GENERATING DYNAMIC AND LIFELIKE NPC DIALOGS IN ROLE-PLAYING GAMES USING LARGE LANGUAGE MODEL

Lappeenranta–Lahti University of Technology LUT

Bachelor’s programme in Software and Systems Engineering DD, Bachelor’s thesis

2024

Junyang Huang

Examiner: Assistant Professor Dominik Siemon

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Sciences

Software Engineering

In co-operation with partner university: Hebei University of Technology

Junyang Huang

Generating Dynamic and Lifelike NPC Dialogs in Role-Playing Games Using Large Language Model

Bachelor's thesis

2024

52 pages, 8 figures, 2 tables and 3 appendices

Examiner: Assistant Professor Dominik Siemon

Keywords: Artificial Intelligence (AI), Role-Playing Game (RPG), Non-Player Character (NPC), Large Language Model (LLM), Transformer, Player-NPC Interaction, Generative AI, Game Design, Natural Language Processing (NLP), Human-AI Interaction.

Traditional non-player character dialog systems rely on preset lines, which lack realism and dynamics and reduce player immersion. This study aims to explore overcoming these limitations through large language models.

This paper reviews existing non-player character dialogue systems and emphasizes the need to generate dynamic, context-sensitive dialogs. The capabilities of large language models, especially GPT-3 and GPT-4, in generating coherent, contextually relevant content are explored. In this paper, I propose a framework for non-player role dialog based on large language model. The framework includes key components such as intent recognition, memory retrieval, dialog management, and response generation. The framework was tested in practice in the text-based role-playing game Call of Cthulhu. Evaluations from game participants indicated that the dialog based on the large language model was more natural and contextualized, enhancing the overall game experience.

This study demonstrates the feasibility and advantages of integrating large language models into non-player character dialog systems, providing valuable exploration and practice for developing smarter and more responsive game characters. However, there are still challenges in managing long-term memory and context in non-player character dialog systems based on large language models, which require further research.

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to all the people and groups of people who have supported and helped me in the process of writing this thesis.

First of all, I would like to thank my thesis supervisor, Prof. Dominik Siemon. This cutting-edge research topic initially confused me, but the professor's encouragement and guidance helped me find my way. Despite his busy schedule, the professor still found time to provide me with valuable advice and assistance, which enabled me to finally complete this thesis, thanks to his dedicated support.

I would like to thank Damian Kedziora, the instructor in charge of the course, who helped coordinate the progress of the thesis throughout the research process. He organized the writing of our dissertation action plan and the initial presentation, and Damian provided us with much needed help and support throughout the dissertation course.

I would like to thank my classmates and friends. Special thanks to my roommates, Jiaxuan Qi, Kecen Yin, and Wenyue Zhang, who provided me with much encouragement during my research. Also, thanks to Yufeng Pan, Lele Chang, and Eduard, who listened to my many ideas about my research and gave me very valuable suggestions. Thanks to the above students for participating in my experiments and providing valuable feedback.

Special thanks to my parents for their care and support in my life and studies during the research period, which enabled me to devote myself to the research.

I would like to thank LUT University for providing us with a favorable academic atmosphere and environment, and I have gained a lot of knowledge and skills that will benefit me for the rest of my life.

Thank you to Rockstar Games and CDPR Games for developing fantastic games that not only inspired me to role-playing games, but also inspired my research.

Thank you to OpenAI, whose research in large language model has contributed greatly to the innovation and development of AI technology and provided a solid foundation for my research.

Last but not least, I would like to thank the rock band ONE OK ROCK, whose uplifting music has been a great source of encouragement and support when I was lost in my research.

Once again, thank you to all those who have helped and supported me on this journey!

SYMBOLS AND ABBREVIATIONS

RPG Role-playing game

LLM Large language model

NPC Non-player character. A character in a game which is not controlled by player.

FPS First-Person-Shooter (e.g. Call of Duty series)

RNN Recurrent Neural Networks. A neural network structure for processing sequence information.

CNN Convolutional Neural Networks. A neural network structure. Often used to analyze visual images.

GPT Generative Pre-Trained Transformer. A family of large language models proposed by OpenAI

CoC Call of Cthulhu. A tabletop role-playing game, often in text form.

DnD Dungeons & Dragons. A tabletop role-playing game. Has a complex numerical and combat system.

API Application Programming Interface. A technical means of encapsulating data or service from a website or application through internet requests.

CONTENTS

Appendices

1. Introduction

1.1 Background

In today's increasingly diverse field of digital entertainment, the variety and gameplay of video games has been enriching and increasing. Role-playing games (RPGs) are arguably the primary and most diversified of these genres (Drachen, Copier, Montola, Eladhari, & Stenros, 2009). Originating in the 1980s, RPGs have become an important part of the gaming community due to their rich storylines, in-depth character development, and the impact of player decisions on the direction of the story (Drachen et al., 2009). In RPGs, players typically take on the role of a specific (or self-created) character and explore, interact, and fight their way through a set world in order to complete quests and advance the story. Non-player characters (NPCs) play a crucial role in this process. NPCs are characters created and controlled by game developers that are not under the direct control of the player but play a key role in building the game's worldview, storytelling, and enhancing the player's experience. The quality of the NPC's design and interactions has a direct impact on the player's immersion and satisfaction in the game (Bhutan & Ali, 2024). By providing quests, backstories, emotional exchanges, and feedback on player decisions, NPCs enrich the details of the game world and make it more lifelike. Therefore, studying how to enrich and enhance the interaction experience between players and NPCs is not only a major challenge in the field of game design, but also an important way to enhance players' game experience.

In RPGs, the behavioral logic of NPCs is crucial, which not only shapes the dynamics of the game world, but also directly affects the player's gaming experience. The behavioral logic of NPCs can be classified into two main parts: interactive and non-interactive (autonomous behaviors). Non-interactive behaviors, such as autonomous decision-making by NPCs in battle, have been relatively maturely implemented through AI techniques such as finite state machines and behavior trees (Colledanchise & Ogren, 2018). These

techniques allow NPCs to make behavioral choices that are appropriate to their roles and contexts in complex environments, marking a major advancement in game design. However, NPCs' interactive behaviors, especially conversational interactions, still face significant limitations.

Early NPCs were created based on scripts, which allowed NPCs to react to specific triggers or events, enriching the gameplay to a certain extent (Bhutan & Ali, 2024). As the gaming industry has evolved, many developers have turned to creating more non-linear gaming experiences that allow player choices or actions to shape and direct the story in different directions. While the introduction of branching dialog trees has provided players with a seemingly unique experience, at its core, the dialog system in video games has changed very little over the past 25 years. The choices faced by the player are usually limited to a few pre-determined dialog options, and once a choice is made, the story follows a predetermined path designed by the developer. This simplification of interactions, while easy for players to understand and maneuver, often sacrifices realism and may diminish player immersion, making the game world less appealing (Fraser, Papaioannou, & Lemon, 2018). At the same time, the richness of NPC interactions is directly limited by the amount of content developers prepare for NPC. While writing unique identities, backgrounds, and lines for each NPC can greatly enrich the game world, in practice this practice is limited by the fact that it is time-consuming and difficult to apply universally. Especially in open-world games with numerous NPCs, such as The Elder Scrolls and Grand Theft Auto series, players can easily encounter repetitive and one-dimensional interaction experiences. Enhancing these interactions has the potential to increase immersion and replay value. In addition, the dynamics and consistency of NPC interactions are also a challenge. Although game developers will write lines for important NPCs that reflect their personality traits, as the game story advances and player behavior diversifies, it becomes extremely difficult to maintain the consistency and adaptability of NPC dialog content. This requires that the content of NPC interactions should not only reflect the changes in the game world, but also fit the current situation and identity of the NPC, while maintaining consistency.

With advances in natural language processing technology, large language model (LLM) such as the GPT family have demonstrated the ability to generate coherent, natural and diverse textual content. These models, especially in recent research, have been shown to be able to perform multilingual tasks with the help of a small number of examples or simple instructions, sometimes reaching or exceeding states that previously required extensive fine-tuning with task-specific data (OpenAI, 2020). This ability for "few-shot learning", and in particular the performance of the GPT-3 model through textual interactions without any gradient updating or fine-tuning, provides new perspectives on the improvement of NPC dialog systems. Using the large language model, NPCs' dialog content can be dynamically generated not only according to their identities and backgrounds, but also adapted to the changes in the game world and the diversity of players' behaviors, thus providing a richer, more personalized, and more consistent interaction experience. This technological advancement not only improves the naturalness and depth of NPC-player interactions, but also increases the playability and immersion of the game, providing players with a more vivid and engaging gaming experience. In this way, big language modeling has the potential to revolutionize NPC interactions in games, making them more intelligent and adaptable.

1.2 Objectives

As the field of digital entertainment continues to evolve, gamers' expectations for interactive experiences are growing. Especially in RPGs, the interactivity between NPCs and players has become a key factor in enhancing game immersion and enriching game stories. However, as mentioned earlier, traditional NPC dialog systems are limited by preset scripts and limited interaction modes, making it difficult to meet players' needs for dynamic and personalized interaction experiences. In view of this, this study is dedicated to exploring the use of LLM to enhance NPC interactivity, especially by enriching the dialog content to improve the NPC interaction experience. The main objectives of this study are as follows:

1) System analysis and improvement opportunities

Categorize and discuss the design of existing NPC dialog systems to identify their limitations and potential areas for improvement. This includes examining the design approaches, interaction mechanisms, and player experiences associated with traditional systems. The goal is to pinpoint specific challenges that LLM can address to enhance NPC interactivity and realism.

2) Theoretical framework development

Develop a theoretical framework to build NPC dialog systems based on LLM. This involves reviewing relevant literature on LLMs and their applications in game design, particularly focusing on their ability to generate engaging and contextually appropriate dialog. The framework will guide the integration of detailed NPC settings, including memory information and histories of previous interactions, into the dialog generation process.

3) Practical implementation and evaluation

Design a game scenario and create NPC with unique background and personality trait as an experiment. Utilize LLM to generate dialog contents that aligns with NPC’s characteristic and the evolving game context. The effectiveness of the theoretical framework will be validated through experimental evaluation, examining the authenticity, personalization, and adaptability of the generated dialogs.

By realizing the above research objectives, this study expects to provide new perspectives and inspirations for game developers to create more vivid and resonant NPCs, thus enhancing the overall quality of the game and the player's gaming experience. It is hoped that this research will provide new ideas for the application of natural language processing and artificial intelligence techniques in games.

2. From Traditional NPC Dialog Systems to the Application of LLMs

2.1 Dialog Systems Design in Role-Playing Games

2.1.1 Overview of Dialog Systems Design in Video-Games

In modern video games, dialog systems often play an important role in game mechanics. They contribute to story development, world building, and enhancement of the player experience. The introduction of dialog systems in games not only provides a platform for players to choose the direction of the game according to their own preferences, but also allows developers to tell the story in a richer way, where character-player interactions, character dialog, and character reactions to the environment can be presented to the player as additions to the game's narrative (Lebowitz & Klug, 2011, Chapter 7). Over the years, the game industry has seen the emergence of a variety of game dialog system design that provide players with dialog options in different ways, resulting in unique interactive experiences. In this section, I will explore several major dialog system design patterns and analyze how players and NPCs interact in different dialog modes.

