Universiteit Leiden

Master Computer Science

[The Socio-Cognitive NPC Interaction Ladder]

Name:	Brandon Kroes
Student ID:	s3329879
Date:	[11/07/2025]
Specialisation:	Artificial Intelligence
1st supervisor:	[dr. Marcello A. Gómez-Maureira]
2nd supervisor:	[dr. Peter van der Putten]

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Large Language Models (LLMs) are advanced artificial intelligence systems trained on extensive text-based datasets to predict and generate sequences of symbols based on a specified input prompt. In the last few years, LLMs have become more popular because they have a property that is similar to intelligence, such as the ability to reason, understand context, and adapt quickly. These abilities have proven to be intriguing to utilize in video games.

In video games, Non-Player Characters (NPCs) are characters with whom players can interact. In the past, these interactions have been limited to pre-scripted interactions. With the advent of LLMs, these interactions can now be dynamically generated to allow for interactive storytelling and immersion. However, the practical challenges of implementing LLM-based NPCs remain largely underexplored. This thesis examines the technical, design, and socio-cognitive complexities of using locally run LLMs to create more dynamic, responsive, and immersive NPC interactions.

To facilitate this research, the Socio-Cognitive NPC Interaction Ladder (SCNIL) is introduced, a framework that categorizes LLM-driven NPC interactions across ten levels of complexity, from basic text-based exchanges to fully multimodal interactions. A series of prototypes were developed to explore these levels, evaluating how NPCs can integrate memory, context awareness, multimodal inputs, and emergent communication while operating within the computational constraints of local hardware.

The study identifies key technical and design challenges, including LLM accuracy in relation to computational overhead, the difficulty of maintaining long-term NPC memory, and the trade-offs between free-form generative responses and structured game design constraints. Through iterative prototyping, a set of design principles is created to help integrate LLM-based NPCs. These principles address issues like response consistency, interaction fluidity, player agency, and game world coherence.

While LLM-based NPCs offer new opportunities for player immersion and emergent storytelling, they require significant infrastructure and contextual management to be viable in real-world game development. The findings suggest that specialized LLMs for games, rather than generic ones, are essential for scalable, top-notch NPC interactions.

This research provides a structured framework, experimental insights, and practical guidelines for game developers seeking to implement LLM-driven NPCs. It also clarifies the present limitations of LLM technology in gaming and outlines prospective avenues for enhancing NPC believability, scalability, and socio-cognitive engagement.

Contents

List of Figures

Chapter 1

Introduction

Large Language Models (LLMs) are advanced artificial intelligence systems trained on extensive text-based datasets to predict and generate sequences of symbols based on a specified input prompt. While often associated with producing human-readable text due to the popularity of chat-based programs such as ChatGPT. LLMs are equally capable of generating structured outputs, including programming code or domain-specific languages. Their rise to popularity can be attributed to an emergent property that resembles intelligence, demonstrating capabilities like reasoning, contextual understanding, and dynamic adaptation.

LLMs' capacity to generate coherent, contextually relevant, and quality content [1] makes them capable of enhancing immersion in video games. Non-player character (NPC) dialogues that were once limited by pre-scripted interactions, can now be made immersive by generating complex responses with an LLM [2]. This opens new possibilities for creating more engaging and immersive video games.

Early research has explored the use of LLMs for generating interactive stories [3], controlling in-game actions, and enhancing procedural content generation [4], though there is no formal set of design principles. This absence leaves video game developers without standardized guidelines, resulting in inconsistent implementations and varying levels of player immersion.

The aim of this thesis is to develop design principles for integrating locally run LLM-based NPC interactions into video games. While there is existing research detailing the implementation of LLMs by modifying or adapting prompts to be applicable in video games[4–6], there have been few attempts to formalize or highlight design principles or a theoretical framework. Furthermore, research on player interaction with dynamic LLMs, outside of prompts, is scarce.

One of the primary challenges in deploying LLMs for game interactions is the computational power required for processing [7]. Most LLMs are run on cloud-based services, which offer the advantage of high computational capacity, allowing for complex and vast LLM models[8]. However, cloud-based solutions introduce significant disadvantages, notably latency[8]. While minor delays may be tolerable for background processing, they are detrimental to real-time NPC interactions, where latency can break player immersion[2]. A 15+ second delay for a simple response from an NPC risks disrupting the fluidity and engagement of gameplay[2]. Latency is by no means the only disadvantage, as pricing[9], availability and scalability are all contributing factors. These challenges underscore the importance of developing design principles for locally runnable LLMs, which offer lower latency and greater control over resource allocation.

