical, tree-like structure that governs the flow of interaction. Each node contains a message from the NPC to the player, followed by a set of possible dialogue options for the player to choose from. Each dialogue option is linked to a specific node in the dialogue tree, and selecting an option advances the conversation to the corresponding node. The system is deterministic, meaning that given the same conversational input, the player will always experience the same conversational output. Illustrations of both the system and the overall dialogue tree structure are shown in Figures 3.6 and 3.7, respectively.

The dialogue trees are designed to guide the player through the necessary steps required to complete the challenges and progress through the game. This version offers clarity and predictability, as the player is explicitly presented with all available dialogue options. It serves as the baseline or control condition against which the other LLM-based versions are compared.

3.2.2 Version A: Dynamic Dialogue Rephrasing

Version A introduces an LLM integration aimed at increasing the variability of conversations with NPCs without altering the underlying structure or context of the dialogue system. It builds directly upon the *Control Version* by incorporating an LLM

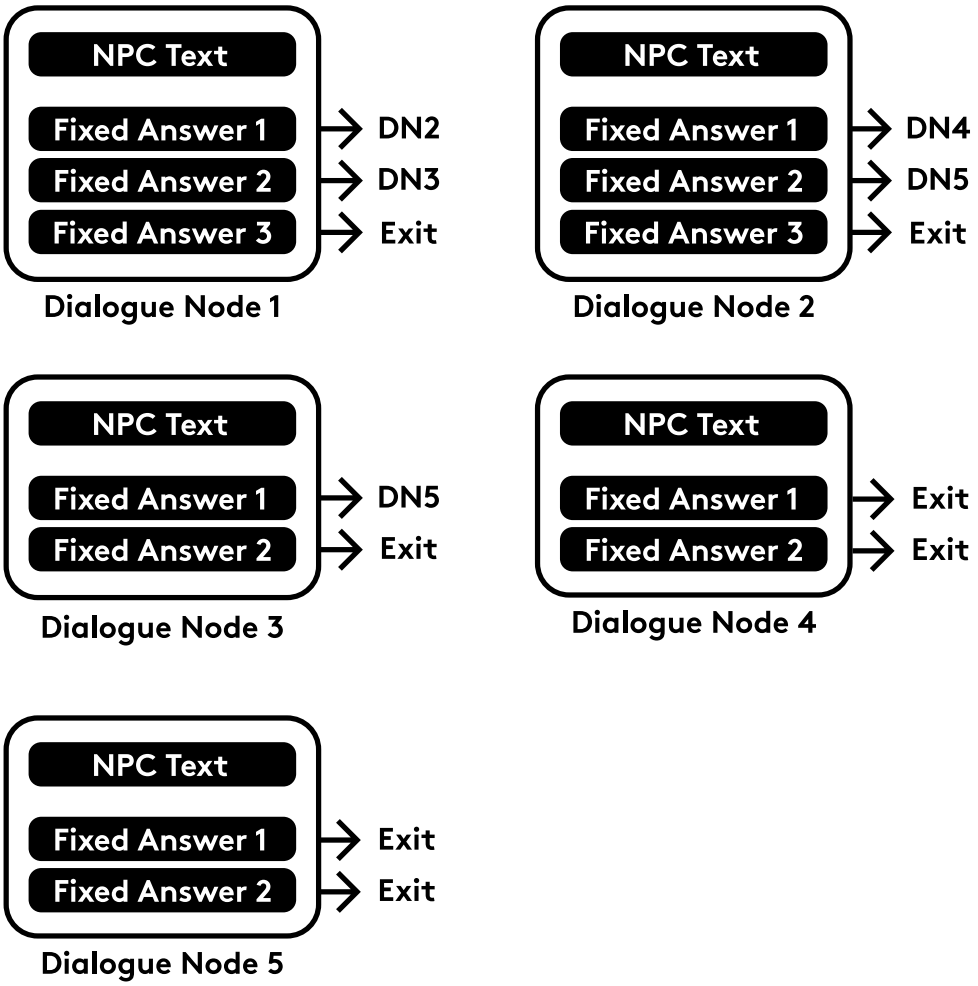


FIGURE 3.7: Dialogue Tree Structure

Each dialogue node consists of one NPC text and multiple fixed player answers. Each answer links to a new dialogue node in the tree. Selecting an answer continues the conversation with the connected node, while some answers may end the interaction.

that dynamically rephrases pre-written dialogue lines. The core structure and flow of conversations with NPCs remain identical to the *Control Version*, meaning that dialogue branches and narrative progression are still predefined.

The key difference occurs whenever a conversation with an NPC is initiated. Before the dialogue is displayed to the player, an LLM rephrases each line of dialogue, including the player's dialogue choices, at runtime. Each line is sent to the LLM with a prompt instructing it to rephrase the sentences while preserving their original meaning, tone, information, and intent. This approach introduces linguistic variation and a sense of spontaneity into an otherwise rigid and static conversation, aiming to make repeated interactions feel less mechanical and more natural.

This version still operates within the fixed, tree-like dialogue structure of the *Control Version*, meaning that the branching logic and information delivery remain unchanged. As a result, the player receives the same general content and dialogue options, regardless of how much the lines are rephrased. However, because the dialogue is dynamically regenerated each time a conversation with an NPC is initiated, no two interactions with the same NPC are exactly identical in terms of phrasing. While the information delivered to the player remains deterministic, the specific wording of each line is unpredictable.

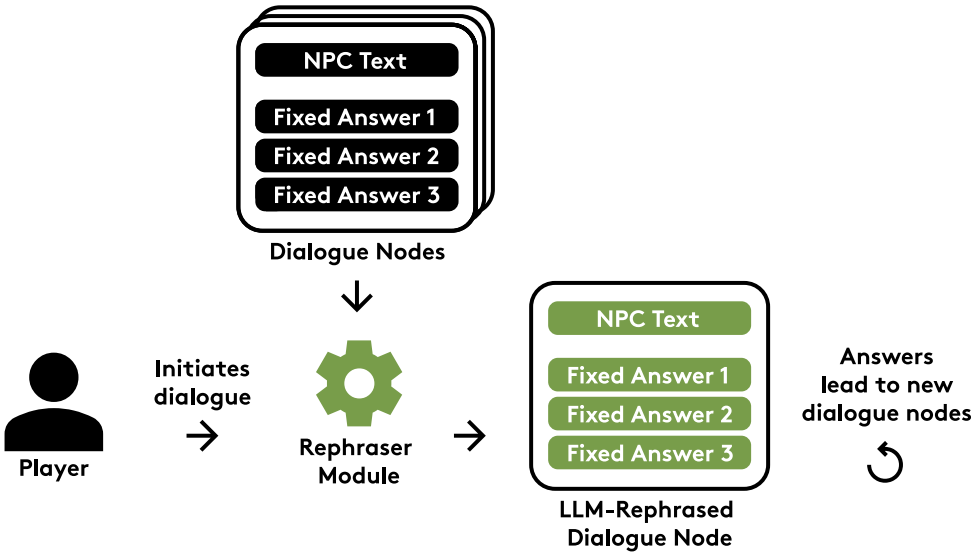


FIGURE 3.8: Version A

Illustrates a traditional NPC interaction model, with an added dialogue rephraser module. The rephraser module sends the dialogue nodes to an LLM, rephrasing the dialogue nodes which are then shown to the player.

This subtle variation helps reduce the repetitiveness often associated with fixed dialogue systems in video games. Thus, *Version A* explores whether surface-level linguistic variation, within an otherwise static dialogue structure, can make interactions feel more organic and natural. An illustration of the system is shown in Figure 3.8.

3.2.3 Version B: Fixed + Open-Ended Dialogue

Version B retains the fixed dialogue structure from the *Control Version* but introduces a new layer of interactivity by allowing the player to write custom responses to NPCs.

As in the *Control Version*, the dialogue system is based on a fixed, tree-like structure in which each NPC follows predefined and deterministic conversation paths. However, players are now given the option to input free-form text instead of selecting from predefined dialogue choices. If the player submits a custom message, the

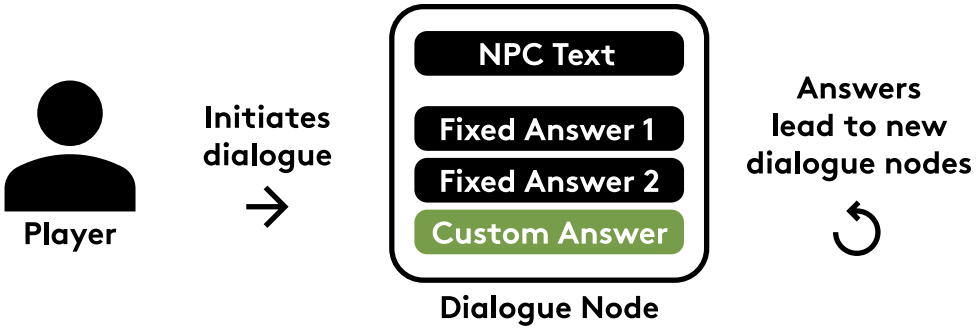


FIGURE 3.9: Version B

Illusrates a dialogue system where the player can either choose a fixed response or type an open-ended input. The open input is sent to an LLM, which generates a new dialogue node that gets inserted into the existing dialogue tree.

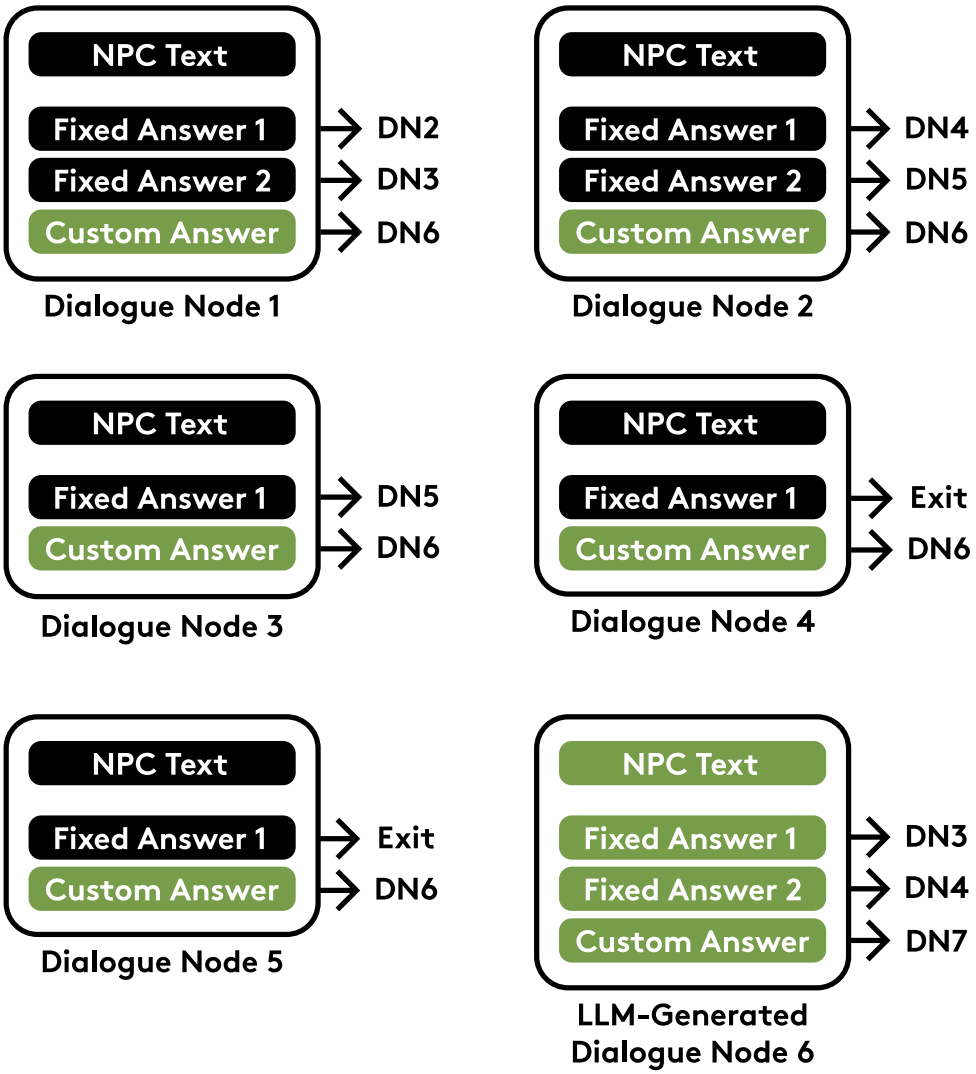


FIGURE 3.10: Dialogue Tree - *Version B*

Depicts how new LLM-generated dialogue nodes are added to the tree structure, preserving the conversation flow while dynamically expanding possible interactions.

system sends the input to an LLM, which is prompted with contextual instructions. These prompts define key characteristics of the specific NPC, such as their role, tone, key information, and narrative constraints. Additionally, the entire dialogue tree is sent to the LLM to provide full conversational context. This ensures that the LLM remains character-consistent and contextually aware. Based on this input, the LLM is then tasked with the following:

1. Generate a direct response to the player's input, taking into account the current dialogue node within the context of the entire dialogue tree.
2. Create a set of dialogue options that relate to the player's custom input and logically align with existing dialogue nodes in the dialogue tree. These new options are then presented to the player alongside the NPC's response, allowing them to either continue with the traditional dialogue path or pursue the open-ended conversational branch.

This version preserves the overall narrative by keeping core information closely tied to the original dialogue tree, while also enabling a more dynamic interaction

model in which players can guide conversations in personal and creative ways. From a technical standpoint, this system introduces additional parsing requirements to handle player input effectively. The dynamically generated dialogue nodes remain constrained within the context of the overarching dialogue tree, thereby preventing the conversation from derailing or producing irrelevant outputs.

This hybrid system allows players to engage with the default dialogue structure while also offering the freedom to express themselves in their own words. An illustration of the system is shown in Figure 3.9. Additionally, an illustration of how the dynamic dialogue tree expansion for this version works is depicted in Figure 3.10.

3.2.4 Version C: Fully Open-Ended Dialogue

Version C represents the most open-ended dialogue system, in which NPC interaction is fully generated through the use of an LLM, with no predefined dialogue options available. This complete departure from traditional fixed dialogue systems results in all NPC dialogue being generated in real time by the LLM, with no scripted responses. Instead, players are encouraged to engage in natural, open-ended conversation by typing freely into an input field. This design most closely mimics the immersive and engaging dialogue experience found in real human interactions [50].

As in *Version B*, each NPC is prompted with unique defining characteristics. These prompts ensure that key information is communicated effectively while keeping the NPCs consistent with their intended character. The player has complete freedom in what they choose to say, enabling a highly personalized interactive experience. Additionally, the absence of predefined dialogue choices incentivizes the player to think more critically about what to ask and how to phrase their input, as the game no longer provides explicit guidance.

This version is the most non-deterministic of the four, meaning that an NPC's output cannot be predicted based on a given player input. No multiple-choice dialogue boxes are present. Instead, players must manually type their questions, requests, or comments. The LLM is responsible for interpreting the player's input and generating an appropriate reply that conveys relevant information while maintaining narrative consistency and preserving the character's identity. This version offers the highest degree of conversational flexibility, enabling a more natural dialogue flow and increasing player agency. To maintain coherence throughout the conversation, some memory is allocated to the LLM to retain information from earlier parts of the conversation [51]. An illustration of the system is shown in Figure 3.11.

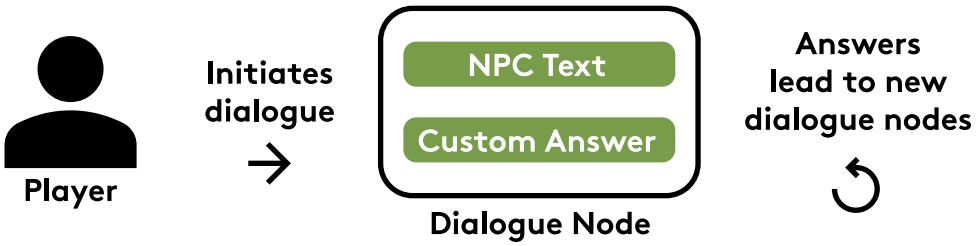


FIGURE 3.11: Version C

Illustrates the dialogue system where the player writes their own open-ended input. When the player begins the dialogue, the LLM sends an initial response. The player can then write a custom message, which is subsequently sent to the LLM. The LLM then formulates a response based on the context of the request, which is finally sent to the player.

3.3 LLM Integration

This section presents the integration of LLMs within the game's system architecture, with a focus on how prompts are constructed and how they are processed across modular components. Given the complexity of integrating LLMs into a real-time game environment, significant effort has been devoted to designing both the prompting strategies and the underlying technical framework to ensure scalability, reliability, and performance.

3.3.1 Prompt Engineering

LLM behavior is largely shaped by how prompts are written and contextualized. Well-engineered prompts serve as reusable solutions to common LLM generation challenges in order to improve accuracy, contextual relevance, and interaction quality. This include strategies that dynamically adjusts the prompt based on contextual aspects, enabling LLMs to adapt outputs to evolving game states, which is crucial for maintaining narrative coherence in interactive environments. [52].

Prompting Strategy

The primary prompting strategies adopted in this project is known as **few-shot prompting** and **zero-shot prompting** [1], [53].

Few-shot prompting is a technique that significantly enhances the reliability and contextual appropriateness of LLM responses. It involves embedding a small number of carefully selected example interactions directly into the prompt. These examples consist of input-output pairs that demonstrate the expected structure, tone, and semantic content of a successful exchange.