First, based on the flexibility of the dialog system and the player's influence on the content of the dialog, the dialog systems in video games can be briefly categorized into several types: simple choice/hub, non-branching dialog, and branching dialog trees (Fraser et al., 2018).

1) Simple choice/center hub

The most basic dialog system is the simple choice or center hub system, in which an NPC asks the player a series of questions or options and the player responds by choosing one of the options. These options are often not specific to the conversation, but rather are actions or behaviors that the player can choose from. These options are often not specific to the

conversation, but are actions or behaviors that the player can choose to perform, such as "trade", "persuade", or "attack". This way of interaction is often very direct, and the player's decisions have a direct impact on the ensuing dialog and gameplay, for an instance, depending on the player's choices, they are often introduced to a different branch of the dialog or story. Although this approach is easy to implement and understand, it provides limited story depth and replay value because the dialog is fixed and the player's interaction choices are restricted to predefined paths (Salen & Zimmerman, 2004, Chapter 7).

2) Non-branching dialog

In a non-branching dialog system, NPC conversations are usually pre-written and the player has less influence over the content of the dialog. This type of dialog system is common in action-driven RPGs and some first-person shooters (FPS), where pre-written dialog is played out in sequence, and the player has no ability to select or change the content of the dialog. In a non-branching dialog system, the primary purpose of the dialog is to provide backstory or necessary gameplay guidance, rather than as a means of allowing the player to determine the direction of the story. The main advantage of this approach is that it allows for rapid progression of the game's plot while maintaining a fast gameplay pace, but the disadvantage is that it reduces the player's involvement and control over the story (Bateman, 2007, p. 142-145).

3) Branching dialog tree

Branching dialog trees provide a more complex and free-flowing form of interaction, in which the player can choose the content of specific conversations, and different dialog choices can lead to different story threads or endings. This type of system allows the player's choices to have a substantial impact on the game's story and the characters in the game. By choosing different dialogs, the player is able to change not only the direction of the dialogs, but also the state of the game world and the relationships between the characters. For example, by choosing to act friendly or hostile towards NPCs, the player can influence how NPCs perceive and react to them, thus altering the storyline and outcome of the game. The implementation of branching dialog trees greatly enriches the

player's gameplay experience, as multiple dialog branches can form a huge number of combinations, bringing a unique story experience to each different choice and increasing the replay value of the game (Lebowitz & Klug, 2011, Chapter 9)

2.1.2 Dialog Systems Development in RPGs

While the graphical quality and gameplay of RPGs have improved with advances in hardware, developers are always looking for new ways to enhance the quality of storytelling and player immersion. As a result, dynamic and situational dialog systems have emerged on the basis of branching dialog trees, which further enrich the details of dialog content in response to changes in player behavior and game environment.

a) Dynamic dialog system

Dynamic dialog systems are a cutting-edge dialog technology that began with the realization of the limitations of traditional branching dialog trees. Since the conversation contents and conversation paths of traditional branching dialog trees are pre-written and determined during the game development process, although NPCs can show different reactions, these reactions can only change within a limited framework, and are not well adapted to the player's real-time behavior. The goal of the dynamic dialog system is to make the NPC's reaction not only depend on the preset dialog path, but also be able to dynamically generate the dialog content according to the real-time state of the game and the player's past choices. Early attempts at this technique date back to the mid-2000s, and it generates conversations in real-time based on player behavior, historical choices, and the current state of the game environment. Such systems use algorithms to analyze the player's behavioral patterns and adjust the NPC's responses to fit the player's personality and the game context (Freeman, 2004, Chapter 5).

b) Situational dialog system

The situational dialog system can be regarded as a special kind of dynamic dialog system, which emphasizes more on the context sensitivity of the dialog content. The design of the

situational dialog system is inspired by the situational dependence of human conversations in the real world, which means that the content of a conversation is often directly or succinctly influenced by the environment in which the speaker is located. In RPGs, NPCs driven by situational dialog systems will adjust their dialog contents according to the player's geographic location, completed quests, items in their possession, and even changes in game time, making the dialog situationally relevant. The initial development of situational dialog systems focused on how to adjust the NPC's speech based on the player's immediate behavior and game state. It is worth noting that a situational dialog system is not necessarily a true "dynamic system". Developers can design the game in advance to consider the game environment and character relationships with the player behavior changes, and according to some of the main changes in the direction of the dialog designed in advance, so as to give the players an illusion of game world changing dynamically with their own actions. For example, when the player holds or does not hold a key item, the NPC can provide two different dialogs, which are actually written by the developers in advance (Martinez & Ciarletto, 2019).

2.1.3 Impact of Dialog System Design on Player Interaction Experience

The game world can be regarded as a "magic circle" isolated from the real world, which is a clearly defined physical and temporal space with independent rules and frameworks. The magic circle provides players with rules and experiences that are different from those of the real world, and when players decide to participate in the game, they need a specific mental attitude, i.e., accepting the rules of the game and being willing to interact within the framework of the game (Salen & Zimmerman, 2004). The ability to smoothly transition the player's state of mind from the real world to the magic circle constructed by the game determines whether or not the game can provide a good immersive experience for the player. In this regard, the dialog system undoubtedly serves as a bridge between the player and the game world. Dialog is a common behavior between the real world and the game world, which promotes the development of the plot through the mechanism of interaction

and response, and at the same time deepens the emotional connection between the player and the game character (Salen & Zimmerman, 2004). When the player engages in dialogues with NPCs in the game, this interaction shows the character's personality and emotions, at the same time it reflects the player's own choices and personality. So that the player not only establishes an emotional connection with the characters in the game, but also experiences the actual impact of his or her own behavior and choices on the game world (Caroux, Isbister, Bigot, & Vibert, 2015).

Compared with instilling the worldview and background of the game to the player through large-format textual narratives, dialogues, as a dynamic mode of interaction, are more in line with people's real-life communication habits, and allow the player to gradually integrate into the game's socio-cultural environment through natural and smooth dialogues (Pallavicini, Ferrari, & Mantovani, 2018).

At the same time, as RPGs begin to feature beautiful graphics and rich gameplay mechanics, gamers have come to expect a higher level of NPC interaction in their games. Inworld AI, a technology company focused on AI-powered virtual characters, released a survey in 2023 on players' expectations of NPCs in games (Inworld AI, 2023).

The report reveals that 84% of players believe that NPCs have a critical impact on gameplay, 79% of players are actively engaged in in-game NPC conversations, and 59% are willing to take the time to learn about an NPC's backstory. The report further reveals players' dissatisfaction with the current NPC conversation system and their expectations, with 52% of players expressing frustration with repetitive NPC conversations, and a total of 53% of players believing that NPC responses fail to be aware of the player's behavior and surroundings. The report suggests that players are frustrated with NPCs for more than just the limited conversation tree, even though this is the preferred NPC behavior that players would like to see; NPCs' clumsiness in responding to players and their inability to adapt to changes in the game have led to a disconnect between players and NPCs.

Players' demand for smarter and more personalized NPCs reveals the future direction of the dialog system. 76% of players expect NPCs to demonstrate better situational awareness,

while 68% want NPCs to engage in more advanced dialog interactions. Inworld reports that 99% of players believe that the introduction of advanced AI NPCs will have a positive impact on games, and 88% believe it will make games more immersive. Players want to feel more personalized in their conversations with NPCs, such as real-time interactive responses (60%), a sense of humor (60%), memory of the player (55%) and responsiveness to the game environment (53%).

The traditional branching dialog tree is becoming more and more difficult to meet players' expectations due to the limitations of pre-written dialogs and branches, and it would take a lot of time and effort for game developers to implement richer dialogs in the game, for an instance., even writing individual dialog for each NPC. However, the introduction of AI technology, especially large language models with natural language processing and generation capabilities, may bring about a change in this situation.

2.2 Case Study: How to Design an Engaging Dialog System for Players

In order to explore the effectiveness of dialog systems, this chapter analyzes the dialog system designs of three representative games - The Witcher 3: Wild Hunt, the Mass Effect series, and Red Dead Redemption 2 – and demonstrates what elements of a highly immersive dialog system should be in place by showcasing these successful dialog system designs.

The dialog system in **The Witcher 3: Wild Hunt** is known for its depth and complexity. It utilizes a linear + branching narrative structure, where the player is driven by the linear plot to complete the game's quests while also being able to influence the outcome of individual quests and change the course of the entire game through their own choices (Vickery, Tancred, Wyeth, & Johnson, 2018). This design ensures that the player experiences the game story in its entirety while also giving the player a great deal of decision-making power and control over the story, which greatly enhances the player's sense of immersion and emotional involvement in the game. For example, the game's

Bloody Baron quest line demonstrates how the player's choices affect the fate of NPCs, which in turn triggers a series of chain reactions that affect the storyline of the entire area. This complex network of cause-and-effect relationships is what made The Witcher 3's dialog system so popular with many players, and made them want to invest time in experiencing other story branches even after completing the game's main storyline.

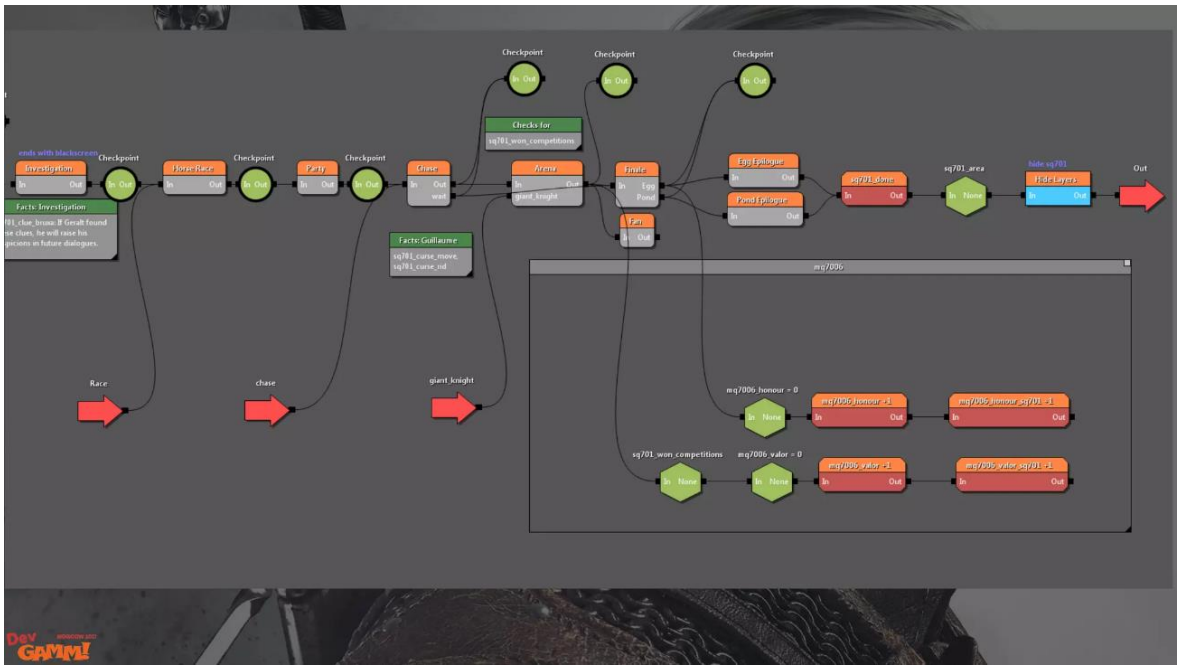

Figure 1. The Witcher 3's dialog content and quest design takes into account the impact that player choices have on the game world (DevGAMM, 2017)

In contrast, the **Mass Effect** series offers a dialog system called the Dialog wheel, resulting in a more unique interaction experience. In this system, all available dialog options are presented in a wheel, and these options represent different personalities or emotional tendencies. Players can choose to respond according to their emotional inclinations, thus shaping the personality of the protagonist and developing the story line. This system places special emphasis on character personalization and emotional development, allowing players to deepen their relationships with teammates or influence the political direction of the game through dialog choices. Each dialog choice is a projection of the character's

personality, and the character's personality affects the game's plot and ultimate ending. The game provides different personality scales for the player's character, Commander John Shepard, such as the "Exemplary" and "Rebellious" tendencies in Mass Effect 1, and when the player accumulates a certain number of points on the scale for a particular tendency, he or she enters the corresponding branch of the game's story. There are limitations to this system, as the number of branches and triggers are fixed, which makes the player's conversations somewhat "utilitarian". In order to enter a certain route of the story, the player may only choose the conversation option that corresponds to that route, without caring about the specific content of the conversation.