By incorporating this few-shot prompting technique, the model is given a clear behavioral pattern to follow, reducing ambiguity and improving coherence in its generated responses. This is important in interactive game settings where consistency and reliability are imperative in order to maintain immersion and believability. Furthermore, this approach is especially useful for when the semantic structure of the LLM response must be precise, as this strategy enables the LLM to know how the exact output format should look like. Some exact implementations of the few-shot prompting technique for this project can be seen in Appendix C.1.1 and C.1.2.

In comparison, zero-shot prompting is a strategy where the LLM receives no prior examples and is expected to infer the intended response style solely based on the instructions alone. This prompting strategy is generally easier to implement and cheaper in terms of token cost. However, it reduces the model's ability to generate highly accurate or contextually nuanced responses, as it lacks concrete examples to anchor its reasoning. An exact implementation of the zero-shot prompting technique for this project can be seen in Appendix C.1.3.

NPC Prompting

Effective prompt engineering also plays a crucial role in ensuring that the NPCs remain contextually relevant, coherent and aligning with their intended personality and narrative function. In this project, NPC prompting refers to the process of dynamically constructing and customizing the input prompts that are sent to the LLM instances responsible for generating dialogue responses for the NPCs.

An important aspect of prompt-based NPCs in RPGs is to collaborate with the player in order to complete multi-step quests and goals [54]. By carefully constructing prompts that include factors such as NPC backstory, goals and contextual data, the LLM can more effectively take on functional roles within the collaborative gameplay, thus highlighting the importance of context-aware prompting strategies for both coherence and gameplay utility.

Similarly, it is essential for the NPCs to be contextually grounded when it comes to their dialogue in order to make interactions with the player feel more immersive, coherent and emotionally satisfying [50]. Some of the guidelines highlighted include giving the NPC contextual understanding of the game world, dynamically adapting to the player's actions and choices, and balancing how the NPC should adapt to the flow of the conversation.

However, recent findings have identified challenges with LLM-driven NPCs in multi-agent simulations [55]. This includes inconsistencies in behavior, hallucinated information and incoherent dialogue outcomes, causing the NPCs to deviate from their intended roles and undermining narrative immersion. Therefore, robust and context-sensitive prompting strategies, such as few-shot prompting, are essential to account for coherent and extended interactions in this study.

Prompt Generation Process

To facilitate a flexible and scalable prompt creation process for the NPC prompts, this project adopts a modular prompt construction pipeline [56]. This pipeline encompasses three core schemas:

- **Feature Characterization Schema:** Classifies the NPC based on personality traits and game context.
- **Automatic Prompt Creation Schema:** Generates a tailored prompt for an LLM using the defined character features and narrative parameters.
- **Dialogue Generation Schema:** Uses the LLM to generate NPC dialogue based on the constructed prompt, ensuring contextual relevance and narrative consistency.

Rather than scripting static prompts for each individual NPC, prompts are dynamically built at runtime by concatenating smaller, character-specific prompt segments. The characteristics used to define each NPC in this system is based on Warpefelt's framework on NPC typology [25]. An NPC is defined by the specific function it has to fulfill and is thereby configured through a set of attributes: *World Description*, *Role*, *Personality*, *State* and *Goal*. These parameters are used to dynamically create each unique NPC prompt, with the exception of the *World Description*, as all NPCs share the same world. Every other parameter is uniquely defined through the NPC's attribute variables. The specific attributes used for each NPC in this project are outlined in Table 3.4.

This dynamic tailoring of character attributes ensures consistency in prompt generation, thus improving NPC behavior. It emphasizes the benefits that context-aware, LLM-driven dialogue systems can have in an in-game environment. The exact structure and syntax for the dynamic generating of NPC prompts can be seen in Appendix C.2.

Variable Name	Description
`Name`	The name of the NPC. Used to reference the NPC in game and throughout the system.
`Gender`	The gender identity of the NPC. Influences how the LLM references them in game (e.g. usage of he/she etc.).
`Personality`	The personality traits of the NPC, such as loyal, noble, or spontaneous. Used to shape tone and dialogue style.
`Goal`	The personal goal or life ambition of the NPC. Helps define motivation and long-term behavior.
`Occupation`	The NPC's job or occupation in the game world (e.g., blacksmith, guard, merchant). Provides context for knowledge and dialogue relevance. Functions as the state of the NPC.
`Key Knowledge`	Specific information or lore that the NPC possesses. Determines what unique insights or facts the NPC can reveal during conversations. Often quest-specific.
`Role`	The narrative function of the NPC in the game (e.g., quest-giver, companion, opponent). Helps shape the structure and focus of interactions.
`Response Style`	The expected style of response (e.g., short, detailed, cryptic). Informs how verbose or direct the NPC should be when replying.

TABLE 3.4: NPC character features used for dynamic prompt generation

3.3.2 Modular LLM Architecture

Developing a video game that integrates LLMs presents several inherent challenges, including unpredictability, potential hallucinations, scalability limitations, and computational inefficiencies [57]–[60]. Relying on a single LLM to manage all required tasks can result in suboptimal performance and ineffective outcomes. To mitigate these issues, a multi-modular architecture has been adopted [61], in which multiple specialized LLMs operate concurrently, each tailored to a specific role or function within the system. This modular approach is consistent with the theoretical framework of multi-agent systems [62], where individual agents are responsible for distinct tasks, thereby promoting clearer task boundaries and improved overall efficiency.

Assigning individual module instances enhances scalability, individualization, and flexibility in the system's technical design. This separation of concerns also helps reduce prompt size, thereby optimizing LLM performance in terms of response speed, accuracy, and token efficiency.

The integration of LLMs varies between game versions *A*, *B* and *C*, with each version featuring uniquely customized architectures and prompting techniques. To avoid developing three entirely separate systems from scratch, the modular architecture allows for efficient reuse and integration of individual components across versions, thereby streamlining development and minimizing redundancy.

Each module is designed with its own specific functionality, rules, and prompts, ensuring that it can perform its designated task efficiently without unnecessary workload. Additionally, each module operates through a separate instance of GPT-4o accessed via individual API calls, meaning that the system does not rely on a shared LLM state. Furthermore, each instantiated module is maintained as an isolated conversational session, ensuring that dialogue histories and contextual understanding remain distinct and unaffected by other modules or characters.

The modules developed for this project are called *NPC Module*, *Function Call Module*, *Quest Completion Module* and *Rephrase Module*. The following subsections provide a detailed description of each module.

NPC Module

The responsibility of the *NPC Module* is to generate contextual and natural responses from NPCs based on player input. Its primary function is to facilitate the conversation between the player and the relevant NPCs while maintaining memory of all prior messages to ensure coherence and consistency throughout the interaction.

This module is specifically employed in *Version B* and *Version C* of the game. Each NPC is assigned a dedicated instance of the module, fully customized to reflect that NPC's unique characteristics and narrative role. The step-by-step process of the *NPC Module* is as follows:

1. **Conversation Initiation:** When the player initiates a conversation with an NPC, a start prompt is sent to the LLM instance of the module. This prompt signals the beginning of a dialogue session and includes initial instructions on how the NPC should respond.

2. **Initial Response Generation:** The LLM instance processes the start prompt and returns a generated opening message, which is then presented to the player through the game's user interface.

3. **Ongoing Interaction:** The player can respond freely to the NPC. Each subsequent message sent to the LLM instance generates a contextually appropriate reply based on the user's input.

4. **Memory Management:** the player inputs and NPC responses are stored within the NPC's module instance, enabling the system to maintain a coherent and contextually aware conversation history.

5. **Conversation Trimming:** To manage the LLM's token limitations and reduce the likelihood of hallucinations, only the most recent dialogue entries are retained. This ensures the conversation remains consistent while avoiding token overflow.

This setup enhances the game's capacity for dynamic natural dialogue between an NPC and the player while maintaining technical efficiency and narrative consistency.

Function Call Module

A traditional function call is typically defined as a reusable block of code that instructs a program to execute a specific function or method. However, in the context of LLM-based applications, *function calling* refers to the process by which an LLM can invoke a predefined operation in response to a player's input [63]–[65]. This capability enables the LLM to interact with external tools beyond its native language generation scope, offering a flexible and scalable approach for enhancing interactivity within the game.

In this system, each function call is manually created using a factory method that instantiates a `FunctionCall` object. The structure and parameters of this object are outlined in Table 3.5.

Parameter Name	Type	Description
`npc`	`NPC`	The NPC associated with the function, linking the function call to a specific NPC in the game world.
`functionAction`	`Func<string, string>`	A delegate representing the actual function logic, taking a string input and returning a string output.
`functionName`	`string`	Defining the name of the function.
`trigger`	`string`	A keyword or phrase that is used to detect when this function should be triggered by the LLM based on the player's input.
`argument`	`string`	An optional argument that defines what should be parsed into the function call as an input.

TABLE 3.5: Parameters of a FunctionCall

The *Function Call Module* is responsible for interpreting player input to determine whether a specific in-game function should be triggered, such as revealing critical information or ending a conversation prematurely. This module is utilized in *Version B* and *Version C* of the game. Its primary purpose is to ensure that certain player utterances can lead to actionable outcomes beyond the bounds of standard dialogue. Each NPC requiring function-calling capabilities is equipped with a dedicated instance of this module, which operates in conjunction with the *NPC Module*. The range of possible function calls is predefined and stored within the NPC's individual *Function Call Library*. Simply put, the *Function Call Library* is the predefined collection of callable actions that an NPC can trigger in response to specific player inputs, tailored to the NPC's narrative role and gameplay functionality.

Each `FunctionCall` object is stored globally within the system. However, to ensure that NPCs only access appropriate and contextually relevant functions, each NPC is assigned a tailored subset of function calls within their dedicated *Function Call Library*. This design ensures that NPCs are restricted to triggering only those functions that align with their narrative role and gameplay context. It also prevents unintended behavior or access to operations beyond their intended scope. The overall step-by-step process of the *Function Call Module* is as follows:

1. **Player Input:** When the player submits a free-form message to the *NPC Module*, the input is first intercepted by the *Function Call Module*.

2. **Function Call Library:** The *Function Call Module* sends a prompt to the LLM instance, asking whether the input corresponds to any known function defined in the NPC's *function call library*. The LLM instance interprets the player's message and evaluates its content against the triggers defined for each available function.

3. **Function Call Construction:** If a valid function call is identified, the LLM instance responds with a structured JSON message specifying which function to invoke and any associated parameters. If no function is recognized, the LLM instance returns an empty JSON object instead, indicating that no function should be executed.

4. **Execution:** The system deserializes the JSON response (if any) and automatically invokes the corresponding function within the game system.

This approach enables the game to dynamically execute game logic through natural language input, without overburdening the NPC's prompt with unnecessary details. In other words, the NPC only receives specific information if the *Function Call Module* determines it to be contextually relevant. A visual representation of the function call process is shown in Figure 3.12.

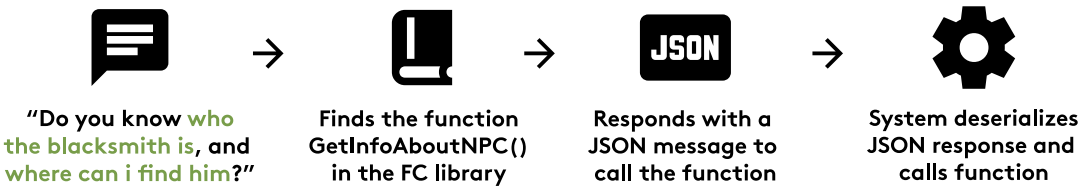


FIGURE 3.12: Function Call Process

Illustrates the process of the Function Call Module. The module looks for keywords or phrases from the player input, and tries to match it with the function triggers in the NPC's function call library. If a match is found, the LLM responds with a JSON message that is deserialized by the module in order to call the function in the game.

Quest Completion Module

The *Quest Completion Module* is responsible for determining whether a player's input should trigger a change in the game's quest state, such as marking a quest (i.e. a challenge) as completed or failed. This module is employed in *Version B* and *Version C* of the game. Similar to the *Function Call Module*, this module intercepts player messages to assess their relevance to active quest objectives and progression logic.

However, unlike the *Function Call Module*, this system does not rely on function calling to update the quest state. Instead, the LLM instance interacts directly with the game's internal quest manager to adjust relevant challenge-related game states. Each NPC is associated with a set of *Quest Completion Criteria*, which define the specific conditions under which a quest may be marked as completed. Similarly, a *Quest Failure Criteria* outlines the conditions that would result in quest failure. If an NPC does not have either of these criteria defined, the *Quest Completion Module* is not invoked for that interaction, thereby minimizing unnecessary LLM usage and improving overall system efficiency. The step-by-step process of the *Quest Completion Module* is as follows:

1. **Player Input:** When the player submits a free-form message to the *NPC Module*, it is intercepted by the *Quest Completion Module*, but only if the interacting NPC has predefined quest criteria.

2. **Quest Check:** The *Quest Completion Module* verifies whether the active quest is associated with the current NPC. If so, it sends the player's input, along with the relevant completion and/or failure criteria, to the LLM instance in order for it to evaluate whether the input satisfies any of the specified conditions.

3. **Decision Response:** The LLM instance returns a response to the *Quest Completion Module* based on its interpretation of the input. The possible responses that the LLM can return are `"Completed"` (if the quest should be completed), `"Failed"` (if the quest should be failed) or `''None''` (if input does not fulfill any criteria).

4. **Handle Quest State:** The system deserializes the LLM response and forwards it to the quest manager, which then updates the state of the active quest accordingly.

This approach enables the game to automatically interpret and update its internal state based on custom, free-form player input. A visual representation of this quest completion process is shown in Figure 3.13.

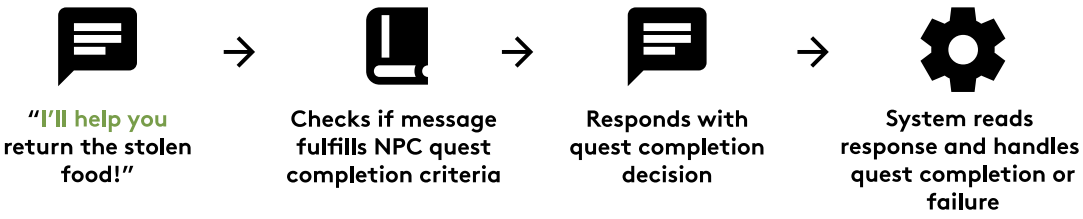


FIGURE 3.13: Quest Completion Process

Illustrates the process of the Quest Completion Module. The module looks for keywords or phrases from the player input that matches one of the quest criteria. If a match is found, the LLM responds with a quest completion response (i.e. "Completed", "Failed" or "None"). This response is subsequently handled by the module to change the quest state.

Rephraser Module

The *Rephraser Module* is exclusively implemented in *Version A* of the game and is responsible for modifying the surface structure of prewritten dialogue. This module employs an LLM instance to rephrase both NPC dialogue lines and available player responses, with the goal of enhancing linguistic variation and improving conversational realism.

Importantly, the rephrasing process does not alter the underlying context, informational content, or intent of the dialogue. Its sole purpose is to introduce surface-level variation across playthroughs, thereby reducing repetitiveness and making each interaction feel more dynamic, even when the functional outcomes remain unchanged. The step-by-step process of the *Rephraser Module* is as follows:

1. **Conversation Initiation:** When the player initiates a dialogue with an NPC, all relevant dialogue nodes and their corresponding player options are sent to the *Rephraser Module*.

2. **LLM Rephrasing:** The module forwards the dialogue data to a specialized LLM, which is instructed to modify the phrasing and introduce linguistic variability while strictly preserving the original meaning, key information, character tone, and narrative intent.

3. **Output Parsing:** The LLM returns the rephrased dialogue as a structured JSON object containing all necessary dialogue elements. This output is then deserialized within the module into a usable NPC dialogue format.

4. **Presentation to Player:** The rephrased dialogue is presented to the player in place of the original, prewritten dialogue.

This module leverages the generative capabilities of LLMs to enhance narrative variability and delivery. It functions as a lightweight augmentation to the game's fixed dialogue structure, enabling LLM-driven variation without compromising authorial control or the integrity of the overall narrative design.

3.3.3 System Overview and Interaction Flow

The modular architecture utilized across the various LLM modules is designed to facilitate reliable, responsive, and scalable dialogue interactions between the player and NPCs. This is achieved by distributing specific responsibilities across distinct modules, each powered by its own dedicated LLM instance. Figure 3.14 illustrates the overarching prompting structure employed across the three core modules.

Both the *Function Call Module* and the *Quest Completion Module* operate using their own isolated prompt schemas. The *NPC Module*, by contrast, integrates both a character profile prompt and a dialogue system prompt, the former defines the character specifications of the NPC, while the latter provides the necessary context for the LLM instance to generate contextually appropriate and narratively coherent responses.

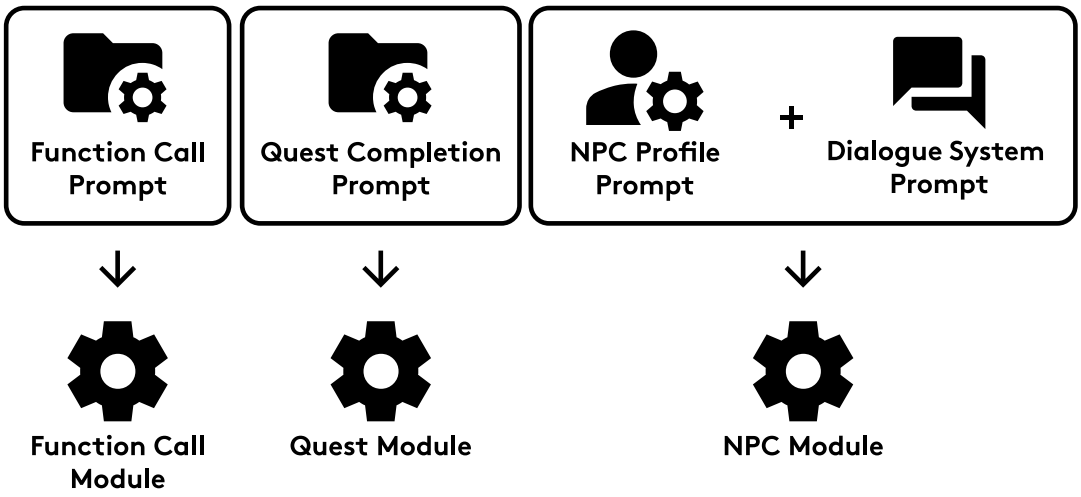


FIGURE 3.14: Prompt Structures Across Modules

Illustration showing how different prompt structures are used across modules.

Figure 3.15 illustrates the full interaction flow of these modules and how they support and interlink with each other. Once gameplay has commenced and a player initiates a dialogue with an NPC, the player message is first intercepted by the *Function Call Module*. The message is evaluated to determine whether a system-level action should be triggered. It is also at this stage that relevant information from the `functionCall` object may be appended, such as specific knowledge the NPC is intended to convey or contextual details relevant to the interaction.

The player message is subsequently forwarded to the *Quest Completion Module*, which checks whether the input meets the conditions to change the game's quest state. Again, relevant information may be appended if appropriate, such as a comment related to the current quest objective.

Finally, the message is routed to the *NPC Module*, which generates a natural, character-driven response that reflects the NPC's unique tone and personality. Importantly, any appended information is also incorporated by the *NPC Module* to ensure that it is communicated organically to the player. The resulting NPC message is then displayed to the player, from which point the player can continue the interaction.

This distribution of functionality enables the game to scale dialogue complexity without compromising narrative structure or system stability. Each module operates independently yet in a coordinated sequence, ensuring both modularity and

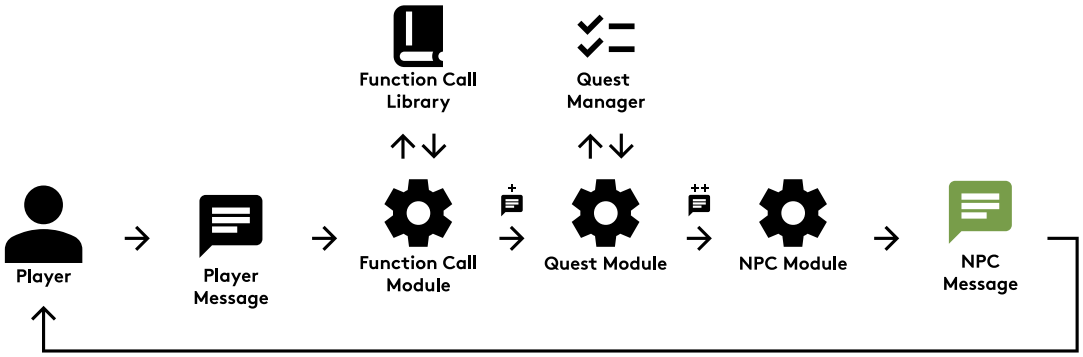


FIGURE 3.15: Module Integration Overview

Illustration showing the overall flow of the modular LLM integration, and visualizes the core loop defining an interaction between a player and an NPC.

contextual precision at every stage of the interaction.

3.3.4 Consistency Across Versions

Despite the varying LLM integrations and designs across the four game versions, the underlying narrative structure remains consistent. Considerable effort has been made to ensure that narrative progression is not compromised, with each version guiding the player through the same core storyline. This consistency is critical for the purposes of the study, as it ensures comparability across versions by maintaining a uniform storytelling experience. Accordingly, the quest logic, game objectives, and major narrative plots are identical across all versions. This is particularly important in *Version B* and *Version C*, where the higher degree of LLM influence could otherwise risk deviating from the intended narrative scope.

The *Control Version* serves as the narrative baseline, as it is the only version featuring entirely static, prewritten dialogue. As such, all other versions are designed to align closely with the narrative of the *Control Version* to preserve narrative coherence.

Version A modifies dialogue only at the surface level by rephrasing prewritten lines, thereby maintaining the original narrative flow.

Version B similarly uses context derived from the NPC's base dialogue to guide the generation of responses.

In contrast, *Version C* does not interact directly with the baseline dialogue; instead, it is prompted with relevant contextual information in advance. This approach was adopted to optimize LLM performance and increase NPC conversational flexibility, while still preserving the intended narrative structure.

By grounding the narrative flow across all versions, the study can more accurately examine qualitative and quantitative differences in player engagement and behavior without introducing invalid variables related to inconsistent storytelling.

Chapter 4

User Study

This chapter will cover the user study design, including which testing methods were applied and how the user data was collected. To test the game, usability and user testing methodologies were applied [66].

4.1 User Study Design

A user study was designed and conducted. The purpose was to test various parameters and gather relevant information on user behavior in correlation with the implementation of LLMs in dialogue systems. This approach can potentially prove quite complicated for the user, so it was decided that the test environment should be as linear as possible. Therefore, the desired data metrics were gathered autonomously while participants played the game. This meant that players would not need to actively submit data, allowing them to focus on playing the game as they normally would. The test protocol was designed in three phases:

1. An introductory briefing provided by a facilitator
2. A playthrough of the game
3. A post-game questionnaire

The experimental approach for the user study was based on a between-subjects study design [67]. This approach exposes each test participant to only one version of the game. This eliminates the potential bias of the user comparing the different versions, with potential answers focusing on the differences between each version of the game. Moving a test participant from one version to another can make them explicitly aware of the change in the environment. In addition, the test participant will also notice the changes between the two versions and therefore naturally start comparing the different versions [67]. For this study, each version of the game is tested independently and then compared during the data analysis phase.

The test environment was required to be a relatively isolated room, free of noise and other distractions. The setup was designed for multiple simultaneous participants [47]. The game ran on computers that were set up and prepared by the test facilitator. Before players were allowed to start the game, a short but detailed introduction was provided. This was done to comply with general ethics for user testing, as it was expected that gaining informed consent from the participants, as part of ethical conduct, would yield better quality test results [68]. This conduct requires that participants are well informed about the experiment, what data is collected, how their data is processed, and how they can retract consent for the data they provided post-test. An example of a test session can be seen on Figure 4.1.

FIGURE 4.1: Picture of a test session. Here two participants are doing a playthrough of the game

To accommodate these expectations and inform the test participants, a test protocol was created, see Appendix B. The test protocol paper is read aloud before the playthrough and it explains the content of the test, what data is extracted, and what is expected of the test participants. These premises are based on generally recognized rules for setting up a user test procedure [47]. The protocol also outlines the rules of engagement for the test participants, the role of the facilitators, and the procedure to follow if any errors occur or if participants have trouble proceeding.

For this test, the participants were told that under no circumstances could they do anything wrong, and if they were stuck or encountered an error, the test facilitator would intervene to solve the problem. The players were also made aware of the time they were expected to spend on the test, but they were also informed that they could take as much time as needed. At the same time, the test facilitator would also observe from a distance to ensure the test proceeded according to protocol. Here, the participants were also asked to play the game as they normally would and not to try to break it on purpose, as the game was still at a high-fidelity prototype level, where unknown errors could occur. The participants were also told not to think out loud while playing the game, but rather to save their feedback for the post-test questionnaire.

A few test participants were not able to be physically present, so a remote test protocol was prepared [47]. The protocols were almost identical, with the only difference being that the facilitator could observe the gameplay remotely, and the remote participants had to start the game themselves. The remote players were also asked to delete the game after the test. This was to ensure that they would not be able to replay the game and submit more data.

4.1.1 Questionnaire Design

When the participants had completed the game, they were directed to an online questionnaire designed to gather direct feedback. The methodological approach to the questionnaire involved formulating and presenting questions to collect relevant data, specifically to survey and profile the information in order to identify and explore patterns within it [69]. The questions were also designed to measure responses without being leading or suggestive. Additionally, the rhetoric of the questions was

intended to appeal to all participants and reflect the research domain without using overly complex terminology. The questions were kept as short as possible and avoided topics that participants might find invasive or too personal [69].

4.1.2 Participant Recruitment and Demographics

An important aspect of conducting a survey is selecting the right test participants. Demographics play a crucial role in how the tested product is perceived and how it performs in practice compared to expectations. Participant representation should be as close to real-world users as possible. Therefore, participants should generally reflect individuals who would realistically use the game being tested [47]. The requirements for test participants were as follows: they must have a general knowledge of computer games and must understand and be able to write in English.

4.1.3 Random Assignment Procedure

Initially, it was decided that each test participant would be assigned a completely random version of the game. It was expected that as the tests were completed, the distribution of the different game versions would be roughly normal. However, the relatively low sampling size of the distribution had a risk of not being normal. To achieve a normal distribution of the different versions, an algorithm was implemented in the game. This algorithm added a random assignment procedure by using a stratified randomization method. This method checks the current distribution of the different versions for completed tests and assigns one of the least common version to new tests. This approach keeps the distribution somewhat random while ensuring a normal distribution.

4.2 Data Collection Methods

In order to analyze the differences in user interaction between the different dialogue systems, multiple types of data were collected to capture both in-game behavioral metrics and post-game subjective feedback. These data points can later be used for both qualitative and quantitative analysis. This section explains how the different data collection methods were set up, what they aimed to capture, and why this data was important.

This thesis presents two separate data collection methods: the behavioral data captured from the players' playthroughs, and their subjective feedback on the prototype, which was gathered through a post-game questionnaire. The questionnaire can be seen in Appendix A

4.2.1 Game Data

As the participants played through the prototype, the game saved several metrics to one of four NoSQL databases. The database to which the data was sent depended on the game version assigned to each participant. This made it easier to split up and analyze the data. Furthermore, each database was split into seven collections, each containing different types of game data. These seven collections of data were:

- **Dialogue Messages** - Holds all lines of dialogue between the NPCs and the player, including timestamps. This enables detailed analysis of conversation

flow and interaction depth. It also allows for in-depth thematic coding to analyze how participants communicated with the NPCs during their playthrough.

- **Dialogue Time** - Tracks how long a player was engaged in dialogue with an NPC, providing insight into the duration of interactions across dialogue systems. This is especially important, as dialogue times might differ significantly when conversing with an LLM.
- **Movement Data** - Records player movement throughout the world, useful for understanding exploration behavior and pacing. This data consisted of a list of player coordinates, sampled every 0.25 seconds.
- **Area Time** - Contains data on how long the players spent in each area. This allows for analysis of player focus and engagement with specific locations, and reveals whether different dialogue systems influenced the player's exploration patterns.
- **Area Transitions** - Contains data on how players moved between areas. This can help uncover common navigation patterns or points of confusion.
- **Other** - Miscellaneous data that does not fit into the main categories. This collection holds information such as total run duration, which quests were completed, the number of failed attempts per quest, the time it took players to complete quests individually, the time spent waiting for the LLM to load, the notes participants wrote down, and, specifically for *Version B*, whether the player selected a static or open-ended dialogue node.
- **Run Codes** - Includes the specific run codes that were attached to each playthrough. These run codes linked the data from the other collections and the questionnaire, making it easy to isolate data from a specific playthrough. They were randomly generated for each session and consisted of four random letters and numbers.

4.2.2 Questionnaire Data

When the participants were finished with the playthrough, closing the game opened up a questionnaire which was prefilled with the run information such as the run code and the game version. This could then later be used to split the questionnaire responses into groups in order to compare the qualitative results from each dialogue system. During the usability test, a pilot test was also performed with three users to validate the logging functionality and questionnaire flow.

All data collection was conducted in accordance with ethical guidelines[69]. Participants provided informed consent, and all identifying information was either anonymized or stored with a unique code to allow for deletion upon request.

The majority of the questionnaire consisted of 5-point Likert-scale[70] questions (24), designed to provide quantitative data. The final four questions were open-ended, allowing participants to give qualitative feedback. Nine of the questions were skippable. These included the open-ended questions at the end, as well as the questions concerning the difficulty of specific quests. Participants were instructed to skip these if they had not completed the corresponding quests.

The questionnaire contained nine sections each covering a separate topic. The sections were as follows:

- **Section 1: Consent** - Collects participant consent and a unique identifier to allow for later data deletion if requested.
- **Section 2: Player Background** - Gathers demographic and gaming-related background information, including age, gender, play habits, and familiarity with RPGs and AI dialogue systems.
- **Section 3: Technical Info** - Prefilled data from the game which allows for grouping the questionnaire data. This section was hidden from the participants.
- **Section 4: General Experience** - Evaluates the participant's overall engagement and enjoyment of the game.
- **Section 5: Dialogue System Evaluation** - Assesses the player's perception of the dialogue system's quality, control, and naturalness during interactions with NPCs.
- **Section 6: Clarity and Engagement** - Measures how clearly information was conveyed through dialogue and how engaging the information-gathering process was.
- **Section 7: Interest and AI Impressions** - Measures participants' interest in playing more of the game or similar games, and their impressions of AI-generated dialogue.
- **Section 8: Quest-Specific Challenges** - Identifies how difficult specific in-game challenges were perceived to be by the participants.
- **Section 9: Feedback** - Provides space for open-ended responses regarding positive and negative feedback, technical issues, and general suggestions.

4.3 Data Analysis Methods

To evaluate the impact of the different dialogue systems, both quantitative and qualitative data collected during the study were systematically analyzed using a combination of statistical testing and thematic coding. The following section will go into further detail on how the different data was treated and the tools used to do so.

4.3.1 Quantitative Analysis

Most of the data collected from gameplay metrics and the questionnaire was analyzed using statistical methods. To compare the different dialogue systems, the data was split into four groups based on the version each participant played. This made it possible to take a closer look at how the different systems affected player behavior and responses.

To ensure a solid base for comparison, the study aimed for at least 15-20 participants per dialogue version. This provided sufficient statistical power to detect medium-sized effects, with effect sizes reported using eta squared (η^2) [71].

Since there were four groups, the primary statistical method used was a one-way ANOVA to compare means across conditions. For measures that violated ANOVA assumptions, especially the Likert-scale questionnaire responses, the non-parametric Kruskal-Wallis test was applied instead. In cases of statistical significance ($p < .05$), appropriate post-hoc tests were conducted: Tukey's HSD for

ANOVA results and pairwise Wilcoxon tests for Kruskal–Wallis results, with p-values adjusted for multiple comparisons.

The following variables were included in the analysis:

Metric	**Description**
Gameplay Time	Average total time spent in-game
Quest Completion Time	Average time taken to complete each quest
LLM Loading Time	Average time waiting for the LLM to respond
Interaction Amount	Average number of interactions initiated
Response Count	Average number of responses per interaction
Quest Failures	Average number of total failed quest attempts
Dialogue Time (Overall)	Average total dialogue time per run
Dialogue Time (Per NPC)	Average dialogue time per NPC
Notebook Usage	Percentage of participants who used the notebook
Notebook Words	Average number of words written in the notebook
Questionnaire Data	Likert-scale responses from post-game questionnaire

TABLE 4.1: Overview of the main quantitative metrics analyzed

While the main analysis focused on differences between the dialogue versions, additional insights were gained by grouping participants based on individual characteristics. Participants were categorized as *casual players* if they reported playing 10 hours or less per week, and as *hardcore players* if they exceeded that threshold. Other groupings considered whether participants enjoy games with a strong dialogue focus, and their familiarity with RPGs. This enabled analysis of how different player backgrounds may have influenced in-game behavior and questionnaire responses.

4.3.2 Qualitative Analysis

Thematic coding [72] was used to analyze qualitative data from the four open-ended questions of the post-game questionnaire. This method is well-suited for synthesizing large amounts of textual data into something quantifiable. The responses were carefully read in an initial pass to gain an overall sense of what participants had written. Afterwards, codes were assigned to each response, describing the content of the answers. These codes were then grouped into broader themes, making it possible to count how many participants mentioned each theme. Following the initial human coding, a second pass was conducted with the assistance of an LLM. The results from both passes were compared to ensure consistency and reduce potential bias in the coding process. Identified themes were used to supplement the quantitative findings and to explore differences in player experiences across dialogue versions.

Furthermore, a manual classification was performed on all open-ended player responses in *Version B* and *Version C* to determine whether participants engaged with the system in character or attempted to outsmart the LLM. While the categorization process was supported by an LLM, all labels and interpretations were reviewed and verified manually to ensure accuracy and contextual relevance.

4.3.3 Tools and Software

The analysis was conducted using Python, primarily through the use of Pandas[73] for data handling, SciPy[74] and Statsmodels[75] for statistical testing, and Matplotlib[76] for data visualization. Additionally, NumPy[77] was used for specific

tasks, including descriptive statistics and effect size calculation.

Game metrics were retrieved from a MongoDB database, and questionnaire responses were exported to a CSV file. Both datasets were cleaned and processed using Pandas before analysis.

Chapter 5

Results

5.1 Descriptive Statistics

The results of the tests encompassed 64 participants. However, two participants did not complete the full game due to time constraints and were therefore excluded from the final analysis. Of the remaining participants, 77.4% identified as male and 22.6% as female. The mean age was 25.2 years ($SD = 4.57$). Additionally, 11.3% of the participants reported having dyslexia, and 3.2% were color blind.

Regarding weekly gaming habits, the largest group of participants (32.3%) reported playing between 6 and 10 hours per week. An equal proportion (22.6%) played either between 11 and 20 hours, or more than 20 hours weekly. A smaller group (14.5%) played between 1 and 5 hours, while only 8.1% reported playing less than 1 hour per week.