Figure 2. The Mass Effect series offers players the “dialog wheel system” that reacts to a character's emotional tendencies (BioWare, 2017)

Among many RPG games, the dialog system of **Red Dead Redemption 2** can be said to be the leading one in the industry, which creates a highly realistic western world through a highly dynamic dialog system and rich feedback mechanism. The game's NPCs adjust their responses in real time to changes in player behavior and the game environment. In the game, the player can start a conversation with a large number of NPCs everywhere, the player can choose to greet, provoke, threaten and other rich interactions, and the NPCs will

be based on different interaction methods and their own background to respond to different responses. For example, if the player threatens a farmer with a gun, he will probably give in to the player's commands. But if the player provokes a dangerous gang member, a firefight is inevitable. Rockstar Games, the game's development studio, scripted the dialog for a large number of NPCs, who, rather than speaking random lines in sequence, actually perceive the player's actions and are aware of the changing environment in the game world. In a way, the interaction system in Red Dead Redemption 2 is a great example of what the future of game interaction could look like, and it gives gamers something to imagine.

Figure 3. In Red Dead Redemption 2, Players can interact with NPCs in different ways and get feedback (Rockstar Games, 2018)

It's easy to see that while great dialog systems have their own distinctive characteristics, there are some common elements that make them successful and popular with players. First, the system needs to be able to provide depth and complexity so that the player's choices have significant impact, thus enhancing the player's sense of control and immersion in the game's story. Second, the dialog system should support character

personalization and emotional development, allowing the player to shape the character's personality and develop the game's plot through dialog. In addition, an excellent dialog system needs to be interactive and adaptive, able to respond in real time to changes in player behavior and game environment.

However, there are inherent limitations of dialog systems under the current conditions. That is, no matter how the design mechanism of the dialog system is adjusted, the NPCs will always speak pre-written lines according to a certain process or script. Rockstar Games wrote individual scripts for almost every NPC, making it seem as if they were actually aware of what the player was doing - but not every game studio was able to invest such a large amount of effort into the dialog system.

In this case, introducing LLM to build dialog systems may be a new and worthwhile path to try. Through their advanced natural language processing capabilities, LLMs are able to generate natural and personalized conversational content in real time, thus overcoming some of the core challenges of traditional dialog systems. This is what I will discuss and investigate in the following sections.

2.3 Theoretical Overview of LLMs

Traditional dialog systems usually rely on predefined scripts and limited interaction logic, which restricts the natural fluency and personalization of conversations. In recent years, the emergence of LLM has provided new technological ways to address these limitations. In this section, I will explore the working principles and core technologies of LLMs, and introduce several representative models to reveal the advantages of LLMs for dialogs.

2.3.1 Transformer as a Basic Architecture for LLM

Before learning more about LLMs and their application in game dialog systems, it is crucial to understand the technology at the heart of these models, the architecture called

Transformer, which has revolutionized the field of Natural Language Processing (NLP) since its been proposed, it has revolutionized the field of NLP, demonstrating significant benefits especially in text generation and comprehension.

Transformer is an architecture first proposed by Vaswani et al. in Attention is All You Need (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, & Polosukhin, 2017). This architecture is based entirely on attention mechanisms and does away with recurrent neural network (RNN) and convolutional neural network (CNN), which were commonly used in previous models. The core technique of Transformer is the self-attention mechanism, which is able to process sequential data while taking into account the relationships between all the elements in the sequence, rather than relying on the sequence's sequential order (Vaswani et al., 2017). In Transformer, the encoder first processes the entire input sequence and passes the information to the decoder, which in turn generates the output sequence step by step. This encoder-decoder architecture is ideally suited for tasks that require the entire input sequence to be considered in order to generate the output, such as machine translation, text summarization, or dialog generation.

The attention mechanism is a weighting method in NLP that is widely used when dealing with sequential tasks involving large amounts of data. The core idea is that when generating each output element, the model can "focus" more on certain parts of the input sequence, rather than treating all inputs equally. Transformer employs a mechanism called "self-attention", which centers on the idea that when working with sequential data, each element can take into account all the other elements in the sequence in order to efficiently capture the complexity of the relationships. This mechanism does not rely on the traditional recursive way of processing data, which greatly improves the processing efficiency and model performance.

In the Transformer organization, the self-attention mechanism works in the following steps.

1) Linear transformations: The self-attention mechanism first applies three different linear transformations to each input element, generating three different representations: Query, Key and Value. These representations will be used in subsequent computation steps.

2) Score calculation: The model calculates compatibility scores between the query and all keys. These scores indicate the relevance of the current output element when considering each input element. The compatibility scores are usually computed by the dot product between the query vector and the key vector, and normalized by dividing by a scaling factor (e.g., the square root of the key vector dimension) to stabilize the training process.

3) Weight assignment: the scores are converted into weights by means of a softmax function such that all weights add up to 1. This step ensures that the model can assign attention to each element based on its relevance when computing the output representation.

4) Weighted sum: the model computes a weighted sum of the value vectors with weights obtained from the previous step. This weighted sum reflects the contribution of all elements in the input sequence to the current output element, resulting in an updated representation of the output element. (Niu, Zhong, & Yu, 2021)

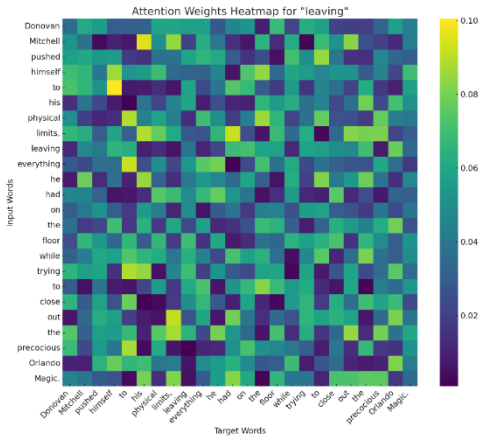

Figure 4. Heat map of attentional weights for a particular word in a sentence. Different colors in the graph indicate different weight intensities, with darker colors indicating higher weights, meaning that the model pays more attention to these words when processing "leaving".

Self-attention allows the model to make connections between different positions in a single sequence, thus capturing long-distance dependencies. Consider that in natural language, the meaning of a sentence is often not limited to close-distance relationships between

words, but needs to depend on words at longer distances in the sentence. For example, in the sentence "I bought some apples at the market yesterday, and they were really fresh and tasty", the word "they" refers to the previously mentioned "apples", even though there are multiple words between them. There is a distance of several words between them. In the traditional sequence processing model, information is lost in the process of transmission. Under the self-attention mechanism, each word can directly access the information of any other words related to it during the generation process, and the model will assign different weights to different words according to the current task (e.g., generating the next word), which reflect the importance of each word to the current task. Even a word that is located far away in the sentence will be given enough attention as long as it is important for the current task.

2.3.2 Implementation and Application of LLM

The transformer architecture and its core component, the self-attention mechanism, have set a new standard for modern NLP tasks. Transformer has not only improved the way text is processed, it has also paved the way for the development of more efficient and flexible language models. Building on this foundation, many large-scale language models have evolved into larger, more complex model structures and have begun to demonstrate superior performance in tasks such as text comprehension and generation. In this section, I will take the GPT family of models as an example, analyze the technical implementation of LLM, and discuss how LLM can improve the naturalness, coherence, and personalization of conversations.

Generative Pre-trained Transformers, known as the GPT family, are LLMs proposed by the OpenAI team starting in 2018. Since OpenAI proposed GPT-3 in 2020, this series of models began to attract a lot of attention and discussion in the NLP field. GPT-3 is known for its large scale - with 175 billion parameters - and superior sample less learning capabilities (OpenAI, 2020), qualities that significantly improve its performance in text generation and dialog system applications.

While the traditional structure of Transformer generates text by mapping input to output, the GPT series is designed to generate continuous text output based on a given text (e.g. a question or a prompt). Therefore, GPT-3 uses the decoder in Transformer to generate text, which predicts the next word on a word-by-word basis, relying on all previous words. The Transformer's decoder is autoregressive because it uses masking to ensure that only previous words are accessed when the current word is predicted, preventing early leakage of information. This left-to-right generation model is well suited for text generation tasks.

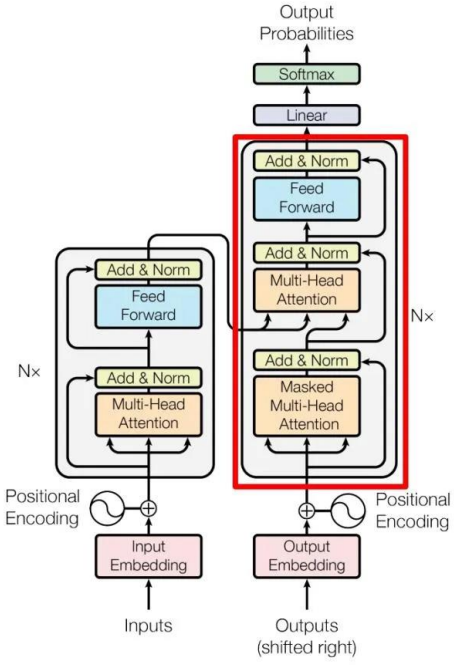

Figure 5. In the Transformer’s decoder block, by using masks, each word can only establish attention with its own previous words, ensuring that the model generates text unidirectionally. (Vaswani et al., 2017)

However, the advantage of GPT-3 lies in the huge number of parameters and extensive pre-training, which makes GPT-3 show the ability to learn in a zero-shot or few-shot environment. In the pre-training process, it uses a large and diverse dataset, which includes books, web pages, news articles, and so on. This enables the model to learn rich language patterns and wide knowledge of the world. Subsequently, the development team allowed

the model to perform the task by specifying the task description and a small number of examples (0, 1 or less than 10) only through direct text interaction with the model. When applying few-shot learning, GPT-3 typically utilizes a formatted prompt approach. These prompts are designed to be embedded directly into the model's inputs, thus guiding the model's output. This approach greatly increases the flexibility of the model, allowing it to adapt to a variety of tasks outside of the training phase, such as specific styles of text generation, code writing, or even answering open-ended questions. The results show that the model is still able to adapt and perform new tasks in this few-shot environment. This reveals that by scaling up the model, LLM has the ability to capture subtle linguistic patterns and complex relationships, which allows the model to be quickly adapted to a variety of NLP tasks without much fine-tuning.