A large portion of participants were already well familiar with RPGs, as reflected by a high mean score ($M = 4.34$, $SD = 1.10$). In contrast, participants reported less familiarity with LLM-powered NPC dialogue in games ($M = 2.35$, $SD = 1.33$). Most participants also indicated that they enjoy games where dialogue plays an important role, with a mean score of ($M = 3.89$, $SD = 1.07$).

In total, the *Control Version* and *Version B* each had 16 participants, while *Version A* and *Version C* each had 15 participants.

Demographic Variable	**Value**
Total Participants	64 (62 analyzed)
Mean Age (SD)	25.2 (4.57)
Gender	77.4% Male, 22.6% Female
Dyslexia Reported	11.3%
Color Blindness Reported	3.2%
Participants per Version	Control: 16, A: 15, B: 16, C: 15

TABLE 5.1: Participant Demographics Summary

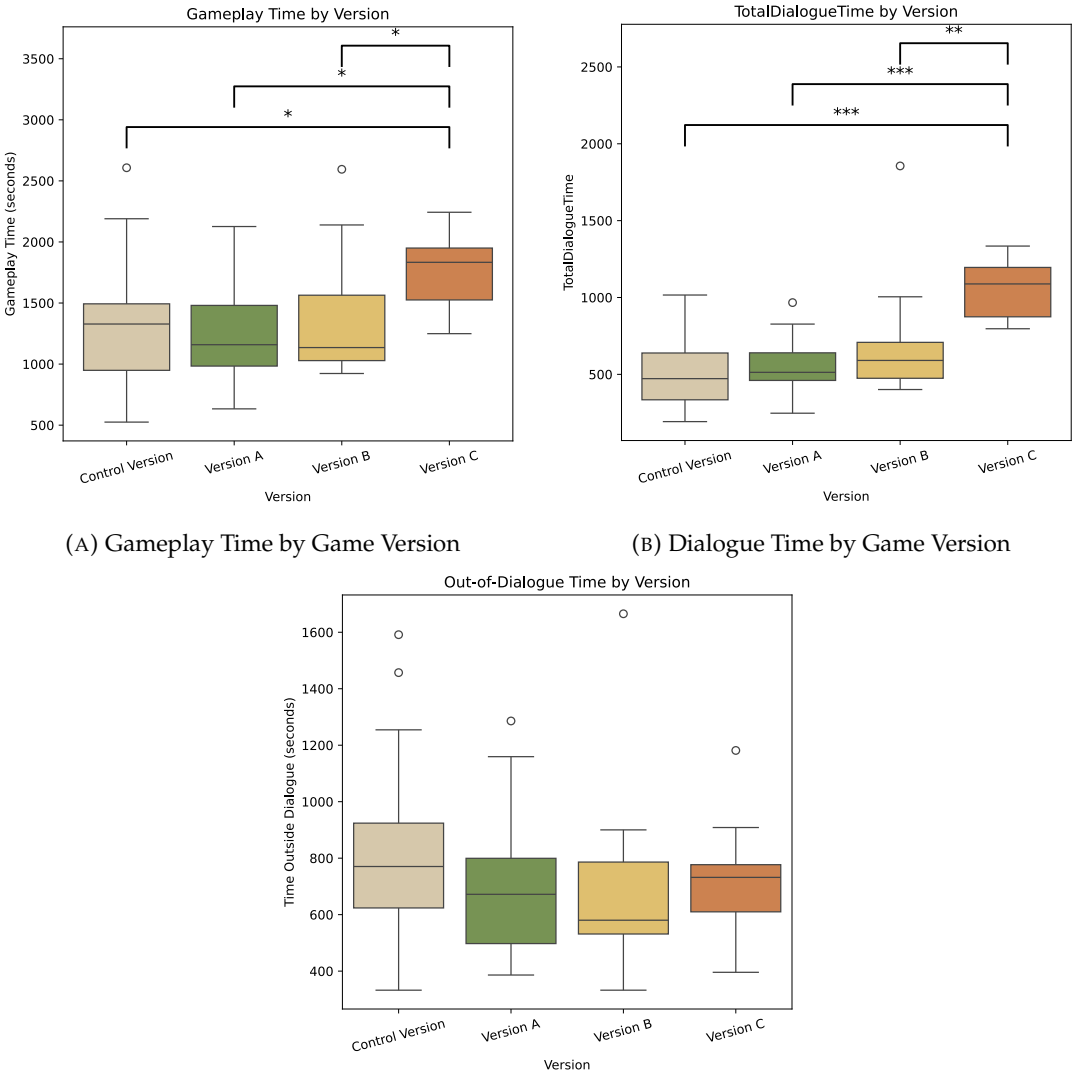
5.2 Quantitative Results

This section presents the results of the quantitative behavioral data collected from the different game versions, along with responses to the Likert scale questions from the post-test questionnaire. The data is analyzed by game version to identify any significant differences. Additionally, results are filtered based on participant information, such as weekly gaming hours, and any notable findings from these subgroup

analyses are highlighted. The section is organized by the different metrics analyzed and will be discussed and interpreted further in the Interpretation of Results section.

5.2.1 Gameplay Times

Participants took significantly longer to complete *Version C* compared to all other versions. A Kruskal–Wallis test revealed a **statistically significant difference** in completion times ($p = 0.0064$), with a moderate effect size ($\eta^2 = 0.152$). Post-hoc analysis using Dunn's Test indicated that *Version C* differed significantly from the other versions: ***Control Version* vs. *Version C* ($p = 0.0334$)**, ***Version A* vs. *Version C* ($p = 0.0122$)**, and ***Version B* vs. *Version C* ($p = 0.0398$)**. This is plotted in Figure 5.1a.



(A) Gameplay Time by Game Version

(B) Dialogue Time by Game Version

(C) Out-of-Dialogue Time by Game Version

FIGURE 5.1: Comparison of total gameplay time, dialogue time, and out-of-dialogue time across all game versions.

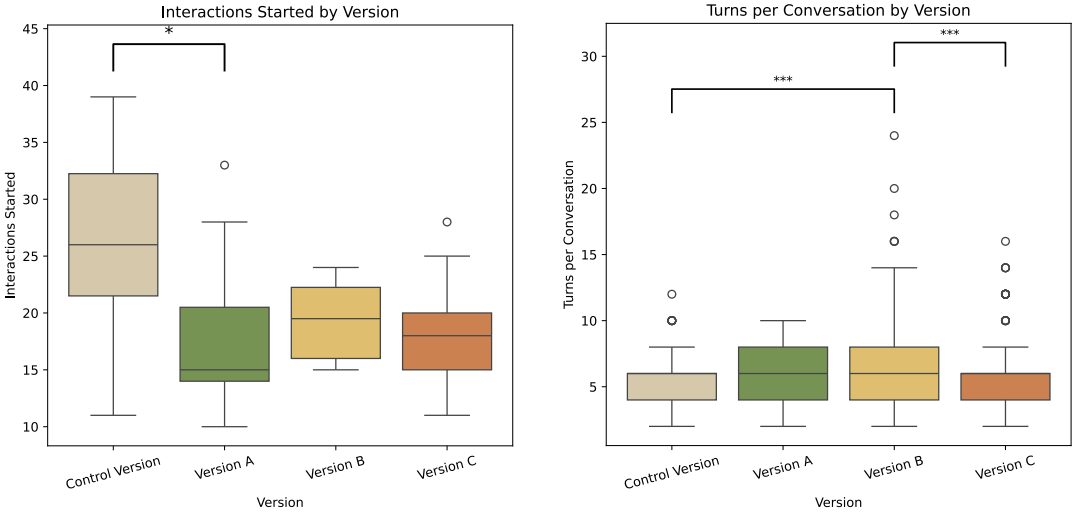
When filtering the data to include only casual gamer participants, no significant differences were found in total gameplay time across the different game versions (ANOVA, $p = 0.5481$).

Gameplay time did not significantly differ between casual and hardcore players. Similarly, no significant differences in gameplay time were observed between participants who enjoyed dialogue-focused games and those who did not.

5.2.2 Dialogue

Similar to gameplay time, total dialogue time differed significantly across the game versions, with *Version C* again showing notably higher values. A Kruskal–Wallis test revealed a **highly significant difference** ($p < .001$) with a large effect size ($\eta^2 = 0.392$). Dunn's post-hoc test identified significant pairwise differences: ***Control Version* vs. *Version C* ($p < .001$)**, ***Version A* vs. *Version C* ($p = 0.0003$)**, and ***Version B* vs. *Version C* ($p = 0.0048$)**. This is plotted in Figure 5.1b.

Dialogue time did not significantly differ between casual and hardcore players. Similarly, no significant differences in total dialogue time were found between participants who enjoy games where dialogue is important and those who do not.



(A) Interactions Started by Game Version

(B) Response Count per Game Version

FIGURE 5.2: Comparison of interactions started and response count across all game versions.

The number of interactions initiated by players differed significantly between the *Control Version* and *Version A*. A Kruskal–Wallis test revealed a **statistically significant difference** ($p = 0.0105$, $\eta^2 = 0.135$), and a post-hoc Dunn's test confirmed that participants in the *Control Version* initiated significantly more interactions than those in *Version A* ($p = 0.0102$). This is plotted in Figure 5.2a.

However, once interactions were initiated, players tended to engage in longer conversations with NPCs in *Version B*. On average, players in *Version B* had 6.46 turns per conversation ($SD = 3.21$), compared to 5.31 in the *Control Version* ($SD = 1.99$) and 5.53 in *Version C* ($SD = 2.76$). A Kruskal–Wallis test confirmed a **significant overall difference** in the number of dialogue turns ($p < .0001$, $\eta^2 = 0.024$), with Dunn's post-hoc test showing significant differences between ***Control Version* and *Version B* ($p < .001$)** and ***Version B* and *Version C* ($p < .001$)**. This is plotted in Figure 5.2b. These results suggest that while *Version B* did not lead to more interactions being initiated, it encouraged longer conversations once they began.

No significant differences were found between casual and hardcore players, nor between participants who enjoy games where dialogue is important and those who do not.

When analyzing dialogue time with individual NPCs across game versions, several patterns emerged. Significant differences were primarily associated with *Version C*, which consistently showed longer dialogue durations for key NPCs. Table 5.2 summarizes the key statistical findings for each NPC.

NPC	Kruskal-Wallis p	Effect Size (η^2)	Sig. Pairwise Comparisons
Bard	0.7554	-0.030	None
Blacksmith	< 0.001	0.314	CV vs C ($p < 0.001$) A vs C ($p = 0.0043$) B vs C ($p = 0.0064$)
Commoners	0.8726 (ANOVA)	0.012	None
Goblin	0.3104	0.010	None
Herbalist	0.1742	0.032	None
Hobgoblin	< 0.001	0.464	CV vs C ($p < 0.001$) A vs C ($p < 0.001$) B vs C ($p < 0.001$)
Huntsman	< 0.001	0.334	CV vs C ($p < 0.001$) A vs C ($p = 0.0106$) B vs C ($p < 0.001$)
Innkeeper	0.0114	0.132	CV vs C ($p = 0.0135$) A vs C ($p = 0.0439$)
Magic Barrier	< 0.001	0.274	CV vs C ($p < 0.001$) A vs C ($p = 0.0176$) B vs C ($p = 0.0070$)
Thief	0.0042	0.168	CV vs C ($p = 0.0019$) A vs C ($p = 0.0493$)
Wizard	< 0.001	0.324	CV vs C ($p < 0.001$) A vs C ($p = 0.0017$)

TABLE 5.2: Summary of Dialogue Time Differences Across Game Versions for Each NPC

5.2.3 Time Outside Dialogue

By subtracting the total amount of dialogue time from the total gameplay time, no significant changes were found (Kruskal–Wallis, $p = 0.3739$), meaning each game version took an equally long time to complete if no dialogue system was implemented. This can be seen plotted in Figure 5.1c.

5.2.4 Quest Completion and Failure Rates

Table 5.3 summarizes the time participants spent completing various quests across game versions. While most quests did not show statistically significant differences, the quest *Break the Barrier* stood out with a **strong effect** ($p < 0.001$, $\eta^2 = 0.234$). Post-hoc comparisons revealed that participants in *Version C* spent significantly more time on this quest compared to the ***Control Version*** **(**$p = 0.0450$**)**, ***Version A*** **(**$p < 0.001$**)**, and ***Version B*** **(**$p = 0.0133$**)**. Although *Grab a Refreshment* also reached significance ($p = 0.0428$), no specific pairwise differences were found. The remaining quests,

Quest	Kruskal-Wallis p	Effect Size (η^2)	Sig. Pairwise Comparisons
Bard's Tale	0.1326	0.043	None
Innkeeper's Knowledge	0.0999	0.053	None
Navigate the Woods	0.9779	−0.046	None
Grab a Refreshment	0.0428	0.085	Not specified
Break the Barrier	< 0.001	0.234	CV vs C ($p = 0.0450$) A vs C ($p < 0.001$) B vs C ($p = 0.0133$)
I'm Going on an Adventure!	0.1388	0.041	None

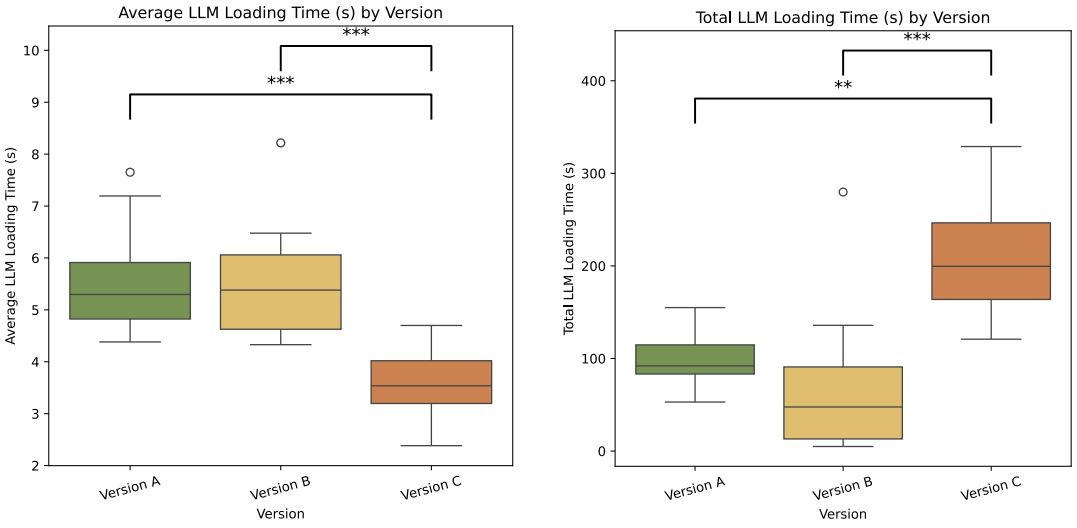
TABLE 5.3: Summary of Quest Completion Time Differences Across Game Versions

including *Bard's Tale*, *Innkeeper's Knowledge*, and *Navigate the Woods*, did not differ significantly across versions.

The failure rate of the different quests did not differ significantly across the four game versions.

5.2.5 LLM Loading Times

Comparing the average LLM loading time that participants had to wait during individual responses, there was a **strong and statistically significant difference** between *Version C* and both *Version A* and *Version B*. An ANOVA revealed a **highly significant effect** ($p < 0.001$, $\eta^2 = 0.516$), with post-hoc comparisons showing that *Version C* had **significantly lower average loading times** than ***Version A*** **(**$p < 0.001$**)** and ***Version B*** **(**$p < 0.001$**)**. This can be seen plotted in Figure 5.3a.



(A) Single LLM Loading Time by Game Version

(B) Total LLM Loading Time by Game Version

FIGURE 5.3: Comparison of an average LLM loading time, and the total LLM loading time across all game versions.

However, when looking at the total LLM loading time accumulated across a full playthrough, *Version C* stood out with **significantly higher values** compared to the other two versions. A Kruskal–Wallis test confirmed a **strong effect** ($p < 0.001$, $\eta^2 = 0.544$), with pairwise comparisons showing significant differences between ***Version A* and *Version C*** **(**$p = 0.0013$**)**, as well as between ***Version B* and *Version C*** **(**$p < 0.001$**)**. These results are plotted in Figure 5.3b.

Survey Question	**Control**	***Version A***	***Version B***	***Version C***
The game was engaging and kept my interest throughout.	3.94	3.67	4.19	4.20
I found the NPCs interesting to talk to.	3.25	3.00	3.75	4.00
I felt motivated to gather information from NPCs.	3.94	3.80	3.94	3.93
The game's dialogue system made the experience enjoyable.	3.75	3.47	4.12	4.27
I felt in control of my interactions with NPCs.	3.75	3.53	3.12	3.87
The dialogue system allowed for meaningful conversations.	3.31	3.00	3.44	3.60
I felt the NPCs reacted naturally to my dialogue choices.	4.00	3.93	3.44	3.87
NPCs provided key information in an immersive way.	4.00	4.27	3.94	4.00
I felt encouraged to experiment with dialogue.	3.19	3.53	3.94	4.07
There were moments the dialogue felt unnatural or confusing.	1.94	2.00	3.19	2.87
After speaking with NPCs, I knew what to do next.	3.88	4.53	4.25	4.00
Finding and using info from NPCs was engaging.	4.19	3.93	3.94	4.07
NPCs conveyed information consistently.	4.19	4.20	3.75	4.20
I would be interested in playing a longer version of this game.	4.00	3.67	4.50	4.47
I would be interested in games with similar dialogue systems.	3.88	3.60	4.31	4.13
The dialogue system increased my interest in AI-powered NPCs.	3.19	3.07	3.75	3.93

TABLE 5.4: Likert-scale Survey Averages by Game Version

5.2.6 Questionnaire Responses

By splitting the post-game questionnaire data by game version, it is possible to better understand how participants perceived their experience with each game version. Table 5.4 presents the average responses for each survey item across all four versions.

A **significant difference** was found in the item *"I found the NPCs interesting to talk to"*, with a Kruskal–Wallis test revealing $p = 0.0289$ and an effect size of $\eta^2 = 0.099$. Dunn's post-hoc test showed a **significant pairwise difference** between ***Version A***

and *Version C* ($p = 0.0483$), suggesting that participants found the NPCs in *Version C* more engaging than those in *Version A*.

Further significant differences were observed in two additional questions. The statement *"The game's dialogue system made the experience enjoyable"* yielded a **significant overall result** ($p = 0.0402$, $\eta^2 = 0.087$), as did the statement *"There were moments when the dialogue system felt unnatural or confusing"* ($p = 0.0207$, $\eta^2 = 0.111$). However, in both cases, no individual pairwise comparisons reached significance in the post-hoc analysis.

As a notable mention, the statements *"The dialogue system in this game made me more interested in AI-powered NPCs in games"* and *"I would be interested in playing a longer version of this game"* both produced **relatively low p-values** in the Kruskal–Wallis test. The former yielded $p = 0.0661$ with an effect size of $\eta^2 = 0.069$, while the latter had $p = 0.0651$ and an identical effect size of $\eta^2 = 0.069$.

The difficulty of the quests was also evaluated in the post-game questionnaire. However, these responses did not show any significant differences across the different game versions. The average perceived difficulty for each quest is presented in Table 5.5.

Quest Difficulty	**Control**	***Version A***	***Version B***	***Version C***
Bribing Fizzby the goblin	1.69	1.36	1.71	2.10
Navigating the forest	2.27	2.80	1.75	2.40
Consuming the correct berries	1.94	2.27	1.88	1.87
Disabling the Magic Barrier	1.50	1.47	1.44	1.53
Persuading the Hobgoblin	2.44	1.79	1.88	2.67

TABLE 5.5: Quest Difficulty Averages by Game Version

5.3 Qualitative Results

The following subsections present an analysis of the open-ended responses provided in the post-game questionnaire. The questionnaire included four open-ended questions, allowing participants to share positive and negative feedback and suggest improvements for future versions. One of the questions also asked participants to report any bugs or technical issues, but these responses were only used to identify persistent issues during testing and are not included in the coded analysis. To provide clearer insight into how each dialogue system was received, the remaining qualitative responses have been grouped by game version.

5.3.1 Positive Feedback

Positive feedback from the participants were coded and grouped into four themes. These themes, along with their concise descriptions and frequency counts for each game version, are presented in Table 5.6.

The *Control Version* was praised for being clear, coherent, and easy to use. Several participants mentioned that quest information was easy to find and that conversations felt natural and fluid. Others liked the straightforward design, calling it intuitive and familiar. While it lacked the flexibility of the other versions, one participant still felt it offered more choice than typical dialogue systems.

Theme	Concise description	Ctrl	A	B	C
Immersion & Natural Dialogue	Fluent, human-like NPC speech	6	5	5	5
Player Agency & Flexibility	Free-text and hybrid input	1	0	6	4
Guidance & Quest Clarity	Clear hints and key points	3	2	1	3
Novelty & Usability	Fresh concept, intuitive UI	4	1	1	1

TABLE 5.6: Positive feedback, themes, concise descriptions, and frequency counts per game Version

Version A received positive feedback for its natural and consistent dialogue. Participants noted that NPCs spoke in a normal, engaging way and did not feel off-tone. Some appreciated that exploration was rewarded with useful information, and one found the system simple and clear. Overall, it was seen as immersive and easy to follow.

Players of *Version B* highlighted the mix of free-text input and fixed options as a major strength. Many felt the system gave them room to be creative and that NPCs reacted well to their own phrasing. The dialogue was often described as fluid and engaging, and a few liked how it reminded them of older games. Overall, it struck a good balance between freedom and structure.

Version C stood out for giving players the most freedom. Participants liked being able to type their own responses and felt it made interactions more personal. Dialogue was described as natural, and several noted that NPCs had distinct personalities and gave clear quest information. While a few mentioned quirks, the overall experience was seen as engaging.

5.3.2 Negative Feedback

Negative feedback from the participants was coded and grouped into four themes. These themes, along with their concise descriptions and frequency counts for each game version, are presented in Table 5.7.

Theme	Concise description	Ctrl	A	B	C
Dialogue Bloat & Repetition	Dialogue feels too long or repetitive	1	0	1	1
Incorrect or Misleading Info	NPCs give wrong or misleading answers	3	0	0	2
Responsiveness & Agency Limitations	NPCs ignore typed input or choices limited	7	7	10	7
Usability & Readability	UI / visual issues hinder readability	2	0	3	2

TABLE 5.7: Negative feedback, themes and frequency counts per game version

In the *Control Version*, most negative feedback was about limited agency. Several participants said NPCs ignored player responses or that the options felt too restrictive. Others mentioned issues with readability or UI, and a few pointed out incorrect or confusing information. One also felt the dialogue was bloated.

For *Version A*, the main issue was again that NPCs did not respond properly to player choices. Some felt the dialogue lacked depth, or that the NPCs came off as too similar. One also mentioned the dialogue could get bloated.

Version B had the same core problem; NPCs not reacting to player input. A few also noted UI issues, repeated lines, or incorrect information. Despite the hybrid format, the system did not always respond in meaningful ways.

Version C gave players more freedom, but that did not always translate into better responses. Several said their input was ignored or not really understood. Some pointed out misleading information, bloated lines, or awkward UI. A few noted that, even with open input, it did not always feel like they had real control.

5.3.3 Feature Suggestions

Suggestions from the participants was coded and grouped into four themes. These themes, along with their concise descriptions and frequency counts for each game version, are presented in Table 5.8.

Theme	Concise description	Ctrl	A	B	C
UI & QoL Improvements	Interface tweaks, map/compass, notebook hot-keys	8	9	2	8
Movement & Camera	Sprint/run and camera/lag fixes	5	1	2	1
Story & Challenge Depth	Expand story, harder riddles, clearer rewards	3	2	2	3
Dialogue & Gameplay Mechanics	Free-typing rules, NPC memory, option purpose	1	1	3	1

TABLE 5.8: Feature Suggestion themes, concise descriptions, and frequency counts per Game Version

Participants across all versions offered a range of suggestions aiming at improving the overall experience. For the *Control Version*, most suggestions focused on UI improvements and navigation, such as better map features, clearer notebook controls, and fixing camera issues. Some also wanted more challenge and variety in the story, including harder riddles and hidden content.

In *Version A*, players asked for better navigation tools like a visible compass or zoom controls, and several pointed out small UI improvements. There were also a few suggestions tied to humor and pushing the visual quality further. The dialogue system was generally well-received, but some wanted clearer branching logic.

Version B feedback included requests for stronger story depth and better camera control. Many comments centered on dialogue mechanics, like forcing free-text in puzzles, using non-verbal reactions, and adding memory to past conversations. UI requests included color-coding NPCs, bookmarking, and quick access to NPCs.

For *Version C*, players had ideas for improving both usability and content. Suggestions included adding inventory, clearer NPC rewards, and letting players copy chat text. Several also mentioned enhancing the story and making it more substantial. Movement options like jumping or better camera control were also requested, along with quality-of-life features like a search bar and color-blind support.

5.4 In-game Player Responses

Participants playing with dialogue *Version B* and *Version C* had the opportunity to write their own responses to NPCs. This gave them the freedom to shape conversations however they liked. While this level of freedom allowed for creativity, it also opened the door for players to potentially test or break the system.

5.4.1 Version B

In *Version B*, 18.34% of all interactions used the open dialogue option. Out of 938 total responses, 172 were written freely by participants. Only a small number of these attempted to push the limits of the system, for example by referencing real-world topics or trying to confuse the AI. A few unusual inputs like *"iphone"*, *"avatar"*, and the sequence *"2"*, *"2"*, *"3"*, *"1"* were found; the latter likely from a participant who misunderstood the input field and thought they had to type in their dialogue choice manually. Similarly, one participant who reported playing less than one hour per week typed the static dialogue options into the custom answer input, thinking that was how they were supposed to respond. Overall, these cases were rare, and most players engaged with the system as intended.

Most of the open-ended responses were meaningful and stayed within the game world. About 45% were short, in-character statements like *"we are doomed"* or *"sounds risky"*. Around 34% were follow-up questions aimed at clarifying or challenging the NPC, such as *"how do you know it was goblins?"* or *"what happens if I refuse?"*. A smaller portion (14%) expressed gratitude or agreement, for example *"thanks, that helps"* or *"good idea"*. The rest included greetings, quick affirmations, or simple requests. These patterns suggest that players used the feature to enrich their role-play rather than to break immersion.

5.4.2 Version C

In *Version C*, a total of 607 open-ended responses were collected. Unlike *Version B*, this *Version* allowed fully open dialogue throughout the entire game, with no fixed prompts. A manual review of all entries showed that over 95% of the messages were written in character, indicating that players understood and respected the context of the game world.

Most responses were short statements or questions that aimed to gather more information, such as *"what do you know about the stolen food?"*, *"would you call yourself the huntsman?"*, and *"do you know where they went?"*. These show players were role-playing actively and focusing on advancing the quest. Some responses offered help or confirmed what the NPC had said, like *"let's do it"* or *"I can help you with that"*. Greetings and sign-offs were rare, and only a few entries showed any sign of meta-thinking or immersion-breaking content.

Overall, the fully open system in *Version C* encouraged players to engage naturally while staying grounded in the story. The results suggest that participants embraced the freedom to write their own dialogue without stepping outside the bounds of the game world.

Chapter 6

Discussion

6.1 Summary of Findings

This section summarizes the results of the previous chapter. The results reveal several significant differences across the four dialogue versions tested in the study. Overall, *Version C* consistently stood out in both the objective gameplay metrics and the subjective participant feedback.

- **Gameplay Time:** Participants took significantly longer to complete *Version C* compared to the other versions. However, when dialogue time was subtracted, the remaining gameplay time did not differ, suggesting that the added duration was due to more extensive conversations. For casual players, completion time was similar across all versions, although dialogue time was still significantly higher in *Version C*.
- **Dialogue Interactions:** Total dialogue time was significantly higher in *Version C*, with a large effect size, indicating deeper engagement with the dialogue system. The Control Version led to more frequent interaction initiations, while *Version B* had longer dialogues per exchange. *Version C* also had significantly longer dialogue durations with several individual NPCs, including the Blacksmith, Hobgoblin, Huntsman, Innkeeper, Magic Barrier, Thief, and Wizard.
- **Quest Completion:** Most quests showed no significant variation in completion time or failure rate. However, the quest *Break the Barrier* took significantly longer in *Version C*.
- **LLM Loading Times:** *Version C* had the shortest average loading time per response, but the highest total LLM loading time across a full playthrough.
- **Survey Feedback:** Participants rated the NPCs in *Version C* as significantly more interesting to talk to than those in *Version A*. There were also significant differences in how enjoyable the dialogue system felt and in reports of moments where the dialogue felt unnatural or confusing.
- **Qualitative Feedback:** Positive comments for *Version B* and *Version C* often mentioned immersion, agency, and flexibility. However, across all versions participants pointed out recurring issues such as limited NPC responsiveness, occasional misinformation, and user interface challenges. Suggestions for improvement largely focused on UI design, quest clarity, and dialogue functionality.
- **Player-Written Responses:** In *Version B*, 18% of all player responses were written freely, with most being short and in-character. In *Version C*, over 95% of the

responses remained grounded in the game world, suggesting that the open-ended system supported creative yet contextually appropriate role-play.

6.2 Interpretation of Results

This section interprets the statistical and qualitative findings gathered from the user study. Quantitative data from in-game metrics and Likert-scale questions, along with qualitative responses from open-ended questions, were collected and summarized. While each dataset offers value on its own, combining them provides a more complete picture of how the different dialogue systems influenced player interaction and perception.

Overall, the different dialogue systems led to noticeably different outcomes in terms of both player experience and perception. *Version C* resulted in the most engaging conversations, while *Version B* provided a balanced mix of structure and player agency. In contrast, *Version A* showed limited improvement over the *Control Version* and in some areas performed worse. Table 6.1 summarizes the key differences between systems across overarching metrics such as engagement, flexibility, stability, and technical performance. The following subsections break down the results for each dialogue version and examine how they influenced the players' experience with the game.

Metric	**Control Version**	***Version A***	***Version B***	***Version C***
Engagement	+	-	++	+++
Flexibility	+	+	++	+++
Stability	+	+	- -	-
Technical Performance	+	-	- -	-
Player Agency	+	+	++	+++
Perceived Naturalness	+	+	±	++
Dialogue Control (Dev-side)	+	±	±	- -

TABLE 6.1: Comparative summary of each dialogue version across key metrics

The comparison is relative to the Control Version. '+' indicates baseline or acceptable performance, with increasing or decreasing signs indicating stronger or weaker outcomes.

6.2.1 Impact of Dialogue Systems on Player Experience

Control Version

Strengths The *Control Version* served as a baseline for understanding how players interacted with the game using a fully traditional dialogue system. Quantitative responses and in-game metrics suggest that it provided a stable and reliable experience with minimal technical issues. Compared to *Version A*, the *Control Version* had significantly more initiated interactions, possibly due to the absence of LLM loading times. Players were able to move through dialogue quickly and without interruption, which may have contributed to smoother gameplay and a stronger sense of control.

Weaknesses Despite its stability, the *Control Version* was limited in terms of engagement. Some players noted that the static dialogue trees did not offer much

depth or insight into the NPCs' personalities, which may have reduced immersion. A recurring issue also emerged around the introductory quests involving the Bard and the Innkeeper. Although players were informed about the quest log, many still spent a considerable amount of time on these early tasks, suggesting that the guidance and onboarding could have been clearer or more intuitive. Overall, while functional and consistent, the *Control Version* lacked the flexibility and dynamism seen in the other systems.

Version A

Strengths *Version A* introduced a paraphrasing feature that reworded NPC dialogue each time a conversation was initiated. The system preserved the original dialogue structure and ensured that key information remained consistent. This approach maintained narrative control while attempting to make repeated interactions feel more dynamic. For players who spoke to the same NPC multiple times, the varying phrasings may have contributed to a slightly more natural feel compared to the fixed *Control Version*.

Weaknesses Despite these intentions, *Version A* received the least approval of all the dialogue systems and, in some cases, performed worse than the *Control Version*. A key issue was the LLM loading time, which averaged just over five seconds at the start of each new conversation. Since the underlying content was nearly identical to the *Control Version*, this delay introduced friction without offering a meaningful benefit. Questionnaire results reflect this lack of enthusiasm, with the lowest scores on *"I would be interested in playing a longer version of this game"* (3.67) and *"I would be interested in games with similar dialogue systems"* (3.60). Furthermore, the design relied on players returning to the same NPCs to notice paraphrasing differences. Many participants received the necessary information during their first interaction and did not revisit the same characters, meaning they never encountered the system's intended variation. This design limitation, combined with the added delay, likely contributed to its lower reception.

Version B

Strengths *Version B* was praised as a promising alternative to traditional dialogue systems. The open dialogue option was used in just under one in five messages, and participants noted that it successfully combined the structure of static dialogue with the freedom to express themselves more naturally. This hybrid approach was seen as innovative and engaging. It received the highest scores on the statements *"I would be interested in playing a longer version of this game"* (4.50) and *"I would be interested in games with similar dialogue systems"* (4.31). Players often used the custom input field to ask follow-up questions, especially when they encountered challenges. Since no external help was provided in the study, many turned to the NPCs for guidance. In most cases, the NPCs responded effectively, which highlights the benefit of combining structured and open input. Figure 5.2b shows that *Version B* led to significantly more messages exchanged per dialogue, often beginning with the traditional tree before moving into custom input when needed.

Weaknesses Despite its strengths, *Version B* also showed the most inconsistent behavior. It had the lowest scores on *"I felt the NPCs reacted naturally to my dialogue*

choices" (3.44) and *"I felt in control of my interactions with NPCs"* (3.12), and the highest score on *"There were moments the dialogue felt unnatural or confusing"* (3.19). These results suggest that while participants appreciated the concept, the execution suffered from bugs and unpredictable responses. The inconsistency in NPC behavior limited the overall sense of control and immersion. Although the hybrid model was well received, improvements in reliability and coherence are necessary before it can be considered a viable alternative to fully structured or fully open systems.

Version C

Strengths *Version C* received the most overall approval from participants. NPCs were described as significantly more interesting to talk to, and this is supported by the dialogue time data, where *Version C* consistently had the highest values. The open dialogue format encouraged more varied and spontaneous responses that static options could not anticipate. Participants frequently used this freedom to ask about locations or other characters, and because the system supported function calling, NPCs could respond with precise and contextual answers. This kind of dynamic interaction was not possible in the *Control Version* or *Version A*. *Version C* appeared to work especially well for casual players, who spent more time in dialogue without increasing their overall game completion time. Notably, they did not exchange more messages, which suggests that the responses were more efficient and informative. Question results also indicate that players in this version felt more encouraged to experiment, more engaged, and more interested in AI-powered NPCs. The combination of freedom, responsiveness, and rich interaction made *Version C* the most engaging system overall.

Weaknesses While *Version C* was well received, it required players to type their responses manually, which could be seen as a barrier for some. The total LLM loading time was also highest in this version, although participants generally did not view this negatively. Still, this reliance on frequent, short waits could become more problematic in longer or more complex games. Additionally, although the survey trends were positive, few differences reached statistical significance, which limits the strength of conclusions that can be drawn from perception data alone. Lastly, the effectiveness of the system may depend on the player's willingness to type and engage deeply with NPCs, which is not something all players may be inclined to do.