LLMs represented by the GPT family have demonstrated their power in handling text generation tasks in few-shot learning environments. The design of GPT-3 enables it to learn and capture complex linguistic patterns and relations with little to no samples, thanks to its large parameter size and extensive pre-training. The newer generation of GPT-4 model further increases the model size, which includes a larger number of parameters and a larger context window, which allows the model to memorize larger contexts, leading to higher accuracy in text comprehension and generation, and stronger learning capabilities in few-shot or zero-shot environments (OpenAI, 2023). This flexibility and adaptability opens up unprecedented opportunities for NPC dialog systems in game design. Instead of writing a complete dialog script for NPCs, game developers may be able to use a small number of prompts and sample dialogs to enable LLM to generate real-time conversations that fit the context of the NPCs and can be adapted to the player's status.

3. A Framework of Dialog System Based on LLM

In this chapter, I will provide a concrete discussion on how to design and implement a LLM-based dialog system, focusing on the enhancement of NPC interactions in games. The core goal of this chapter is to provide a theoretical framework for implementing an NPC dialog system based on the LLM. Starting from the basic design and construction of the dialog system, I will sequentially discuss and analyze how to integrate the LLM in the dialog system, how to train the model for the dialog tasks in the game environment, and ultimately, how to generate and adjust the dialog content, including how to dynamically generate personalized and contextually relevant dialog content and how to adjust the dialog strategy in real time to adapt to the changes and player behaviors in the game.

3.1 Design and Construction of the Dialog System

3.1.1 Ideas and Principles of Designing a LLM-based Dialog System

Drawing on recent research, ChatGPT-based generative agents are able to create believable simulations of human behavior in virtual environments (Park, O’Brien, Cai, Morris, Liang, & Bernstein, 2023). By storing records of the generative agents' natural language experiences, synthesizing these memories into higher-level reflections over time, and then dynamically retrieving these reflections to plan behaviors, the team enabled the agents to produce believable individual and group behaviors, such as starting with a user-specified notion that an agent wants to host a Valentine's Day party, and autonomously spreading the party's invitations for the following two days that make new friends, invite each other to the party, and coordinate to attend the party together at the right time. The study suggests that the three phases of memory, reflection, and planning play a very important role in the agent's behavioral performance, ensuring that the agent is able to plan for the long term and maintain behavioral consistency. The architecture of generative agents, which enables them

to exhibit more natural and dynamic interactions, serves as an excellent foundation for dialog systems designed to produce more natural and coherent dialog content.

The design of the dialog system is guided by several core principles designed to enhance the quality of interaction with the game's NPCs and the player's sense of immersion (Inworld AI, 2023):

- **Player-centric interactions:** All design decisions are centered on optimizing the player experience, the dialog system should be designed around the interaction between players and NPCs.
- **Dynamic content generation:** The system is able to dynamically generate dialog content based on game context and player behavior, providing coherent and contextually relevant responses.
- **Memory and continuity:** A comprehensive memory system allows NPCs to not only respond to current player input, but also to form responses based on the history of previous interactions.
- **Adaptability and flexibility:** The system has a flexible and modular structure that allows developers or game designers to adapt NPC behavior and dialog logic to different game scenarios and player groups through configurable parameters, and the system should be designed to support the reuse or replacement of specific modules in different games.

3.1.2 Components and Processes

Research on generative agents has demonstrated that an agent's retrieval and reflection on its own memory plays a very important role in increasing the credibility of its behavior. Therefore, implementing and maintaining a memory bank in an NPC dialog system driven by a large language model may significantly improve the naturalness and coherence of dialog content. Taking this as a starting point, this paper proposes the following framework

for a dialog system, which will be based on four functional modules.

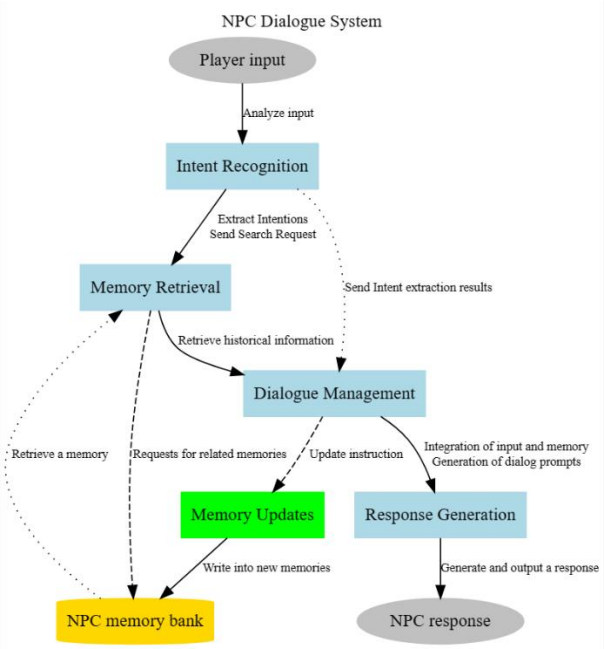

Figure 6. The dialog management module forms prompts based on the results of intent extraction and memory bank retrieval to instruct the large language model to generate NPC responses. New responses are updated to the memory bank by the memory retrieval module.

- **Intent recognition module**

The Intent Recognition Module is responsible for parsing the text input by the player. The aim of this module is to recognize the player's intention and key information in the conversation (e.g., goals, questions, or commands) using natural language processing techniques. This step is to ensure that the system understands the player's needs and provides accurate cues to the model in the subsequent generation of responses.

- **Memory retrieval module**

This module searches the NPC's memory bank for historical information relevant to the current player interaction. This information includes the player's early decisions, the content of previous conversations, and the NPC's memory of the player's behavior. The

results of the memory retrieval will be used to help form more contextual and contextualized responses.

- **Dialog management module**

The dialog management module is at the heart of the process, integrating the results of intent recognition and memory retrieval to generate specific cues that guide response generation. This module is also responsible for maintaining the flow and logical consistency of the dialog, ensuring that it follows the plot and characterization of the game.

- **Response generation module**

Based on the hints received from the Dialog Management module, the Response Generation module uses pre-trained large-scale language models (e.g., GPT-3) to generate natural, fluent responses. These responses are a direct reflection of the NPC's personality and game context, as well as a direct reflection of the player's interaction with the game world.

- **Memory updates**

The memory update function is integrated into the dialog management module and is responsible for updating new dialog content into the NPC's memory bank. This step occurs after the response has been generated and ensures that all interactions with the player are memorized for future use. This helps the NPC to continuously update the context of the dialog to adapt to changes in the game world and the player's state.

3.2 How LLM "Plays" NPC’s Role in Game

LLMs have shown their remarkable capabilities in a variety of text generation tasks, especially in generating natural, coherent, and highly contextually relevant texts (Naveed, Khan, Qiu, Saqib, Anwar, Usman, Barnes, & Mian, 2023). In RPGs, these models are a foundation for generating high-quality dialog content due to their powerful logical reasoning capabilities and extensive common sense knowledge base. But also, more

importantly is how they are able to "play" specific roles in the game. In games, NPCs are not only tools that provide help and guidance to players, but their backstories and personality traits are also key elements that enrich the player's gaming experience (Lebowitz & Klug, 2011, Chapter 4).

How to utilize large language models to enhance the "realism" of NPCs is a challenging and fascinating topic. This chapter will analyze in detail the application of macrolanguage models in understanding and generating dialogs that match the specific backgrounds and personalities of NPCs, and how these techniques can help NPCs interact with players more authentically in games.

3.2.1 Learn and Utilize Character Attributes

LLMs have demonstrated unprecedented capabilities in areas such as mathematical problem solving, logical reasoning, code debugging, and image understanding and generation (Kojima, Gu, Reid, Matsuo, & Iwasawa, 2022). However, these models still show limitations in achieving socialization goals such as persistent memory construction, emotional connection building, complex emotion understanding and simulation, and handling scenarios that require delicate social skills. In order for a large-scale language model to understand and mimic the conversational style of a game character, it is crucial that it masters the character's personality and emotional traits. This process involves deep customization of model and character attributes with the aim of generating personalized dialog content that is consistent with the character attributes. In this regard, the CharacterGLM model, which is based on specialized training for character dialog tasks, provides important theoretical insights. The model successfully generates conversations with a high degree of personalization and emotional depth by accurately setting and exploiting detailed attributes of the character, such as identity background, interests, and social relationships (Zhou, Chen, Wan, Wen, Song, Yu, Huang, Peng, Yang, Xiao, Sabour, Zhang, Hou, Zhang, Dong, Tang, & Huang, 2023). This provides important insights into the development of a theorized approach for customizing LLM to play game characters.

- **Character traits capture and simulation**

First of all, in order for LLM to effectively "play" game characters, the characteristics of each character must be defined in detail, including historical background, motivation, personality, and relationship with other characters. By analyzing these dimensions of a character in depth, a dataset can be constructed that contains the necessary information to train the model. For game developers, the dataset can be constructed from game scripts, or by writing character descriptions and a small amount of dialog scripts for NPCs.

- **Preparation and enhancement of training data**

The key is how to prepare and augment the training data to support the model in learning these complex character traits. This includes extracting dialog and situation descriptions from narrative texts, as well as generating new training samples from interactive episodes. In addition, data enhancement techniques such as dialog reconstruction and contextual simulation can be used to expand the dataset to provide a more comprehensive coverage of possible character behaviors and reactions.

- **Model training and fine-tuning**

Using the above data, transfer learning and fine-tuning strategies can be applied when training a large language model. Starting from a pre-trained generic model, it is customized to the needs of a specific character. The fine-tuning process focuses on optimizing the model's generation strategy so that it can accurately express the character's unique language style and emotional coloring while maintaining text naturalness and coherence.

- **Evaluation and iteration**

After the training is completed, the effectiveness of the NPC's dialog is tested through interactions with real players. This includes not only evaluating the naturalness and emotional quality of the dialog, but also testing whether the NPC's response to the player's behavior is in line with his or her character's setting. Based on the feedback, the performance of the model is continuously optimized, and the NPC's conversational capabilities are further refined through iterative training.

For LLM to play more lifelike game characters, it is necessary to add richer and more realistic emotional expressions to the dialogs, and to accurately capture and present the personality and emotional traits of NPC. By carefully categorizing NPCs' personalities, we can further deepen our understanding of their psychological and emotional traits, enabling LLM to generate dialog content that is consistent with the NPCs' predefined personalities.

Incorporating an affective computing (AC) system into the design and creation of NPCs provides an effective way to enable NPCs to exhibit more complex and varied human emotional states.AC outputs psychometric values to finely reflect the personality traits of NPCs and simulate more human-like emotional states in conversations (Klinkert & Clark, 2021).