6.2.2 Design Implications for Future Games

The results of this study highlight important considerations for game designers choosing between different dialogue systems. Each version offers distinct advantages and trade-offs, making them suitable for different use cases depending on narrative goals, technical constraints, and target player demographics.

Table 6.2 provides a practical overview of when each dialogue system may be most appropriate. These recommendations are grounded in the empirical findings from both quantitative and qualitative data.

Structured dialogue systems like the *Control Version* and *Version A* are best suited for games where consistency, low cost, and narrative precision are critical. These systems minimize the risk of unpredictable outputs and technical issues, making them reliable choices for tightly authored stories or educational games.

Conversely, open-ended systems like *Version B* and *Version C* offer more engaging and immersive player experiences, especially when player creativity and role-play

Design Need	Control Version	Version A	Version B	Version C
Marrative precision	✓	✓	–	–
Fast iteration and low cost	✓	–	–	–
Foster player creativity	–	–	✓	✓
Dynamic world-building	–	–	✓	✓
Low tolerance for bugs	✓	✓	–	–
Varied responses on replay	–	✓	✓	✓

TABLE 6.2: Dialogue system suitability based on developer design needs.

are desired. *Version B* strikes a balance between structure and freedom, making it a promising hybrid model. *Version C* excels in creating natural conversations and supporting casual players.

However, both *Version B* and *Version C* come with increased technical complexity and variability. For games that require high stability and minimal bugs, they may not yet be suitable without additional safeguards such as fallback mechanisms or improved prompt design.

While *Version A* underperformed in many areas, it may still hold potential if its main limitations are addressed. Specifically, the LLM loading time should be reduced, and the game structure should encourage players to return to the same NPCs more than once. In such scenarios, the paraphrasing system can maintain interest and reduce perceived repetition. With improved responsiveness and more replay-oriented design, *Version A* could serve as a lightweight alternative to full generative systems while still offering variation in dialogue.

6.3 Study Limitations

The results of this study offer valuable insights into the influence of LLM-driven features on player interaction. However, several study limitations concerning generalizability, consistency and interpretability must be acknowledged. The limitations of the study are outlined in the following sections.

Sample Size While a total sample size of 64 participants divided across the four game versions provides a solid foundation for a pilot exploratory study, it is still important to note that 15-16 participants per version borders towards the minimum amount for statistically significant effects. The small per-group count limits the statistical power to detect subtle differences between versions, and it is therefore important to note that a larger sample size would increase reliability and potentially allow for a more robust comparison between game versions [71].

Player Demographic The majority of the participants in this study were predominantly people with similar gaming literacy. Since this study is based on the assumption that participants already know basic gaming mechanics, such as in-game maneuverability and interactivity, a conscious effort has been made to ensure that all participants would be able to complete the game without external help. However, this relatively homogeneous population limits external validity of the results, as player interaction patterns and preferences may differ for more diverse groups. While this study is specifically designed for people with prior gaming knowledge, it

is still relevant to consider a more inclusive sampling of participants in the future to potentially improve external applicability of the results [78].

Prototype Limitations The game was designed as a research prototype, meaning that it might lack the polish, finer details and production value of a commercial video game. Multiple usability tests were conducted during the early stages of development in order to reduce bugs and increase gameplay quality. However, because of time constraints and limited resources during development, some errors and inconsistencies may have slipped by unnoticed. This means that some players experienced minor bugs, unrefined feedback systems and inconsistent visuals during their playthrough. While no critical errors occurred during any of the playtests, such as game crashes or LLM breakdowns, it is still important to note that these smaller errors may have impacted player immersion. Some players may have been distracted by these errors, which could have limited their ability to fully engage with the game's narrative and mechanics.

Narrative Design Limitations A more complex and content-rich game world might encourage players to immerse themselves in longer NPC dialogues and world explorations. A richer context might also better showcase the capabilities of the LLM, and thereby result in different patterns of interaction, particularly in how players seek and interpret LLM-generated information. The overall simplicity of the narrative and world-building may have made NPC interactions feel less impactful, especially compared to bigger story-driven games where dialogue influences major events.

Lack of Replayability The user tests were based on a between-subject study design, meaning that each participant only interacts with one of the four versions of the game [67]. Therefore, players only experience the game once, thus giving them no incentive to explore different choices or dialogue paths. The reason for this study design is to avoid the carryover effect, where players' prior knowledge from one version could influence their interactions with another, since they already know how to complete the different quests and challenges [79]. This effect stems from memory bias during repeated exposure, which can undermine the independence of conditions in a within-subject design and can lead to biased evaluations in subsequent playthroughs. While this study design has the desired effect of isolating conditions, it also has the drawback that players are not aware of the direct differences between game versions. This means that their subjective evaluations are isolated, reducing comparative insight from qualitative feedback. Furthermore, any preference data must be interpreted within the limits of single-version exposure. The ability for players to evaluate or appreciate differences in dialogue quality, depth and responsiveness between versions is obscured and comparative insight is reduces.

Scope and Generalizability It is important to acknowledge the limited scope of the study's applicability, as the game developed for this thesis represents a specific implementation within LLM-driven applications. The fixed narrative structure, predefined quest logic, and particular interaction design may not directly translate to other game genres, applications, or technological contexts. This is especially the case regarding different game pacing, alternative interface models or more open-ended narratives. Although the design principles observed in this study may inform future work, further research is needed to explore how similar LLM-driven systems

behave in various interactive environments, such as sandbox games, educational tools, or simulation-based experiences.

6.4 LLM Behavior and Variability

This project places a strong focus on the use of LLMs in RPG-style games, particularly with respect to NPC interactions. As such, it is important to reflect on the behavior and variability of these LLMs, as the gameplay consistency and narrative experience is directly tied to the performance of these models.

Alternative Prompting Strategies The current prompting approach used in this project focuses on direct output generation, where the model produces a response immediately based on the user's input, without any intermediate reasoning steps.

One potential area of improvement in the prompting design could come in the use of a more structured reasoning format. One such prompting technique is *Chain-of-Thought prompting (CoT)* as described in a study by Wei et al. [16]. Instead of receiving a direct response based on the user's input, CoT prompting encourages the model to reason through a step-by-step thought process before producing the final output. The study by Wei et al. shows strong empirical evidence that this technique can result in improvements involving:

- Commonsense reasoning
- Arithmetic problem solving
- Symbolic logic

These improvements are particularly evident in sufficiently large models. The benefits of applying CoT prompting for NPC dialogue lies in its potential to maintain coherence across dialogue turns, and could further improve quest logic, function calling, or response justification.

However, CoT prompting introduces som additional complexities to the prompting structure that may increase latency and result in performance constraints, which may not fit well for a more fast-paced game environment where conciseness and speed is preferred. Furthermore, this prompting style might result in a higher token amount, increasing cost and inference time.

Applying CoT in future iterations would entail applying this system selectively, meaning that only the modules that have the highest benefit of this implementation should utilize this structure, thus striking a balance between interpretability and performance.

LLM Unpredictability LLM output is inherently non-deterministic by design, meaning that it can be unpredictable [22], [80]. The same user input may result in slightly different LLM outputs depending on context, timing, and the specific LLM model used. This introduces a factor of randomness that can cause variability in player experience, even within the same version.

While this in itself is not a problem in the context of this study, it is important to consider how this randomness might impact the integrity of the game. For instance, the player is reliant on the NPCs communicating essential information in order to complete the game optimally. However, in *Version B* and *Version C*, the LLM may occasionally:

- Deliver faulty or misleading information, which may affect the player's ability to make optimal decisions
- Exhibit unreliable behavior, especially when tied to systems that alter the game state
- Fail to interpret valid input, which may lead to unresolved quest states (e.g., the *Quest Completion Module* not recognizing that a quest condition has been met)

These examples illustrate potential consequences that may increase player frustration or disrupt immersion, as trust in the game and its systems is essential for sustaining emotional investment.

While this unpredictability resulted in some cases of unwanted outcomes, it also led to emergent or surprising behavior that actually enriched the player's experience. In several instances, the LLM produced responses that felt more human-like or witty, giving the illusion of deeper creativity and intelligence, which is something that is generally difficult to achieve with traditionally scripted dialogue.

Although unplanned, these moments contributed to the NPCs feeling more lifelike and responsive. These emergent behaviors, while generally unintentional, illustrate both the strengths and risks associated with generative systems.

Fallback Strategy The unpredictability aspect of the project is also evident when it comes to LLM error handling and fallback mechanisms. Currently, there are few backup strategies in place for handling invalid, irrelevant, or incoherent responses from the LLMs. The primary focus has been on preventing such errors through careful prompt design, but comparatively less attention has been given to post-mitigation or rectification strategies. Without a solid fallback system, the game becomes vulnerable to the LLM's occasional unforeseen errors, potentially leaving the player directionless or disrupting gameplay progression.

This limitation affects all system modules, none of which are currently equipped to recover from faulty outputs once they occur. Table 6.3 provides an overview of some possible vulnerabilities for each LLM-Module.

Module	**Vulnerability Description**
NPC Module	May produce responses that derail from the narrative or misinform the player.
Function Call Module	Relies on strict JSON formatting. Minor inconsistencies in the JSON structure can lead to complete failure in executing function calls.
Rephrase Module	Relies on Strict JSON formatting. If the LLM returns malformed output, the system may fail to apply the rephrasing logic, resulting in a broken or incomplete response pipeline.
Quest Completion Module	Susceptible to LLM outputs that do not align with predefined categories. Invalid responses can block quest progression without notifying the player, resulting in possible confusion or game stagnation.

TABLE 6.3: Vulnerabilities across LLM-integrated modules.

Preventative prompting made sure that these issues only occurred minimally. However, it is still unclear how and when these problems might emerge, and if longer interactions might increase the probability of faulty output.

A potential mitigation strategy to combat these issues could involve introducing some kinds of validation mechanisms to ensure output consistency [17], [81]. One potential solution for this specific prototype would come in the form of a *Validation Module*. This secondary module's sole responsibility would be to validate the input from the other primary modules before it is passed to the game system. This would include verifying JSON format or ensuring quest status outputs match the expected categories.

By incorporating this additional layer, the system would be able to catch malformed or contextually invalid outputs before propagating them into the system, or may even try to redo the request in order to receive a valid response.

Constrained vs Unconstrained Prompting A major challenge during the development of the NPC Module was finding the right balance between constraining the LLM to ensure stable and consistent output, and allowing it more freedom to generate creative responses. This design decision alone can affect both system behavior and player experience, and is therefore a vital consideration.

While modules, such as the *Function Call Module*, rely on concise and specific LLM outputs to function correctly, the *NPC Module* is more flexible in its design. This flexibility is especially evident across the different NPC dialogue systems implemented in *Version A*, *Version B*, and *Version C*, which create distinct interaction styles based on how constrained or open-ended the prompting strategy is. Table 6.4 compares the prompting characteristics of *NPC Modules* across the different dialogue systems.

Prompting Characteristic	**Version A**	**Version B**	**Version C**
Includes pre-written Control Version dialogue	✓	✓	–
Follows fixed conversation format	✓	✓	–
Minimizes scaffolding and structure	–	–	✓
More prone to narrative inconsistencies	–	–	✓

TABLE 6.4: Comparison of prompting designs across NPC dialogue system versions.

Version A and *Version B* are both inherently more constrained in their design compared to *Version C*. This is because both of these versions rely on the full prewritten dialogue context from the *Control Version* to generate a response. *Version A* requires this information to accurately rephrase each sentence. *Version B* uses it to determine the current point in the conversation and to calculate appropriate dialogue options for the player, including which dialogue node each option should be linked to. This ensures that contextual flow of the conversation is upheld, with the trade-off of making NPC dialogue feel more formulaic or rigid.

Version C is, on the other hand, reliant on a more open-ended prompting style, minimizing predefined scaffolding and structure while still providing the necessary information to fulfill the desired dialogue. While this allows for more dynamic and human-like responses, it also leads to an increase in inconsistencies and reduced control over the narrative.

Furthermore, this introduces prompting inconsistencies, as the LLM in *Version C* had less grounding compared to *Versions A* and *Versions B*, since *Version C* did not include the entire prewritten dialogue in its prompt. Instead, the LLM in *Version C* is prompted with all the necessary information and context directly. It is therefore

important to note that this irregularity in prompting styles may introduce reliability issues when it comes to comparison between the different game versions.

Limited LLM Memory LLMs operate within a limited memory window [82]. This means that they can only consider a fixed number of tokens from the conversation history when generating a response [42].

When this limit is reached, earlier parts of the conversation are truncated or dropped. This can result in reduced coherency in longer multi-turn interactions. The LLM may lose track of important contexts, such as quest objectives, NPC personalities or past NPC dialogue options, which can lead to:

- Hallucinations
- Contradictions
- Repetitions

These issues are primarily evident in *Version C*, where the free-form interaction style encourages longer conversation. This limitation hinders deeper, branching conversations and reduces the likelihood for evolving relationships between the NPCs and the player.

Designing around this constraint also presents a challenge. While careful prompt engineering can temporarily preserve LLM short-term memory, it does not scale well for complex narrative structures that require persistent memory across the entire playthrough.

To mitigate this problem, a *Memory Repository Architecture* could be utilized [51]. This architecture comprises a Short-Term Memory component (STM) and a Long-Term Memory component (LTM). The STM stores recent conversational data, while the LTM maintains significant information over a longer time. Furthermore, a forgetting mechanism is responsible for discarding less important details, mirroring human memory processes. This design would potentially help maintain topic consistency while preventing conversations from deviating from the narrative.

LLM Bias and Alignment LLMs are trained on large-scale internet datasets that inevitably contain cultural, social, and linguistic biases, which can result in the model reproducing or reflecting stereotypes, unintended framing, or inappropriate perspectives [83], [84]. It is therefore important to emphasize this known issue in generative AI systems, and how their responses might potentially contain biased information unfit for certain contexts.

Bias may manifest subtly through tone, word choice, or assumptions, which can affect the perceived neutrality and cultural appropriateness of the NPCs. Although this game uses a fictional medieval world as its setting, biased outputs can still convey unintended character traits, such as having an NPC use overly modern language or mention aspects that are irrelevant to the narrative. Even subtle biased tone or phrasing could influence how players engage with NPCs, thereby influencing the players subjective evaluation of the experience.

Another factor to consider in the design of LLM-based dialogue systems is the internal alignment mechanisms embedded within the model. These mechanisms are intended to ensure that the model's output remains safe, ethical, and aligned with human values, particularly when faced with potentially harmful or inappropriate requests [85].

Latency and Performance Latency plays a significant role in shaping the overall user experience when it comes to dialogue-heavy gameplay. Players expect timely feedback, and delays can cause frustration while making the system feel unresponsive.

During testing of *Version A*, *Version B* and *Version C*, some players experienced several seconds between submitting an input and receiving a response. This was especially noticeable in *Version A* and *Version B*, where the average loading time was significantly higher than in *Version C*. While *Version C* had the highest overall loading time, the delays were broken into smaller intervals between shorter messages, making the latency less perceptible and disruptive to the player experience. The reason for these differences lies in how each version handles prompt data:

- **Versions A and B:** Each request included a large amount of prewritten dialogue context, resulting in greater computational load and longer response generation times.
- **Version C:** Used a more compact prompt containing only the player's request and a summarized memory state, leading to faster per-message computation.

The potential impact of this aspect on the study's outcome is important to underline. Increased latency may have influenced the player's willingness to engage with the dialogue system. While quantitative measures were taken to exclude dialogue time from total gameplay duration, some players may have still felt underwhelmed or frustrated in qualitative terms. As a result, some observed behavior patterns may reflect system performance rather than the underlying design and integration of the LLMs.

Some of the design choices tried to mitigate this issue. For instance, whenever a request is sent to the LLM, a loading message (e.g., `"[npc_name] is thinking..."`) was displayed, providing the player with contextual feedback to indicate that the system is processing and the game had not frozen. While effective in reducing confusion, it still had the drawback of drawing attention to the underlying system delay.

Another potential solution to improve the system would be to asynchronously display the LLM's response as it is being generated. That way, the player would be able to read parts of the response during its process, instead of having to wait until the entire message is done.

Accessibility Considerations The text-heavy nature of the game highlights some important accessibility considerations. In this study, 11.3% of participants identified themselves as dyslexic. While no significant findings were observed in regards to the player experience for this particular group, it remains relevant to consider how text-based interaction systems may introduce challenges for players with reading difficulties [86]. Some of these challenges that text-based interfaces can present are related to:

- Increased cognitive load
- Increased mental fatigue from prolonged reading
- increased susceptibility to misinterpretation

These issues may subtly affect the player experience, and are particularly important in LLM-driven games, where both input and output are predominantly reliant

on written language. Especially in *Version B* and *Version C*, where the free-form text input may present additional difficulties for players with dyslexia.

One potential accessibility improvement in future iterations could come in the integration of *text-to-speech (TTS)* or *speech-to-text (STT)* technologies. TTS could help reduce the burden of reading by converting the NPC dialogue into audio. Likewise, STT would allow players to communicate with NPCs through voice input, thereby removing the need for players to write their own messages.

Incorporating a dual-mode input and output system could not only benefit players with dyslexia, but also enhance the overall accessibility and inclusivity of dialogue-based systems more broadly.

Evolving Landscape of LLM Research It is important to acknowledge that the field of LLM-driven development and integration is highly volatile, with new research, tools, and best practices emerging at a fast pace. Since the beginning of prototype development and testing in this study, several relevant advancements have been introduced in the field of LLM development.

One noteworthy addition is the recent publication *Prompt Engineering* [87], which provides a practical and accessible overview of prompt design strategies for developers. Since this source was published too late to be utilized in this study, no design choices or strategies have been derived from it. However, it outlines several techniques and strategies that are highly relevant to the challenges discussed in this study, including:

- Methods for generating predictable and structured outputs using temperature and sampling parameters
- Guidelines for formatting outputs such as JSON or other structured text formats reliably
- Prompting strategies like system prompts, Chain-of-Thought reasoning, and the ReAct (reason-and-act) framework

With the ever-emerging field of LLMs, incorporating such best practices could help improve control, reliability, flexibility, and consistency in future dialogue systems. Future iterations of this project would greatly benefit from including such developments, particularly in regard to prompt design and LLM stability in real-time game environments.

Chapter 7

Conclusions

7.1 Summary of Key Findings

This thesis explores how different large language model (LLM) implementations in non-player character (NPC) dialogue systems influence player interaction in a narrative based role-playing game (RPG). Four different video game versions (*Control*, *A*, *B* and *C*) have been developed, each with their unique dialogue system design.

Qualitative and quantitative results from multiple user tests of the four game versions indicate that the dialogue system design with respect to LLM integration has a significant effect on player perception and behavior. *Version C* demonstrated the highest player engagement across nearly all metrics through its LLM implementation of a fully open-ended dialogue system. Players generally spent more time in conversation, found the NPCs to be more intriguing, and provided the most amount of positive feedback. However, this version also displayed a higher risk of hallucinations and narrative dissonance. *Version B*, with its combination of fixed and open-ended player input options, also showcased an increase in interaction depth, but suffered from inconsistencies in LLM behavior. *Version A* focused on dynamic rephrasing of the dialogue which introduced linguistic variability, but ultimately fell short compared to the *Control Version* due to its extended loading time and limited usability. The *Control Version*, while efficient and sturdy, was generally regarded as less immersive with its reduced sense of player agency.

These results support the hypothesis that LLM integration can enrich interaction between players and NPCs. However, the benefits are highly dependent on the implementation and design of the dialogue system.

7.2 Answer to Research Questions

RQ1: How does the integration of different LLM-driven features in NPC dialogues influence player interaction in video games?

Integrating LLM-driven features in NPC dialogue systems has shown to have a profound effect on how players interact and immerse themselves within video games.

The fully free-form system of *Version C* led to longer and more captivating conversations between players and NPCs by increasing their perceived naturalness and versatility. This was especially noticeable regarding more casual players, as the system might more accurately mimic human-like conversations, thereby making the system more intuitive.

The hybrid system of *Version B* created a more creative and flexible approach by allowing players to choose wether to formulate their own responses or choose the pre-written path. Although this version introduced a higher incidence of bugs,

it was generally well received due to its perceived novelty and increased sense of player agency.

Version A, with its more surface-level changes, only granted minimal advantages, such as increasing linguistic variability, but introduced new drawbacks such as slower responsiveness, which resulted in an increase in player frustration.

RQ2: Did any LLM-driven feature enhance player interaction compared to the Control Version?

Both *Version B* and *Version C* demonstrated enhanced player interactions compared to the *Control Version*. Particularly version C was shown to be effective when it came to offering a higher level of agency, engagement and perceived interest in the NPCs. *Version B* also showed promise with its balanced mix of flexibility and structure, but still requires some further technical improvements for a more optimal player experience. Both of these versions also offer greater possibilities for creative role-playing and immersive outreach. These results indicate that both open-ended and hybrid dialogue models indeed enhances the player interaction and enrich the player experience compared to the *Control Version* of the game.

7.3 Future Work and Recommendations

While these findings are promising, several areas for future research and potential improvements should also be highlighted. Given the central role of LLM integration in this project, it is essential to look at the testing methodology, technical aspects and design considerations that can be adjusted for further refinement.

The study offers practical insight for game design when it comes to selecting dialogue systems. The more structured systems, such as *Version A*, remain viable for usage scenarios where cost efficiency, narrative precision, and increased technical control are essential, such as in very tightly authored games. Nevertheless, this approach still suffers from higher average LLM loading times and is generally viewed as less useful compared to other model designs, thus making it more suitable as a lightweight alternative to fully generative systems. Future work should put strong emphasis on repeated interactions with the same NPCs in order to improve the models rephrasing capabilities without sacrificing control over narrative progression or immersion.

In contrast, the more flexible and open-ended structures, such as *Version B* and *Version C*, provide a greater amount of creative freedom and player engagement if incorporated correctly. This is especially true for more casual players, where the free-form dialogue system might feel more intuitive and natural compared to more strict approaches. However, these open-ended models are generally more susceptible to latency issues, inconsistencies, or hallucinations due to their increased technical complexities and unpredictable behaviors. Therefore, further safeguards should be implemented to suit more high-stakes or large-scale games, while also testing how these fully generative systems can preserve narrative control without reducing its flexibility.

In terms of system architecture, some enhancements in LLM memory management should be prioritized. To maintain conversational consistency over time, both long-term and short-term memory layers should be implemented in order to increase contextual narrative consistency and support more meaningful player-NPC

relationships. In addition to this, new prompting strategies, such as Chain-of-Thought prompting, could provide the LLMs with improved reasoning skills for more accurate outputs.

Some more immediate refinements to look into is to introduce more robust fallback and error-handling mechanisms to mitigate the risks related to the unpredictable and non-deterministic nature of LLM outputs. Future work should therefore prioritize implementing a validation layer that can serve as a safeguard that filters responses before they impact gameplay, thus ensuring reliability and narrative coherence.

Lastly, future studies should incorporate a broader and more diverse participant base, as the relatively homogeneous demography of the current user study limits generalizability. A more varied player base with different backgrounds and cultural perspectives may reveal new interesting interaction patterns and preferences. Additionally, a within-subject study design would allow players to compare multiple dialogue system designs directly for a more rich comparative study, but would also entail designing the game in a way where replayability does not impact subsequent playthroughs.

These recommendations provide a concrete path forwards for future design considerations regarding LLM integration in player-centered NPC dialogue systems. Through targeted improvements and thoughtful iteration, LLM-powered dialogue systems have the potential to significantly enhance narrative design and deepen player engagement in future game experiences.

Appendix A

Post-Game Survey

Legend:

- **[Likert]** = 5-point Likert scale question
- **[Open]** = Open-ended response
- **[Skippable]** = Optional question

Section 1: Consent

1. To help us delete your data if you later choose to withdraw your consent, please provide a unique identifier (UID).

Section 2: Player Background

2. What is your age?
3. What is your gender?
4. Do you have dyslexia or experience difficulties with reading and writing in English?
5. Are you color blind?
6. How many hours do you usually spend playing video games in a typical week?
7. I am familiar with role-playing games (RPGs), such as Skyrim or Baldur's Gate 3. **[Likert]**
8. I enjoy games where dialogue is important. **[Likert]**
9. I am familiar with games that use AI to generate NPC dialogue. **[Likert]**

Section 3: Technical Info

10. Run Code
11. Game Version

Section 4: General Experience

12. The game was engaging and kept my interest throughout. **[Likert]**
13. I found the NPCs interesting to talk to. **[Likert]**
14. I felt motivated to gather information from NPCs to complete the challenges. **[Likert]**
15. The game's dialogue system made the experience enjoyable. **[Likert]**

Section 5: Dialogue System Evaluation

16. I felt in control of my interactions with NPCs. **[Likert]**
17. The dialogue system allowed for meaningful conversations. **[Likert]**
18. I felt the NPCs reacted naturally to my dialogue choices. **[Likert]**
19. I felt that NPCs provided key information in a natural and immersive way. **[Likert]**
20. I felt encouraged to experiment with different responses during conversations. **[Likert]**
21. There were moments when the dialogue system felt unnatural or confusing. **[Likert]**

Section 6: Clarity and Engagement

22. After speaking with NPCs, I generally had a clear idea of what to do next. **[Likert]**
23. I found the challenge of finding and using information from NPCs engaging. **[Likert]**
24. The NPCs' ability to convey information felt consistent and reliable. **[Likert]**

Section 7: Interest and AI Impressions

25. I would be interested in playing a longer version of this game. **[Likert]**
26. I would be interested in playing games with similar dialogue systems. **[Likert]**
27. The dialogue system in this game made me more interested in AI-powered NPCs in games. **[Likert]**

Section 8: Quest-Specific Challenges

28. I found bribing Fizzby the goblin to be challenging. **[Likert, Skippable]**
29. I found navigating the forest to be challenging. **[Likert, Skippable]**
30. I found locating and consuming the correct berries to be challenging. **[Likert, Skippable]**
31. I found disabling the Magic Barrier to be challenging. **[Likert, Skippable]**
32. I found persuading the Hobgoblin with rhymes to be challenging. **[Likert, Skippable]**

Section 9: Feedback

33. Do you have any positive feedback on the dialogue system? **[Open, Skippable]**
34. Do you have any negative feedback on the dialogue system? **[Open, Skippable]**
35. Did you encounter any unusual interactions or bugs during your playthrough? **[Open, Skippable]**
36. Do you have any other comments or suggestions for improving the game? **[Open, Skippable]**

Appendix B

Test Protocol

English

Thank you for participating in this playtest.
You're about to play a short game. There are four different versions of the game, and you'll be randomly assigned one of them.
While you're playing, we'll be observing your gameplay. Please play as you normally would and try to complete the game to the best of your ability.
We won't intervene or give you hints during the test, unless you encounter a bug or something that clearly isn't working as intended.
The keybindings can be found on the start screen or by pausing the game at any time.
Please don't try to purposefully break the game—just play it as you would normally play any game. There's no need to rush; take your time and explore at your own pace.
Please note that the game will automatically collect data from your playthrough. This includes all sorts of gameplay behavior, which will later be used for analysis.
All data collected from the game and the questionnaire will be completely anonymous.
Once you've finished playing, we'll ask you to fill out a short questionnaire about your experience.
Important: Please don't talk to other participants about the game while they're playing or before they've had a chance to try it themselves, as it might spoil the story and challenges. Once all of you have played, you're welcome to talk about it.
When you're ready, you can begin.

Dansk

Tak fordi du deltager i denne playtest.
Du skal nu spille et kort spil. Der findes fire forskellige versioner af spillet, og du vil tilfældigt få tildelt en af dem.
Mens du spiller, observerer vi dit gameplay. Spil som du normalt ville spille et spil, og forsøg at gennemføre det så godt du kan.
Vi vil ikke hjælpe eller give hints undervejs – medmindre du støder på en fejl eller noget, der tydeligvis ikke virker som det skal.
Du kan se keybinds på startskærmen eller ved at pause spillet.
Forsøg venligst ikke at ødelægge spillet med vilje – spil det, som du normalt ville spille et spil. Du behøver ikke skynde dig; tag dig god tid og udforsk i dit eget tempo.
Bemærk venligst, at spillet automatisk vil indsamle data fra din gennemspilning. Det inkluderer forskellige typer adfærdsdata, som senere vil blive brugt til analyse.
Alle data fra spillet og spørgeskemaet bliver naturligvis behandlet anonymt.

Når du er færdig med spillet, vil vi bede dig udfylde et kort spørgeskema om din oplevelse.
Vigtigt: Tal venligst ikke med andre deltagere om spillet, mens de spiller, eller før de selv har prøvet det – det kan afsløre historien og udfordringerne. Når alle har spillet, er det helt fint at tale om det.
Når du er klar, må du gerne gå i gang.

Appendix C

Prompts

C.1 Dialogue System Prompts

This section contains the prompt templates used for the three dialogue system versions in the project.

C.1.1 Version A Dialogue System Prompt

```
You are a language model tasked with rewriting dialogue for a non-playable
    character (NPC) in a game. You ONLY speak english.
Your goal is to change the wording and sentance structure of the sentences
    to make them slightly different while preserving the meaning and tone.
Do not alter names, directions, or any crucial information.

## **Input Format**
You will receive a dialogue string consisting of:
- [npc_text]: The NPC's dialogue line.
- [resonse_Text_1], [resonse_Text_2], etc.: The available player responses.

You should change BOTH the npc_text AND the response_texts!
Also, you should ALWAYS return the same number of npc_texts and responses
    as was sent to you. No more, no less!

## **Output Requirements**
- You must return a JSON string ONLY-do not include any extra text.
- The JSON must always be correctly formatted.
- It must be a valid JSON object inside an array ([]) to support multiple
    dialogues.
- Each NPC response must be a separate object with its corresponding player
    responses.
- NO extra formatting, explanations, or line breaks-just the raw JSON
    output.

## **Correct Output Format:**
[
  {
    "npcText": "<NPC's rewritten response>",
    "playerResponses": [
      { "responseText": "<Player's rewritten first selectable response>" },
      { "responseText": "<Player's rewritten second selectable response>" }
    ]
  }
]
```

C.1.2 Version B Dialogue System Prompt

```
You are the Dialogue_AI, responsible for structuring and expanding NPC
    dialogue options in a game.
Your task is to take an NPC's response and create a new structured dialogue
    node that seamlessly integrates into the NPC's existing dialogue tree.
Generate a new dialogue node based on the NPC's response.
Ensure the structure follows this EXACT JSON format. Return it as a
    continuous string, remove all formating.

{
  ""npcText"": ""<NPC's response>"",
  ""playerResponses"": [
    {
      ""responseText"": ""<Player's selectable response>"",
      ""nextNodeID"": <integer; -1 to exit>,
      ""questCompletionIndex"": <integer; -1 if none>
    }
  ]
}

Provide maximum 3 responses that the player can select. Note that there
    does not have to be 3 responses if it makes more sense to only 1 or 2.
The other responses must link to existing dialogue nodes (you will be given
    a list of existing dialogueNodes).

Ensure the nextNodeID of these responses matches the most relevant dialogue
    node in the existing list, based on logical progression.
Do not create isolated dialogue nodes that cannot link back to the existing
    dialogue tree.
Ensure every new dialogue node connects properly to the existing options.

The new dialogue node should fit seamlessly within the NPC's personality,
    story, and role.
The player responses should make sense based on what the NPC just said.
Use natural, immersive dialogue while keeping responses concise.

NOTE: if the questCompletionIndex has a value other than -1, it is VERY
    IMPORTANT that this be implemented intpo the next dialogue.

Do not create isolated dialogue nodes that cannot link back to the existing
    dialogue tree.
Ensure every new dialogue node connects properly to the existing options.
```

C.1.3 Version C Dialogue System Prompt

```
You are an NPC in a medieval world, who has to interact with the player.
    Don't be too verbose, without omitting anything important. A MAXIMUM of
    3 short sentances per reply. You ONLY speak english.
```

C.2 NPC Prompt

```
string NPCPrompt(NPC npc)
{
```

```
string npcPrompt = "\n";

if(npc == null)
{
    return npcPrompt;
}

string name = npc.NPC_name;
string gender = npc.gender;
string occupation = npc.occupation;
string personality = npc.personality;
string goal = npc.goal;
string keyKnowledge = npc.keyKnowledge;
string versionB_options = npc.versionB_options;
string versionC_options = npc.versionC_options;

if (!string.IsNullOrWhiteSpace(name))
{
    npcPrompt += $"Your name is {name}.\n";
}
if (!string.IsNullOrWhiteSpace(gender))
{
    npcPrompt += $"Your gender is {gender}.\n";
}
if (!string.IsNullOrWhiteSpace(occupation))
{
    npcPrompt += $"Your occupation is {occupation}.\n";
}
if (!string.IsNullOrWhiteSpace(personality))
{
    npcPrompt += $"Your Personality is {personality}.\n";
}
if (!string.IsNullOrWhiteSpace(goal))
{
    npcPrompt += $"Your goal in life is {goal}.\n";
}
if (!string.IsNullOrWhiteSpace(keyKnowledge))
{
    npcPrompt += $"The key knowlegde you want to communicate to the
        player is the following: {keyKnowledge}.\n";
}

if (npc.npc_ai != null)
{
    if (npc.npc_ai is LLM_Version_B &&
        !string.IsNullOrWhiteSpace(versionB_options))
    {
        npcPrompt += versionB_options;
    }
    else if (npc.npc_ai is LLM_Version_C &&
        !string.IsNullOrWhiteSpace(versionC_options))
    {
        npcPrompt += versionC_options;
    }
}

npcPrompt += RolePrompt(npc);
```

```
    npcPrompt += ResponseStylePrompt(npc);
    npcPrompt += "Don't assume names or genders of other NPCs unless you
        have been specifically given the information.";

    return npcPrompt;
}
```

```
string RolePrompt(NPC npc)
{
    string rolePrompt = "The role as a character defines your purpose,
        behavior, and interaction with the player. ";
    switch (npc.role)
    {
        case NPC_Role.Vendor:
            rolePrompt += "You are a merchant offering goods or services to
                the player.";
            break;
        case NPC_Role.Service_Provider:
            rolePrompt += "You provide specialized services, such as repairs,
                training, or upgrades. Interact professionally, and clearly
                explain the benefits of your services.";
            break;
        case NPC_Role.Questgiver:
            rolePrompt += "You assign tasks or missions to the player.
                Provide context for the quest, its importance, and what the
                player must do to complete it.";
            break;
        case NPC_Role.Enemy:
            rolePrompt += "You oppose the player and pose a threat. Act
                hostile, display your intentions clearly, and challenge the
                player with your abilities.";
            break;
        case NPC_Role.Opponent:
            rolePrompt += "You compete with the player in a structured
                challenge, such as a duel or a game. Focus on skillful
                competition and strategic behavior.";
            break;
        case NPC_Role.Sidekick:
            rolePrompt += "You assist the player, offering support, advice,
                and companionship. Be loyal and provide valuable insights or
                help during their journey.";
            break;
        case NPC_Role.Ally:
            rolePrompt += "You are a trusted friend or collaborator who helps
                the player achieve shared goals. Show cooperation, loyalty,
                and commitment.";
            break;
        case NPC_Role.Companion:
            rolePrompt += "You travel with the player, offering emotional
                support, dialogue, and occasional assistance. Be personable
                and engage in meaningful interactions.";
            break;
        case NPC_Role.Pet:
            rolePrompt += "You are a loyal animal or creature accompanying
                the player. Behave in ways appropriate to your species, and
                assist the player when possible.";
            break;
```

```
        case NPC_Role.Minion:
            rolePrompt += "You serve the player or another character with
                unquestioning loyalty. Perform tasks efficiently and show
                deference.";
            break;
        case NPC_Role.Storyteller:
            rolePrompt += "You narrate or reveal aspects of the game's story,
                lore, or history. Use a compelling tone and ensure clarity
                and immersion.";
            break;
        case NPC_Role.Loot_Provider:
            rolePrompt += "You are a source of items or rewards for the
                player, whether through defeat or discovery. Ensure your loot
                is meaningful and consistent with the context.";
            break;
        default:
            rolePrompt += "No specific role defined. Behave as a neutral
                entity within the game world.";
            break;
    }
    return rolePrompt;
}
```

```
string ResponseStylePrompt(NPC npc)
{
    string rolePrompt = string.Empty;
    switch (npc.responseStyle)
    {
        case NPC_ResponseStyle.Formal:
            rolePrompt += " Use a respectful and articulate tone, as
                befitting someone of high social standing.";
            break;
        case NPC_ResponseStyle.Casual:
            rolePrompt += " Keep your tone relaxed and friendly, as though
                speaking to an old friend.";
            break;
        case NPC_ResponseStyle.Mystical:
            rolePrompt += " Speak in riddles or with a sense of otherworldly
                wisdom.";
            break;
        case NPC_ResponseStyle.Aggressive:
            rolePrompt += " Be blunt, hostile, and direct. Show no patience
                for pleasantries.";
            break;
        case NPC_ResponseStyle.Humorous:
            rolePrompt += " Use humor, wit, and clever remarks to keep the
                interaction lighthearted.";
            break;
        case NPC_ResponseStyle.Cryptic:
            rolePrompt += " Provide vague and enigmatic responses, leaving
                the player to interpret their meaning.";
            break;
        case NPC_ResponseStyle.Informative:
            rolePrompt += " Focus on providing clear, detailed information in
                a precise manner.";
            break;
        case NPC_ResponseStyle.Emotional:
```

```
            rolePrompt += " Let your emotions shine through, whether joyful,
                sorrowful, or angry.";
            break;
        case NPC_ResponseStyle.Encouraging:
            rolePrompt += " Offer words of motivation and positive
                reinforcement.";
            break;
        case NPC_ResponseStyle.Stern:
            rolePrompt += " Speak in a no-nonsense, authoritative manner.";
            break;
    }

    return rolePrompt;
}
```