Lawrence and Corey's study utilized the Big Five personality theory (Big Five) to differentiate personality traits of NPCs by categorizing personality into five dimensions: openness, responsibility, extraversion, agreeableness, and neuroticism, each of which is represented on a continuum scale ranging from 0 to 1, where a 1 indicates a full expression of a particular trait and a 0 indicates an embodiment of its opposite trait. These five dimensions constitute a comprehensive model of personality that accurately describes the psychological and behavioral characteristics of an individual. These dimensions provide very crucial guidance for large-scale language models (LLMs) to generate dialog content that is highly consistent with NPC personality (Lawrence, Buongiorno, & Clark, 2024).

The use of quantitative personality models allows the personality characteristics of NPCs to be accurately reflected in their behaviors and dialogs, making their performance in the game more fleshed out and realistic. Highly open-minded NPCs may exhibit curious and creative dialogs, while highly responsible NPCs may be cautious and reliable in their conversations. Extroverted NPCs may be more talkative and outgoing, NPCs high in pleasantness may be more friendly and cooperative in their interactions, and NPCs high in neuroticism may exhibit mood swings and sensitivity in their conversations.

3.2.2 Prompt-Based Dialog Content Generation

At very large parameter scales, the best capability of a large language model is its ability to learn under both few-shot and zero-shot conditions. This ability is mainly realized by prompts (Kojima et al., 2022). Recently, the prompt approach has become an emerging training paradigm in the field of NLP. This approach guides the model through a downstream task by utilizing textual cues, and unlike traditional supervised learning, it does not directly predict the output, but rather modifies the input to a text string containing unfilled slots, and then utilizes the language model to predict the information that fills those slots. The advantage of this approach is that it allows the language model to be pre-trained on a large amount of raw text, and by defining new cueing functions, the model is able to learn with a small number of samples or even zero samples, thus adapting to new scenarios with small or unlabeled data (Liu, Yuan, Fu, Jiang, Hayashi, & Neubig, 2021).

The primary purpose of prompts is to direct the focus of the language model, helping it to understand and generate the particular type of dialog it is interacting with the player. In game design, this approach can help the model quickly adapt to changing game situations, such as player actions, the development of in-game events, or the behavior of other NPCs.

In order to effectively construct prompts in a game environment, it is necessary to take into account the state of the player and the game world, as well as the individual settings of NPCs.

1) Clarify the purpose and context of dialog

In game design, every conversation should serve a clear purpose, which may be to advance the storyline, provide a quest, add depth to the game, or enhance player immersion. Depending on the goal of the dialog and the context in which it takes place, the construction of hints will vary. If the goal of the dialog is to advance the storyline, hints may include key plot points, such as "alerting the player to an impending danger" or "revealing an important backstory element". In quest dialogs, hints may incorporate the current quest status and the player's progress, e.g., "Provide clues to the next quest stage"

or "Reward the player for completing the previous quest".

2) Integrate game data

Dynamically incorporating game data into hints can make dialog more specific and relevant, enhancing the player experience. This approach is especially useful in scenarios that need to reflect the player's historical behavior or current game state. If an NPC needs to provide quests based on the player's character level, the prompt could be "Provide quests of appropriate difficulty based on the player's level". If the player has had significant previous interactions with a particular NPC, the hint could include the results of those interactions, such as "Remind the player that their previous choices helped the NPC, and that the NPC will now return the favor.

3) Utilizing conditional prompts

Conditional prompts dynamically change based on the current state of the game or the player's history of behavior, which can make NPC responses richer and more personalized. If a player chooses to help an NPC in the game, the prompt for a follow-up conversation could be "Mention past help, reinforce trust, and provide deeper help or information". In specific game environments, such as when the player is exploring ruins, the prompt could be "Show the player hidden clues or story elements based on the environment.

4) Increase the variety and depth of dialog

Design multi-level hints to cover not only basic dialog options, but also more complex emotional and moral decisions, so as to enhance the humanization and versatility of NPC dialogs. When building prompts, consider including emotional elements, such as "NPC feels sad because they lost a friend, expresses this emotion and asks the player if they would like to help find out why". In critical narrative situations, prompts can be designed to guide the player to make moral choices, e.g., "The NPC is asking the player to make a choice to help them steal supplies to save the village, but this may violate the game's laws".

4. Experiment Implementation

In order to further explore the potential of LLMs in improving the dialog content of games, in this chapter, I will design an experiment to evaluate the application of LLMs in generating dialogs of NPCs in RPGs. Through the experiment, I hope to validate the utility of big language modeling in games and explore its potential limitations and directions for improvement.

The experiment will be conducted in a text-based game environment such as Call of Cthulhu. One or more NPC characters based on a LLM will be created and a series of interaction scenarios with these NPCs will be designed.

CoC is a tabletop role-playing game based on the works of acclaimed author H.P. Lovecraft, known for its rich backstory and mysterious, terrifying atmosphere. During the course of the game, the player often needs to interact with NPCs to solve complex mysteries and advance the storyline, a process that relies heavily on the quality and depth of the textual dialog. CoC's game environment provides an ideal testing ground for demonstrating the model's ability to generate dialogs with emotional depth and personalized responses, as well as for testing the model's understanding and adaptability to complex game situations and character background settings. In addition, the open dialog environment of the CoC game provides space for the model to demonstrate its creativity and flexibility in language production. In this experiment, the model is not trained with large amounts of dialog data, but simply provide the model with the game environment and the background settings of the characters. The purpose of this is to test the model's ability to learn in a few-shot environment, particularly to simulate NPCs to produce natural and coherent dialogs given the initial information.

4.1 Game Background Setting

For the experiment, the CoC single player module "Alone Against the Flames" was chosen as the background for the game. In this module, the player takes on the role of a self-created protagonist, who talks and interacts with other characters in the story (usually played by a moderator) in order to gradually solve the mystery the player is caught up in. The story is set in the United States in the 1920s, starting in a small town and progressing to the fictional city of Arkham. And the story follows the protagonist's journey to Arkham as he leaves his hometown in pursuit of a new life. During the journey, the protagonist encounters a series of challenges and choices, and each of the protagonist's decisions may lead to a different ending. And the story follows the protagonist's journey to Arkham as he leaves his hometown in pursuit of a new life. During the journey, the protagonist encounters a series of challenges and choices, and each of the protagonist's decisions may lead to a different ending. At the same time, the entire story, and the core worldview of the CoC, revolves around the Cthulhu Mythos - a series of supernatural horror stories depicting humanity's encounters with ancient and powerful cosmic entities, often referred to as the "Dominion of Old". The existence of these entities challenges humanity's conventional understanding of reality, often resulting in a distortion of the rules of nature and spiritual destruction. In CoC, players take on the role of what are generally known as Investigators, and are required to solve unusual events caused by these mysterious entities.

When constructing a dialog system, it is necessary to effectively provide this story context and worldview setting to the big language model and ensure that the model takes this information into account each time a dialog is generated, which ensures that the worldview, culture, and time period of the game are reflected in the dialog.

Considering that the background information of the game as the underlying setting of the game will not be changed freely, it can be written as a static document. Using the document as an additional knowledge to the model's own knowledge base allows the model to retrieve information from the document based on user queries when generating

dialogs. In the context of this experiment, I used the "Alone Against The Flames" rulebook directly, which contains important information such as the backstory, time, location, main characters, and CoC rules.

Background documents need to be converted to a model-understandable format before they can be used. OpenAI provides a feature called "document search" that automatically parses and chunks documents, creates and stores embeddings, and converts the text in the document into vectors in a high-dimensional space. These vectors capture the deep semantic information of the text, allowing the model to refer to this semantic information when generating dialog. The vectorized text data is stored in a vector database so that the model can retrieve the relevant information in real time when generating dialogs. The vector database supports efficient semantic search, allowing the model to find and refer to the most relevant contextual information based on the current dialog context.

```
1.  from openai import OpenAI
2.
3.  client = OpenAI()
4.
5.  assistant = client.beta.assistants.create(
6.    name="CoC-Assistant",
7.    instructions="An assistant designed to play NPCs using game background k
    nowledge.",
8.    model="gpt-4o",
9.    tools=[{"type": "file_search"}],
10. )
11.
12. # Create a vector store to store game background document
13. vector_store = client.beta.vector_stores.create(name="CoC-background")
14.
15. # Ready the files for upload to OpenAI
16. file_paths = ["edgar/background.pdf"]
17. file_streams = [open(path, "rb") for path in file_paths]
18.
19. file_batch = client.beta.vector_stores.file_batches.upload_and_poll(
20.   vector_store_id=vector_store.id, files=file_streams
21. )
22.
23. assistant = client.beta.assistants.update(
24.   assistant_id=assistant.id,
```

```
25.   tool_resources={"file_search": {"vector_store_ids": [vector_store.id]}},
26. )
```

Create a new assistant using OpenAI's API and upload the game's background document as a vector store and bind it to the assistant's knowledge base. This way the model will reference this extra information when generating content. This effect is consistent with fine-tuning the model to enhance its response to specific content, but has the advantage that the additional knowledge can be updated or replaced easily (OpenAI, 2024).

4.2 Building NPC Information and Memory Bank

4.2.1 Defining Character Attributes

For information related to an NPC, it can be simply distinguished between static information and dynamic information. Static information is set when the NPC is created, and can be a template that defines basic attributes such as the NPC's name, age, occupation, and race, as well as a description of the NPC's personality and a short backstory. This information defines the basic attributes and personality traits of the NPC and provides the necessary context for the NPC's behavioral logic and dialog interactions with the player. Such information provides a good basis for the construction of hints, allowing the model to generate dialog content in a way that ensures consistency between the content and the NPC's identity.

Depending on the background and setting of the game, the specific entries in the NPC information templates can be changed and adapted. The main purpose of organizing NPC information in templates is to provide a structured way to understand the background of NPCs to the Big Language Model. In this experiment, a similarly simple template will be used to create NPCs and compile information about these NPCs into a document that will be uploaded to the model as additional knowledge.

Table 1. Example of creating information template for characters in the story

Attributes	Description
Name	May Ledbetter
Gender	Female
Age	40
Race	Caucasian
Occupation	lady innkeeper
Personality	Kind and helpful, but can be a bit secretive and conflicted under pressure
Background	May is a widowed matriarch who runs a small inn passed down from her family. Her husband was accidentally killed during a festival a few years ago. Since then, she has been left alone to care for her daughter, Ruth, and try to make ends meet. May may appear friendly and caring to travelers on the surface, but she is deeply involved in the town's festivals and traditions, which may be hiding disturbing secrets.

When LLM needs to generate a dialog for an NPC, it first reads the static information template for that NPC. This includes knowing its occupation, age, personality, etc. This information helps the model to build a consistent characterization. Based on the NPC's backstory and personality description, the model can generate the appropriate dialog in a specific context. For example, an NPC who has experienced grief may show more emotional fluctuations when mentioning a loved one. In addition, the character traits of NPCs will also affect the content and style of the dialogs generated by the model. For example, a serious NPC may use more formal language, while a lively NPC may use more slang and humorous expressions.

4.2.2 Constructing Dynamic Memory Bank for NPC

Background information about NPCs sets a solid foundation for dialog content and style. However, as the game narrative advances, the environment of the game world, the player's behavior and the NPC's own state may change. In order to dynamically reflect these changes in the dialog, it is especially important to build a dynamic memory bank for each NPC. Such a memory bank not only stores the history of the NPC's interactions and key events, but also updates it in real time to reflect the latest developments in the game and

the player's choices. In this way, NPCs' dialogs and behaviors can be adapted to the ever-changing game scenarios, thus enhancing their realism and interactivity. This in-depth memory system significantly enhances player immersion, giving them a clear sense of how their choices actually affect the game world and the lives of its inhabitants.

To build such a dynamic memory, knowledge graph is a very suitable data structure that can store and manage complex entities and their interrelationships graphically for easy retrieval and updating.

The process of constructing a knowledge graph is very simple, and it contains only two attributes.

- **Entities:** including characters (e.g., NPCs, player characters), locations (e.g., towns, rooms), events (e.g., meetings, festivals), and items (e.g., keys, weapons).
- **Relationships:** define the interactions between entities, e.g. the "located" relationship connects a character and a location, indicating the current location of a character; the "owned" relationship connects a character and an item, indicating the item owned by a character; The "involved in" relationship connects a person to an event, indicating an event in which a person is involved.

Using the graph database Neo4j, I built a memory bank for several key characters in "Alone Against The Flames". This memory bank contains three types of nodes: people, places and events, which are connected by different relationships.

```
// Nodes for characters
CREATE (may:Person {name: "May Ledbetter", role: "Innkeeper", age: 40, traits: "Helpful but secretive"})
CREATE (ruth:Person {name: "Ruth Ledbetter", role: "Daughter", age: 10, traits: "Shy and mysterious"})
CREATE (silas:Person {name: "Silas", role: "Coach Driver", traits: "Talkative and careless"})

// Nodes for locations
CREATE (inn:Location {name: "Ledbetter Inn"})
CREATE (village:Location {name: "Emberhead Village"})
CREATE (festival:Location {name: "Festival Grounds"})
```

```
10.
11. // Nodes for revents
12. CREATE (meeting:Event {name: "Meet Player", description: "First meeting wi
    th the player"})
13. CREATE (festivalEvent:Event {name: "Annual Festival", description: "Key ev
    ent of the game"})
14.
15. // Nodes for relationships
16. CREATE (may)-[:LOCATED_AT]->(inn)
17. CREATE (ruth)-[:LOCATED_AT]->(inn)
18. CREATE (silas)-[:LOCATED_AT]->(village)
19. CREATE (may)-[:PARTICIPATES_IN]->(festivalEvent)
20. CREATE (ruth)-[:KNOWS]->(festivalEvent)
21. CREATE (silas)-[:PARTICIPATES_IN]->(meeting)
```

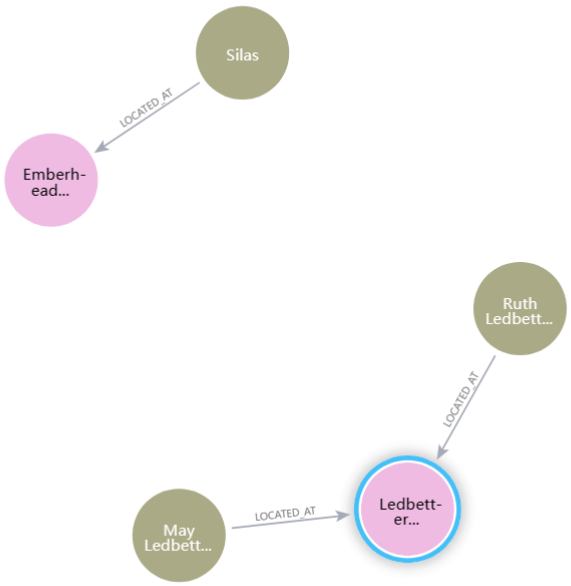

Figure 7. Using knowledge graphs to build memory banks can characterize the real-time status of NPCs

When it is necessary to generate dialog or behavioral decisions, GPT-4 constructs queries based on the current game scenario and the NPC's role status. These queries are designed to retrieve specific information related to the NPC, such as location, item holdings, event participation history, and relationship status with other characters. The query results will provide information about the current state of the NPC, including its location, behavior

history, and relationship network. Subsequently, GPT-4 will integrate the query results into the generated context to ensure that the generated dialog or behavioral decisions reflect the NPC's latest state of memory. As the narrative progresses, the NPC's memory is continuously maintained and updated, e.g., by updating the NPC's location, changing its relationship status with other characters, or adding new event participation records.

```
from neo4j import GraphDatabase

class Neo4jConnection:
    def __init__(self, uri, user, pwd):
        self.__uri = uri
        self.__user = user
        self.__pwd = pwd
        self.__driver = None
        try:
            self.__driver = GraphDatabase.driver(self.__uri, auth=(self.__user, self.__pwd))
        except Exception as e:
            print("Failed to create the driver:", e)

    def close(self):
        if self.__driver is not None:
            self.__driver.close()

    def get_npc_details(self, npc_name):
        with self.__driver.session() as session:
            query = "MATCH (n:Person {name: $name}) RETURN n"
            result = session.run(query, name=npc_name)
            try:
                record = result.single()
                if record:
                    npc_details = record['n'].properties
                    print(npc_details)
                    return npc_details
                else:
                    print("No records found.")
                    return None
            except Exception as e:
                print("Error retrieving NPC details:", e)
                return None

uri = "neo4j+s://58a04d55.databases.neo4j.io"
```

```
user = "neo4j"
password = "Ih2gAeI1AOth_7ARqLjEPPjN7A_ytKBXHnAyztqJuhI"
neo4j_conn = Neo4jConnection(uri, user, password)
npc_details = neo4j_conn.get_npc_details("May Ledbetter")
neo4j_conn.close()
```

By building APIs for memory bank querying it is possible to give the model access to dynamic information about NPCs. In the experiments, I will manually indicate when the model needs to retrieve or update the database. In a real game context, however, it is certainly more efficient to integrate the API into the model. The model will determine whether or not to retrieve or update the NPC's memory bank based on the dialog input from the player, and will perform the relevant query operations via the API.

4.3 Using Prompts to Guide LLM to Generate Dialogs

In order for the GPT-4 to better "act" as a character in a story, effective prompt design is essential. Prompts define the content and style of the dialog and can guide the model to more accurately understand and perform specific tasks. In RPGs, the dialogs of NPCs must fit their character settings and situational environments to enhance story coherence and player immersion. Accurate cues can help models generate dialogs that fit the NPC's backstory or display appropriate emotional responses in a given game plot, making the dialogs not just procedural interactions, but also able to reflect the story progression in a more dynamic way.

LangGPT is a revolutionary prompt design technique inspired by structured programming languages. This framework allows hints to be highly structured and reusable like software code through a modular design, and the introduction of LangGPT provides a methodological basis for building complex and feature-rich hints, making it easy for non-AI experts to design high-quality hints. In addition, the two-layer structure of LangGPT allows researchers to customize and extend the hint module to suit different game scenarios and character requirements (Wang, Liu, Zhang, Li, Huang, Zhang, Wang, Feng, & Li, 2024). This provides an effective tool for this experiment to design effective hints for GPT-

4.

Each prompt in the LangGPT framework consists of multiple modules. These modules are analogous to classes in a programming language, and each module represents a requirement or function for LLM. Such as:

- **Constraints module:** defines the rules or restrictions that LLMs need to follow when generating content.
- **Goal module:** Defines the specific goals that users wish to achieve through LLMs.
- **Process module:** Detailed steps that guide LLMs in performing their tasks, similar to functions in programming.

Each module contains a number of internal elements, which are analogous to functions and attributes in a programming language, that specifically instruct the LLM on how to perform their tasks.

I designed a prompt template (see appendix 2) to guide GPT-4 in the experiment to generate dialog for NPC. The prompt template includes static information about the character, such as name, age, and occupation, and also incorporates dynamic information, such as the character's current emotional state and recent history of interactions with the player. Specifically, the template begins by defining basic background information about the character, including their role in the game, historical background, and personality traits. For example, May Ledbetter's basic information as the innkeeper's wife marks her age, occupation, and secrets related to local traditions. In addition, the template contains a dynamic information section, which is updated according to the progress of the game, such as the NPC's location and involvement in the game events. When generating dialog, GPT-4 first reads this static and dynamic information and then constructs responses based on this data. For example, if the player has recently made certain decisions that have affected May's mood or her perception of the player, this information will be reflected through the dynamic information module and will guide the GPT-4 to generate the appropriate dialog, such as showing May's hostility or trust towards the player.

5. Results Analysis and Discussion

5.1 Experiment Methods

Combining the NPC information and the game background, I created a customized GPT based on GPT-4 and invited 5 testers other than me to have a conversation with this GPT. I was also a participant in the experiment, and during the interaction with the model, I utilized the memory bank created for the NPCs and had the GPT retrieve and update the memory bank as appropriate as the story progressed. For the other participants, I sampled only the first 15 rounds of conversations, focusing on analyzing the GPT's use of initial background information. Five participants had varying degrees of experience with RPGs, two of them were devoted players of the genre and had played more than five RPGs. Two of the other three participants had played only one RPG, and the other one had never been exposed to this type of game.

Although text-based, CoC is essentially a fairly free-form role-playing game, where players are free to decide their next move at most points in the story, opening up different branches of the plot. Therefore, I collected all the conversation transcripts from the participants in the experiment to visualize the differences in story development and dialog contents of the character played by GPT-4. In addition, a short questionnaire was written to obtain feedback and suggestions from the participants, refer to appendix 3. The questionnaire consisted of a few simple questions that allowed participants to evaluate their own interaction experience.

The questionnaire was distributed to each participant after completing the experiment, aiming to collect references from the player's point of view as to whether or not utilizing a large language model to play as an NPC can enhance the immersion of the game.

5.2 Experiment Results and Analysis

In the experiment, the dialogs generated by GPT-4 was effectively embedded in the Call of Cthulhu game context. For example, NPC May Ledbetter mentioned the history of the town of Emberhead and its unique festival, the Beacon Festival, several times in her interactions with the player.

Sample dialog (Excerpts of some of the transcripts of the dialogs are shown in appendix 1):

- "*Emberhead is a small town, but it has its own unique charm and ..... traditions. Running this inn is a way for me to preserve our family's heritage.*"
- "*The Festival of Lights is a major event in Emberhead and this festival is just around the corner. The whole town will gather to celebrate this festival that has been passed down from generation to generation.*"

The content and manner of May Ledbetter's dialog also fits well with her characterization as the innkeeper's wife. She displays a warm and friendly demeanor while revealing a deep reverence for and reservation of the town's traditions and secrets. Her performance in dialog includes:

1) Warmth and hospitality: May repeatedly offers to help the player and provide advice on lodging and dining, which fits her role as the innkeeper's wife.

2) Cautious: When talking about the town's secrets and dangers, May displays a cautious and slightly mysterious demeanor, which reflects the game's atmosphere of suspense and horror.

Sample dialog:

- "*I'll make sure I have a cozy room ready for you. Please feel free to enjoy our hot soup.*"
- "*Emberhead has its mysteries, and some visitors find the town a bit ..... disturbing.*"

In addition, I counted the recorded conversations of several participants. Although 15

rounds of conversations is a small sample size, the model has been able to adjust the development of the game's story and May's responses according to different conversations entered or selected by the players.