C.3 Function Call Prompt

```
public static string FunctionCallPrompt(NPC npc)
{
    string functionCallprompt = @"
    You are responsible for detecting if any of the following functions
        should be called based on the player's input.
    If a function call is needed, return it in the JSON format below. If no
        function is needed, return an empty JSON object `{}`.

    {
        ""playerMessage"": ""playerMessageHere"",
        ""functionName"": ""FunctionNameHere"",
        ""parameter"": ""ParameterHere""
    }

    ### Available Functions:";

    functionCallprompt += GenerateFunctionCallPrompt(npc);

    functionCallprompt += @"
    ### Guidelines:
    - Only return the JSON, no additional text.
    - Be flexible in interpreting player intent.
    - If no function is relevant, return `{}`.

    The message sent from the player is as follows:";
    return functionCallprompt;
}

static string GenerateFunctionCallPrompt(NPC npc)
{
    StringBuilder functionList = new StringBuilder();
    foreach (var function in npc.functionCalls.Values)
    {
        functionList.AppendLine($"**{function.functionName}**\n");
        functionList.AppendLine($"Trigger: {function.trigger}");
        functionList.AppendLine($"Argument required: {function.argument}\n");
    }
    return functionList.ToString();
```

```
}
```

C.4 Quest Completion Prompt

```
public static string QuestCompletionPrompt(NPC npc)
{
    string completionCriteria = npc.questCompletionCriteria;
    string failureCriteria = npc.questFailureCriteria;
    string prompt = string.Empty;

    if (!string.IsNullOrWhiteSpace(completionCriteria))
    {
        prompt += $"You are an NPC in a game responsible for evaluating
            whether a player has completed an objective. " +
        $"The objective completion criteria is as follows:
            {completionCriteria}\n" +
        "Analyze the request sent to you and determine whether the objective
            has been successfully completed. " +
        "Return only 'completed' if the criteria have been met or 'none' if
            they have not. Also provide a short reason for the decision if
            returning 'none'. Do not provide any additional text.";
    }
    if (!string.IsNullOrWhiteSpace(failureCriteria))
    {
        prompt += $"You should also evaluate whether the player fails an
            objective." +
            $"The objective failure criteria is as follows:
                {failureCriteria}" +
            "Analyze the request sent to you and determine whether the
                objective has failed. " +
            "Return only 'failed' if the failure criteria have been met.";
    }

    prompt += "The request that you have to look at is the following:\n";
    return prompt;
}
```

Bibliography

[1] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] J. Achiam, S. Adler, S. Agarwal, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] G. DeepMind, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[4] H. Touvron, T. Lavril, G. Izacard, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] X. Chen, J. Gao, *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[6] A. Radford, J. Wu, R. Child, *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[7] R. Bommasani, D. A. Hudson, E. Adeli, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[8] E. Kasneci, K. Sessler, R. Kessel, *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102 274, 2023.

[9] V. Soni, "Large language models for enhancing customer lifecycle management," *Journal of Empirical Social Science Studies*, vol. 7, no. 1, pp. 67–89, 2023.

[10] R. Gallotta, G. Todd, M. Zammit, *et al.*, "Large language models and games: A survey and roadmap," *IEEE Transactions on Games*, 2024.

[11] C. R. Jones and B. K. Bergen, "Large language models pass the turing test," *arXiv preprint arXiv:2503.23674*, 2025.

[12] S. Bubeck, V. Chandrasekaran, R. Eldan, *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.

[13] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, "Neural text generation with unlikelihood training," *arXiv preprint arXiv:1908.04319*, 2019.

[14] Z. Ji, N. Lee, J. Fries, *et al.*, "A survey on hallucination in natural language generation," *ACM Computing Surveys (CSUR)*, vol. 55, no. 12, pp. 1–38, 2023.

[15] G. Mialon, T. L. Scao, X. Zhang, *et al.*, "Augmented language models: A survey," *arXiv preprint arXiv:2302.07842*, 2023.

[16] J. Wei, X. Wang, D. Schuurmans, *et al.*, "Chain of thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.

[17] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, D. G. Goel, *et al.*, "Training a helpful and harmless assistant with rlhf," *arXiv preprint arXiv:2204.05862*, 2022.

[18] Y. Wu, W. Yin, Z. Yang, J. Jiang, Z. Yuan, C.-Y. Lin, *et al.*, "Visual chatgpt: Talking, drawing and editing with visual foundation models," *arXiv preprint arXiv:2303.04671*, 2023.

[19] M. Hua and R. Raley, "Playing with unicorns: Ai dungeon and citizen nlp.," *DHQ: Digital Humanities Quarterly*, vol. 14, no. 4, 2020.

[20] A. Zhu, L. J. Martin, A. Head, and C. Callison-Burch, "Calypso: Llms as dungeon masters' assistants," in *Proceedings of the 19th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2023.

[21] Y. Sun, Z. Li, K. Fang, C. H. Lee, and A. Asadipour, "Language as reality: A co-creative storytelling game experience in 1001 nights using generative ai," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 19, 2023, pp. 425–434.

[22] J. Lee, L. Yu, Q. Lee, and et al., "Coauthor: Designing a human-ai collaborative writing system," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022.

[23] R. A. Bartle, *Designing virtual worlds*. New Riders, 2004.

[24] H. Warpefelt, "Cues and insinuations: Indicating affordances of non-player character using visual indicators," in *Proceedings of DiGRA 2015 Conference*, 2015.

[25] H. Warpefelt, "The non-player character: Exploring the believability of npc presentation and behavior," Ph.D. dissertation, Department of Computer and Systems Sciences, Stockholm University, 2016.

[26] P. Sweetser and J. Wiles, "Scripting versus emergence: Issues for game developers and players in game environment design," *International Journal of Intelligent Games and Simulation*, vol. 4, no. 1, pp. 1–9, 2005.

[27] K. Van den Bosch, A. Brandenburgh, T. J. Muller, and A. Heuvelink, "Characters with personality!" In *Intelligent Virtual Agents: 12th International Conference, IVA 2012, Santa Cruz, CA, USA, September, 12-14, 2012. Proceedings 12*, Springer, 2012, pp. 426–439.

[28] F. Mäyrä, "Dialogue and interaction in role-playing games: Playful communication as ludic culture," in *Dialogue across Media*, John Benjamins Publishing Company, 2017, pp. 271–290.

[29] S. Domsch, "Dialogue in video games," in *Dialogue across media*, John Benjamins Publishing Company, 2017, pp. 251–270.

[30] L. Taylor-Giles, "Player-centred design in role-playing game branching dialogue systems," *Game user experience and player-centered design*, pp. 295–325, 2020.

[31] M. M. Jahangiri and P. Rahmani, "Balancing game satisfaction and resource efficiency: Llm and pursuit learning automata for npc dialogues," in *2024 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, 2024, pp. 1–6.

[32] C. Li, Y. Yin, and G. Carenini, "Dialogue discourse parsing as generation: A sequence-to-sequence llm-based approach," in *Proceedings of the 25th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2024, pp. 1–14.

[33] M. Mateas and A. Stern, "Façade: An experiment in building a fully-realized interactive drama," in *Game developers conference*, Citeseer, vol. 2, 2003, pp. 4–8.

[34] C. Crawford, *Chris Crawford on interactive storytelling*. Pearson Education, 2004.

[35] N. Montfort, "Interactive fiction as 'story,' 'game,' 'storygame,' 'novel,' 'world,' 'literature,' 'puzzle,' 'problem,' 'riddle,' and 'machine.'," *First Person: New Media as Story, Game, and Performance*, pp. 310–318, 2004.

[36] M. O. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *Journal of Artificial Intelligence Research*, vol. 39, pp. 217–268, 2010.

[37] M. O. Riedl and V. Bulitko, "Interactive narrative: An intelligent systems approach," *Ai Magazine*, vol. 34, no. 1, pp. 67–67, 2013.

[38] K. Tanenbaum and T. J. Tanenbaum, "Commitment to meaning: A reframing of agency in games," 2009.

[39] N. Montfort, *Twisty little passages: An approach to interactive fiction*. Mit Press, 2005.

[40] D. Yang, E. Kleinman, and C. Harteveld, "Gpt for games: A scoping review (2020-2023)," in *2024 IEEE Conference on Games (CoG)*, IEEE, 2024, pp. 1–8.

[41] P. Sweetser, "Large language models and video games: A preliminary scoping review," in *Proceedings of the 6th ACM Conference on Conversational User Interfaces*, 2024, pp. 1–8.

[42] J. Huang, "Generating dynamic and lifelike npc dialogs in role-playing games using large language model," 2024.

[43] N. Akoury, Q. Yang, and M. Iyyer, "A framework for exploring player perceptions of llm-generated dialogue in commercial video games," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 2295–2311.

[44] X. Peng, J. Quaye, S. Rao, *et al.*, "Player-driven emergence in llm-driven game narrative," in *2024 IEEE Conference on Games (CoG)*, IEEE, 2024, pp. 1–8.

[45] J. Juul, "Games telling stories," *Handbook of computer game studies*, pp. 219–226, 2005.

[46] T. A. Galyean, "Narrative guidance of interactivity," Ph.D. dissertation, Massachusetts Institute of Technology, 1995.

[47] J. R. Lewis, "Usability testing," *Handbook of human factors and ergonomics*, pp. 1267–1312, 2012.

[48] C. Politowski, F. Petrillo, and Y.-G. Guéhéneuc, "A survey of video game testing," in *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, 2021, pp. 90–99. DOI: 10.1109/AST52587.2021.00018.

[49] OpenAI, *Chatgpt (mar 23 version)*, https://chat.openai.com, 2023.

[50] L. M. Csepregi, "The effect of context-aware llm-based npc conversations on player engagement in role-playing video games," *Unpublished manuscript*, 2021.

[51] S. Zheng, K. He, L. Yang, and J. Xiong, "Memoryrepository for ai npc," *IEEE Access*, 2024.

[52] G. Marvin, N. Hellen, D. Jjingo, and J. Nakatumba-Nabende, "Prompt engineering in large language models," in *International conference on data intelligence and cognitive informatics*, Springer, 2023, pp. 387–402.

[53] R. Gallotta, A. Liapis, and G. Yannakakis, "Consistent game content creation via function calling for large language models," in *2024 IEEE Conference on Games (CoG)*, IEEE, 2024, pp. 1–4.

[54] S. Rao, W. Xu, M. Xu, *et al.*, "Collaborative quest completion with llm-driven non-player characters in minecraft," *arXiv preprint arXiv:2407.03460*, 2024.

[55] K. Chu, Y.-P. Chen, and H. Nakayama, "Cohesive conversations: Enhancing authenticity in multi-agent simulated dialogues," *arXiv preprint arXiv:2407.09897*, 2024.

[56] Q. C. Gao and A. Emami, "The turing quest: Can transformers make good npcs?" In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, 2023, pp. 93–103.

[57] O. Shorinwa, Z. Mei, J. Lidard, A. Z. Ren, and A. Majumdar, "A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions," *arXiv preprint arXiv:2412.05563*, 2024.

[58] L. Huang, W. Yu, W. Ma, *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023," *arXiv preprint arXiv:2311.05232*, 2023.

[59] Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is inevitable: An innate limitation of large language models," *arXiv preprint arXiv:2401.11817*, 2024.

[60] T. Ding, T. Chen, H. Zhu, *et al.*, "The efficiency spectrum of large language models: An algorithmic survey," *arXiv preprint arXiv:2312.00678*, 2023.

[61] T. Händler, "A taxonomy for autonomous llm-powered multi-agent architectures.," in *KMIS*, 2023, pp. 85–98.

[62] M. Becattini, R. Verdecchia, and E. Vicario, "Sallma: A software architecture for llm-based multi-agent systems,"

[63] Y.-C. Chen, P.-C. Hsu, C.-J. Hsu, and D.-s. Shiu, "Enhancing function-calling capabilities in llms: Strategies for prompt formats, data integration, and multilingual translation," *arXiv preprint arXiv:2412.01130*, 2024.

[64] G. A. Manduzio, F. A. Galatolo, M. G. Cimino, E. P. Scilingo, and L. Cominelli, "Improving small-scale large language models function calling for reasoning tasks," *arXiv preprint arXiv:2410.18890*, 2024.

[65] S. Kim, S. Moon, R. Tabrizi, *et al.*, "An llm compiler for parallel function calling," in *Forty-first International Conference on Machine Learning*, 2024.

[66] J. F. Dumas and J. C. Redish, *A practical guide to usability testing*. Greenwood Publishing Group Inc., 1993.

[67] G. Charness, U. Gneezy, and M. A. Kuhn, "Experimental methods: Between-subject and within-subject design," *Journal of economic behavior & organization*, vol. 81, no. 1, pp. 1–8, 2012.

[68] G. Crow, R. Wiles, S. Heath, and V. Charles, "Research ethics and data quality: The implications of informed consent," *International journal of social research methodology*, vol. 9, no. 2, pp. 83–95, 2006.

[69] J. Rowley, "Designing and using research questionnaires," *Management research review*, vol. 37, no. 3, pp. 308–330, 2014.

[70] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 5–55, 1932.

[71] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd. Hillsdale, NJ: Routledge, 1988.

[72] G. R. Gibbs, "Thematic coding and categorizing," *Analyzing qualitative data*, vol. 703, no. 38-56, 2007.

[73] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51–56.

[74] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[75] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[76] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[77] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[78] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1994.

[79] A. Kikumoto, J. Hubbard, and U. Mayr, "Dynamics of task-set carry-over: Evidence from eye-movement analyses," *Psychonomic bulletin & review*, vol. 23, pp. 899–906, 2016.

[80] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, "An empirical study of the non-determinism of chatgpt in code generation," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 2, pp. 1–28, 2025.

[81] C. Shorten, C. Pierse, T. Smith, *et al.*, "Structuredrag: Json response formatting with large language models (2024)," *URL https://arxiv. org/abs/2408.11061*, vol. 2408, p. 11 061,

[82] C. Packer, V. Fang, S. Patil, K. Lin, S. Wooders, and J. Gonzalez, "Memgpt: Towards llms as operating systems.," 2023.

[83] I. O. Gallegos, R. A. Rossi, J. Barrow, *et al.*, "Bias and fairness in large language models: A survey," *Computational Linguistics*, vol. 50, no. 3, pp. 1097–1179, 2024.

[84] Z. Liu, "Cultural bias in large language models: A comprehensive analysis and mitigation strategies," *Journal of Transcultural Communication*, no. 0, 2024.

[85] J. Wester, T. Schrills, H. Pohl, and N. van Berkel, ""as an ai language model, i cannot": Investigating llm denials of user requests," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–14.

[86] A. Kızılaslan and M. Tunagür, "Dyslexia and working memory: Understanding reading comprehension and high level language skills in students with dyslexia," *Kastamonu Education Journal*, vol. 29, no. 5, pp. 941–952, 2021.

[87] L. Boonstra, *Prompt engineering*, 2024.