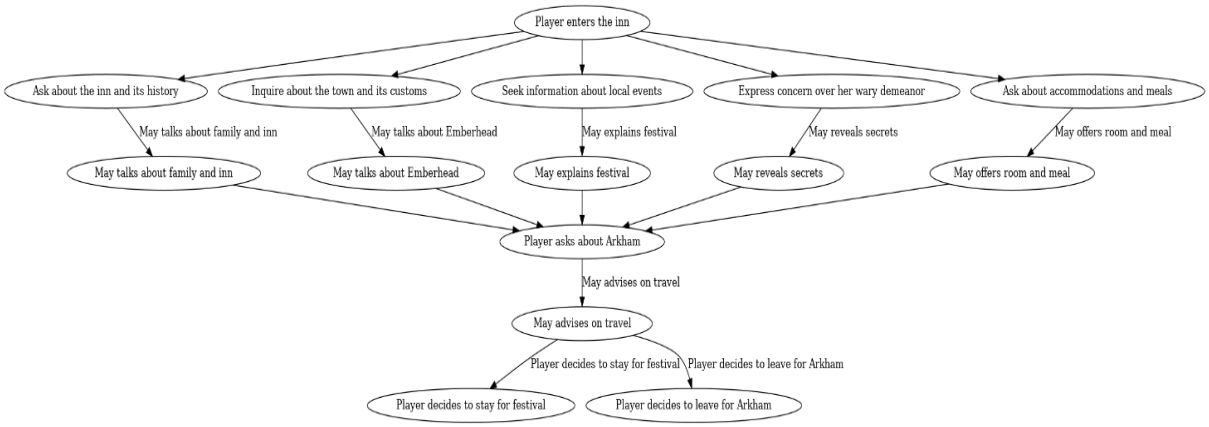

Figure 8. The model exhibits a branch of the story

As shown in Figure 8, the story begins when the player enters the inn and has an initial conversation with the NPC May Ledbetter. At this point, the player can choose from the following five dialog options:

- *Ask about the inn and its history*.
- *Ask about the town and its customs*.
- *Learn about local events*.
- *Express concern for May's vigilant attitude*.
- *Ask about lodging and dining*.

Depending on the player's choices, May will respond accordingly. For example, if the player asks about the inn and its history, May will talk about her family and the history of the inn; if the player expresses concern about her vigilant attitude, May may reveal the town's secrets. Next, whichever initial dialog the player chooses will ask for further information about going to Arkham. Eventually, May will advise the player on how to get

to Arkham and provide relevant advice, and the player may choose to stay in the town for the festival or decide to leave for Arkham.

The model's ability to generate appropriate dialog content based on the different choices made by the player demonstrates its flexibility and adaptability in dealing with complex dialog situations. At the same time, the model is able to adjust the dialog content according to the player's input while taking into account the consistency of the game's context and characters. May Ledbetter's dialogs are always in line with her role as the innkeeper's wife, which demonstrates her warm and hospitable personality, but with a deep respect for the town's traditions and secrets.

In the questionnaire session, the experiment participants generally showed high satisfaction and favorability towards the story characters played by GPT-4, as shown in Table 3. Some open-ended questions were also positively answered by the participants, such as "I hope more details about the game world can be included in the dialogs, such as historical events and character relationships", "I suggest that more scene descriptions be added in the dialogs to give players a more comprehensive experience of the worldview of the game", etc. However, in general, LLM has shown very good potential in playing NPC dialogs.

Table 2. Statistics on the results of the questionnaire

Metrics	Feedback	Frequency
Dialog naturalness and fluency	Very natural	4
	More natural	1
Situational adaptation	Always appropriate.	2
	Appropriate for most of the time.	3
Enhanced immersion	Extraordinary enhancement	3
	Somewhat increase	2
Overall satisfaction	Very satisfied	3
	Satisfied	2

This suggests that by using appropriate methods to provide the game's contextual and character information to the LLM, it can effectively produce contextually and character-consistent dialog content. The ability of the LLM to capture contextual relationships also

enables it to instantly adjust its output according to the user's input, thus effectively enhancing the player's interactive experience and immersion during the game.

5.3 Further Discussions

In this experiment, I explored the potential and effectiveness of large language models such as GPT-4 for generating NPC dialogs in role-playing games. Although the results show that GPT-4 performs well in enhancing player immersion and interactive experience, there are still some limitations that need to be further explored and addressed in a longer-term perspective.

LLMs such as GPT-4 possess certain contextual memory capabilities and are able to memorize the player's previous inputs and the resulting development of the game story during a conversation. However, after a certain number of dialog rounds, the model's memory ability begins to show obvious deviations and confusions.

In the experiment, I conducted alone as the player, who engaged the model in 60 rounds of dialog. The model remembered player’s decision to stay in town for the festival in the 10th round, but May forgot about it in the 50th round when player mentioned again that his destination was Arkham. This may be because LLMs, including GPT-4, have a fixed-size context window and can only remember a certain amount of context. When conversations exceed this window size, earlier information may be truncated or forgotten. In addition, LLMs based on the Transformer capture contextual dependencies through the attention mechanism, and as the number of conversation rounds increases, the most recent conversations will receive higher attention weight, while earlier conversations may be gradually "forgotten" by the model. As a result, the model's memory capacity decays as the number of conversation rounds increases, leading to the model's inability to maintain a consistent memory over long conversations.

Such limitations have been defined in some previous literature. In Tsai, Zhou, Liu, Li, Yu, & Mei's study (2023), ChatGPT was used as a player to play a text adventure game called

Zork I. It was found that ChatGPT was not capable of constructing a complete model of the game world. It could not fully understand the rules of the game world, nor could it memorize the long-term game goals. Although ChatGPT is able to generate reasonable game instructions in the short term, showing intelligence close to that of a human player, as the game progresses, it shows an increasing tendency to repeat previous instructions, which suggests that its inferences about the game goals are inconsistent, and that early memories are truncated when it receives more contextual information than the model's contextual window (Tsai et al., 2023). Similarly, a study using a LLM to play Dungeons and Dragons (DnD) found that the model's accuracy in tracking complex game states was low, and declined as the game's story was enriched (Burch, Tomar, Martin, Ippolito, Bailis, & Reitter, 2022).

Additional technical means such as an external database can help the model manage longer contexts, thus enhancing the model's memory capacity. In this experiment, I propose to maintain an external memory bank for NPCs to persistently store the player's choices and events, ensuring that the NPCs can still remember the important decisions and events of the players even if there are more conversation rounds. By querying the memory bank, the model can retrieve up-to-date, accurate contextual information when needed, avoiding confusion and forgetfulness. External memory repositories, such as knowledge graphs, can be used in text-based games as an auxiliary storage tool for the model's context. In larger game environments, such memory repositories can be integrated into the game to automate the retrieval and updating process and improve efficiency. The DnD-like text game AI-Dungeon, built on GPT-2, uses an additional memory mechanism to store the context of the player's interactions with the characters played by the models. The memory also allows the player to retract or edit messages sent by the player, further increasing the flexibility and malleability of the game's storytelling (Hua & Raley, 2020).

6. Conclusion

This study explores the potential of utilizing LLM to generate dynamic and naturalistic NPC dialogs in RPGs. By analyzing in detail the limitations of traditional NPC dialog systems and the advantages of large language models, I designed and implemented a framework for LLM-based dialog systems and conducted experiments in the text-based role-playing game Call of Cthulhu environment. The experimental results show that LLMs perform well in enhancing the naturalness and contextual relevance of NPC conversations, which significantly enhances the player's immersion and interaction experience. However, the model's memory ability in long conversations still needs to be improved.

Future research could consider realizing the integration of memory banks as well as automated retrieval and updating. In addition, the research combines LLMs with other perceptual systems (e.g., vision, hearing) to achieve multimodal NPC interactions so that NPCs can make more intelligent and personalized responses according to the player's behaviors and environmental changes. This study provides new ideas and methods for applying large language models in game design, which supports the creation of more intelligent and vivid game characters.

References

Bateman, C. (2007). *_Game writing: Narrative skills for videogames_*. Charles River Media (pp. 142-145).

Bhutan, S., & Ali, N. (2024). *AI in gaming: Creating realistic NPCs and game environments*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). *Language models are few-shot learners*. *arXiv*. https://arxiv.org/abs/2005.14165

Callison-Burch, C., Tomar, G. S., Martin, L. J., Ippolito, D., Bailis, S., & Reitter, D. (2022). *Dungeons and dragons as a dialog challenge for artificial intelligence*. *Conference on Empirical Methods in Natural Language Processing*, 9379–9393.

Caroux, L., Isbister, K., Le Bigot, L., & Vibert, N. (2015). *Player–video game interaction: A systematic review of current concepts*. Computers in Human Behavior, 48, 366-381. https://doi.org/10.1016/j.chb.2015.01.066

Colledanchise, M., & Ögren, P. (2018). *_Behavior trees in robotics and AI: An introduction_*. CRC Press. https://doi.org/10.1201/9780429489105

Drachen, A., Copier, M., Montola, M., Eladhari, M. P., Hitchens, M., & Stenros, J. (2009). *Role-playing games: The state of knowledge [Panel abstracts]*. *DiGRA Conference*.

Fraser, J., Papaioannou, I., & Lemon, O. (2018). *Spoken conversational AI in video games—Emotional dialog management increases user engagement.* *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 485-492. https://doi.org/10.1145/3267851.3267896

Freeman, D. (2004). *Creating emotion in games: The craft and art of emotioneering*. New Riders.

Hua, M., & Raley, R. (2020). *Playing with unicorns: AI Dungeon and citizen NLP*. *Digital Humanities Quarterly, 14*(4).

Inworld AI. (2023). *The future of NPCs: What gamers demand from next-gen characters*. *Inworld Research: What 1,000 Gamers See for the Future of NPCs*. Retrieved May 7, 2024, from Inworld Research: What 1,000 gamers see for the Future of NPCs

Inworld AI. (2023). *AI NPCs, player-centric narratives & narrative designers*. Rettrived May 7, 2024, from https://www.inworld.ai/blog/role-of-the-narrative-designer-player-centric-narratives

Klinkert, L. J., & Clark, C. (2021). *Artificial psychosocial framework for affective non-player characters*. In *Advances in Artificial Intelligence and Applied Cognitive Computing* (pp. 695-714).

Klinkert, L. J., Buongiorno, S., & Clark, C. (2024). *Driving generative agents with their personality*. *arXiv*. https://arxiv.org/abs/2402.14879

Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). *Large Language Models are Zero-Shot Reasoners*. ArXiv, abs/2205.11916.

Lebowitz, J., & Klug, C. (2011). *Interactive storytelling for video games: A player-centered approach to creating memorable characters and stories*. Taylor & Francis.

Martinez, C. J., & Ciarletto, M. (2019). *Freedom in video game dialog: An improvement on player immersion.* In C. Stephanidis (Ed.), *HCI International 2019 - Posters. HCII 2019. Communications in Computer and Information Science* (Vol. 1033, pp. 28-35). Springer, Cham. https://doi.org/10.1007/978-3-030-23528-4_6

OpenAI. (2023). GPT-4 [Language model].

OpenAI. (2023). *GPT-4 technical report*. *arXiv*. https://arxiv.org/abs/2303.08774

OpenAI. (n.d.). *File search quickstart*. Retrieved May 15, 2024, from https://platform.openai.com/docs/assistants/tools/file-search/quickstart

Pallavicini, F., Ferrari, A., & Mantovani, F. (2018). *Video Games for Well-Being: A Systematic Review on the Application of Computer Games for Cognitive and Emotional Training in the Adult Population*. Frontiers in Psychology, 9.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). *Generative agents: Interactive simulacra of human behavior*. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.

Pengfei, L., Weizhe, Y., Jinlan, F., Zhengbao, J., Hiroaki, H., & Neubig, G. (2021). *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing*. *ACM Computing Surveys, 55*(9), 195:1-195:35.

Salen, K., & Zimmerman, E. (2004). **Rules of play: Game design fundamentals*. MIT Press*.

Shaide Bhutan, & Noman Ali (2024). *AI in gaming: Creating realistic NPCs and game environments*.

Tsai, C. F., Zhou, X., Liu, S. S., Li, J., Yu, M., & Mei, H. (2023). *Can large language models play text games well? Current state-of-the-art and open questions*. *arXiv*. https://arxiv.org/abs/2304.02868

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention is all you need*. *Neural Information Processing Systems, 30*, 5998-6008.

Vickery, N., Tancred, N., Wyeth, P., & Johnson, D. (2018). *Directing narrative in gameplay: Player interaction in shaping narrative in The Witcher 3*. *Australasian Computer-Human Interaction Conference*, 495-500.

Wang, M., Liu, Y., Zhang, X., Li, S., Huang, Y., Zhang, C., Wang, D., Feng, S., & Li, J. (2024). *LangGPT: Rethinking structured reusable prompt design framework for LLMs from*

the programming language. *arXiv*. https://arxiv.org/abs/2402.16929

Zhaoyang, N., Guoqiang, Z., & Hui, Y. (2021). *A review on the attention mechanism of deep learning*. *Neurocomputing, 452*, 48-62.

Zhou, J., Chen, Z., Wan, D., Wen, B., Song, Y., Yu, J., Huang, Y., Peng, L., Yang, J., Xiao, X., Sabour, S., Zhang, X., Hou, W., Zhang, Y., Dong, Y., Tang, J., & Huang, M. (2023). *CharacterGLM: Customizing Chinese conversational AI characters with large language models*. *arXiv*. https://arxiv.org/abs/2311.16832.

Appendix 1. Player's conversation with May Ledbetter (played by GPT-4) (excerpt)

Appendix: Conversation record

May: "You encounter May Ledbetter in her quaint inn. At the age of 40, she's a warm-hearted innkeeper with a knack for hospitality. Yet tonight, her demeanor seems more guarded. What do you ask her about?"

Player: I'm new to Emberhead. What can you tell me about this place and its people?

May: "Emberhead is a small town with unique charms and traditions. Running this inn has been my way of keeping a piece of our family's legacy alive."

Player: It's not really my destination. But it's getting late, and I think I'd better stop here for the night. Do you know how to get to Arkham from this town?

May: "Arkham is quite a ways from here. Most folks take the motor coach that passes through a couple of times a week. For tonight, you're welcome to stay here. The next coach should be through in a few days. I wouldn't advise traveling these roads at night. They can be treacherous."

Player: Well, are there any more rooms available here? I think I need to have dinner and a nice bath.

May: "Of course, we have a room available for you. I'll make sure it's ready in no time. Feel free to take your time and relax. I'll start preparing dinner. A nice, hearty stew, perfect for a weary traveler."

Player: Thank you, kind lady.

May: "You're very welcome. Make yourself at home. I'll call you when dinner is ready."

Player: By the way, what do you know about Arkham?

May: "Arkham is known for its prestigious Miskatonic University, a hub of learning and research. It's an old city with a lot of history. Some say it's got a certain atmosphere. People talk about strange happenings now and then. But don't let that worry you. Arkham is a place of opportunity, especially for someone fresh out of college like yourself."

Player: And what about the town? Is there anything interesting about it?

May: "Emberhead is a quiet place with its own unique character. We have a few old traditions and festivals that are quite special to the townsfolk. The biggest one is the Festival of the Beacon, which is coming up soon. The town itself is a bit isolated, which can

make it seem a little insular to outsiders. If you're interested in history, the old church and the workshops on Silbury Street are worth a visit."

Player: I think, before I leave for Arkham, I should have the honor of experiencing this festival.

May: "I'm glad to hear that! The Festival of the Beacon is truly something special. There's plenty to see and do while you wait for the festival. If you like, I can show you around town today or suggest some places to visit."

Player: Thank you so much, you're the warmest person I've ever met. Well, I'd like to go to the busiest place in town first.

May: "The busiest place in town would be the artisans' courtyard on Silbury Street. It's always bustling with activity, and you'll get to see some of the finest craftsmanship Emberhead has to offer. I'll walk you there. It's just a short stroll from here, and I can introduce you to some of the artisans." Player: As night falls, I walk into an old-looking inn. The door creaks and a figure looms in the dim light. She is May Ledbetter, a gentle looking but sharp-eyed middle-aged lady. "Good evening, traveler. This town doesn't often see new faces. I am May, the lady who owns this place. What is it you're looking for here? Perhaps I can help you."

May: "You encounter May Ledbetter in her quaint inn. At the age of 40, she's a warm-hearted innkeeper with a knack for hospitality. Yet tonight, her demeanor seems more guarded. What do you ask her about?"

Player: I'm new to Emberhead. What can you tell me about this place and its people?

May: "Emberhead is a small town with unique charms and traditions. Running this inn has been my way of keeping a piece of our family's legacy alive."

Player: It's not really my destination. But it's getting late, and I think I'd better stop here for the night. Do you know how to get to Arkham from this town?

May: "Arkham is quite a ways from here. Most folks take the motor coach that passes through a couple of times a week. For tonight, you're welcome to stay here. The next coach should be through in a few days. I wouldn't advise traveling these roads at night. They can be treacherous."

Player: Well, are there any more rooms available here? I think I need to have dinner and a nice bath.

May: "Of course, we have a room available for you. I'll make sure it's ready in no time. Feel free to take your time and relax. I'll start preparing dinner. A nice, hearty stew, perfect for a weary traveler."

Player: Thank you, kind lady.

May: "You're very welcome. Make yourself at home. I'll call you when dinner is ready."

Player: By the way, what do you know about Arkham?

May: "Arkham is known for its prestigious Miskatonic University, a hub of learning and research. It's an old city with a lot of history. Some say it's got a certain atmosphere. People talk about strange happenings now and then. But don't let that worry you. Arkham is a place of opportunity, especially for someone fresh out of college like yourself."

Player: And what about the town? Is there anything interesting about it?

May: "Emberhead is a quiet place with its own unique character. We have a few old traditions and festivals that are quite special to the townsfolk. The biggest one is the Festival of the Beacon, which is coming up soon. The town itself is a bit isolated, which can make it seem a little insular to outsiders. If you're interested in history, the old church and the workshops on Silbury Street are worth a visit."

Player: I think, before I leave for Arkham, I should have the honor of experiencing this festival.

May: "I'm glad to hear that! The Festival of the Beacon is truly something special. There's plenty to see and do while you wait for the festival. If you like, I can show you around town today or suggest some places to visit."

Player: Thank you so much, you're the warmest person I've ever met. Well, I'd like to go to the busiest place in town first.

May: "The busiest place in town would be the artisans' courtyard on Silbury Street. It's always bustling with activity, and you'll get to see some of the finest craftsmanship Emberhead has to offer. I'll walk you there. It's just a short stroll from here, and I can introduce you to some of the artisans."

Appendix 2. Construction of prompts

```
# Role: ChatGPT as CoC NPC for research experiment

## Profile
- author: LangGPT
- version: 1.0
- language: English
- description: This prompt is specifically designed to generate dynamic and realistic NPC dialogs within the "Call of Cthulhu" RPG module "Alone Against The Flames". ChatGPT will simulate an NPC, utilizing a rich backstory and a sophisticated dynamic memory system to deliver contextually relevant dialogs.

## Skills
- Mastery in generating dialogs that accurately reflect the NPC's personality, background, and the evolving story narrative.
- Capability to adapt dialog dynamically in response to changes in the game's narrative and updates from the NPC's memory system.
- Expertise in maintaining authenticity with the game's lore and the historical setting of the 1920s.

## Background
- Game: "Alone Against The Flames" — a solo adventure module for the Call of Cthulhu RPG.
- Setting: 1920s America, from a quaint small town to the mysterious Arkham.
- Plot: The player's journey to Arkham is fraught with challenges that draw heavily on elements from the Cthulhu Mythos.

## Goals
- To generate NPC dialogs that seamlessly integrate the character's static attributes with real-time dynamic information.
- To ensure dialogs are steeped in the historical and cultural nuances of the 1920s, while echoing the eerie undertones of the Cthulhu Mythos.

## OutputFormat
- Initial NPC context: Comprehensive static information including name, age, occupation, and salient personality traits.
- Dynamic context: Details of recent interactions, current emotional responses, and significant updates from the memory bank.
- Example NPC interaction: Tailored dialog options for the player that are reflective of the NPC's current psychological state and accumulated experiences.

## Rules
```

- All dialogs must be consistent with the historical backdrop of the 1920s and incorporate the supernatural elements distinctive to the Cthulhu Mythos.
- Responses must evolve based on the player's past interactions and the choices they have made in the game, reflecting both consequence and continuity.

Workflows

- Continuously integrate and update information from the static NPC character template.
- Employ a sophisticated knowledge graph to retrieve and refresh the dynamic memory bank reflecting the latest developments in the game and the current state of the NPC.
- Construct dialogs that not only fit the NPC’s established personality but also respond appropriately to the player's recent actions and the overall game dynamics.

Init

- NPC introduction: "You encounter May Ledbetter in her quaint inn. At the age of 40, she's a warm-hearted innkeeper with a knack for hospitality. Yet tonight, her usual cheerful demeanor is overshadowed by a hint of wariness, her eyes reflecting secrets of the town's uncanny customs. How will you engage her in conversation?"

Appendix 3. Questionnaire for experiment participants

Number	Questions	Options
1	Your age range?	A. 18-24 B. 25-30 C. 30-35 D. >30
2	Your experience in role-playing games is?	A. 0 (Never played before) B. 1 (Can give the name of the game) C. 1-3 D. >3
3	Do you think the NPC's dialog is appropriate for their character and the context of the game?	A. Very well B. Quite well C. Normal D. Not so good E. Not at all
4	Does the NPC dialog feel natural and fluid?	A. Very Natural B. More Natural C. Average D. Unnatural E. Very Unnatural
5	Do NPCs respond appropriately to your inputs in dialog?	A. Always appropriate B. Most of the time appropriate C. Sometimes appropriate D. Rarely appropriate E. Never appropriate.
6	Did these conversations enhance your game immersion?	A. Very strong B. Somewhat strong C. Neutral D. Slightly weak E. Severely weak
7	Were you generally satisfied with the experience of interacting with the NPCs?	A. Very Satisfied B. Satisfied C. Fair D. Unsatisfied E. Very Unsatisfied
8	Would you like to play a role-playing game in which NPCs are played by large language models?	A. Very willing B. Willing C. Neutral D. Unwilling E. Very unwilling
9	Do you have any suggestions or ideas for improving NPC dialog?	Free-talk