This thesis focuses on the creation of prototypes to explore the challenges with LLM-based NPC interactions that are immersive, responsive, and contextually consistent, while remaining feasible to run locally. Prototypes are developed to create examples that can be experimented with to ensure real-world applicability. These prototypes and the analysis of their development cycle will then culminate in a formal set of design principles that was previously missing.

The remainder of this thesis discusses how we get to those design principles. In the next chapter, we discuss the background in which we detail the foundational knowledge of LLMs, NPCs and socio-cognitive interactions. This is followed by a discussion of the research objectives in chapter three, where the deliverables and research questions are outlined. Chapter four introduces the theoretical framework (SCNIL) that provides a structured path for the development of NPC interactions. Chapter five details the methodology, in which we discuss the research methodology used for the integration of locally run LLM-based NPC interactions. In chapter six, we describe the experimental setups with which we evaluated the SCNIL framework. Chapter seven details the prototypes of the experimental setups, in which we discuss how each prototype works and what it is meant to evaluate. This chapter is followed by the eighth chapter, in which we present the results of our experiments. Based on those results, we present a series of design principles in chapter nine. The tenth chapter is centred around the discussion in which we reflect on discoveries, implications, and analytical insights. The eleventh chapter is centred around potential future works based on our new insights. Finally, Chapter 12 concludes the thesis by summarizing our process, answering the research questions and our concluding thoughts.

Chapter 2

Background

This chapter provides the necessary context for understanding the integration of LLMs into video games. Particularly, we focus on the process of enhancing NPC interactions. We first introduce LLMs and their capabilities, followed by detailing the role of NPCs in creating immersive game experiences. Next we will explore the potential for LLM-driven NPCs to create dynamic, engaging gameplay experiences. Finally, we discuss the socio-cognitive aspects of player-NPC interactions, which we will build upon during the later stages of the thesis for the development of both the prototypes and the design principles.

2.1 LLMs

During the introduction, we briefly discussed what LLMs are, we defined them as advanced artificial intelligence systems trained on extensive text-based datasets to predict and generate sequences of symbols based on a specified input prompt. Their usability can be attributed to emergent properties that resemble human intelligence, demonstrating capabilities such as reasoning, contextual understanding, and dynamic adaptation. Communication with LLMs is done via a "prompt". A "prompt" refers to the input or query provided by a user to guide an AI model in generating a response [10]. A prompt is sent to the LLM, which will then provide a response, which is usually a generated sequence of symbols. There are two output types of interest for this thesis:

- Unstructured output/Human-readable text: This is the type of output that is directly consumed by the user. For instance, a user asks the LLM for the height of the Eiffel Tower, which results in: "The height of the Eiffel Tower is approximately 330 meters (1,083 feet), including its antennas at the top. The tower itself, without

the antennas, stands at about three hundred meters (984 feet).” The assumption of this text is that the player directly receives the output from the LLM and is the final interpreter of the information.

- Structured output/Machine-interpretable text: This is the type of output that is first interpreted by a machine. It therefore requires a machine interpretable syntax. Examples would be comma-separated values (CSV) or JavaScript object notation (JSON). Defining such an output can be done by providing the LLM with a grammar with which to adhere. For example, if one were to prompt the LLM for the height of the Eiffel Tower, with a specified grammar summer such as “height: meters” then LLM will then respond with “height:330”.

2.1.1 Relation to NPCs

LLMs are a significant advancement in artificial intelligence (AI), and they have brought with them the capacity to generate coherent, contextually relevant, and quality content that was previously not attainable. In the field of video games, these models have the potential to revolutionize NPC interactions, where they were previously limited to pre-scripted dialogues, they can now be used to create responsive, adaptive NPCs that can engage players in meaningful ways. NPCs can play a pivotal role in the player experience, as NPC interactions contribute to narrative depth and world-building. By using LLMs, NPCs can become capable of generating real-time, realistic, and narratively rooted responses, thereby increasing player immersion, and allowing players to engage with the game world in a more natural way.

2.1.2 LLM Architecture

LLMs are built upon the transformer architecture, introduced in the 2017 paper “Attention Is All You Need” [11]. A key feature of the transformer architecture is the so-called attention mechanism, which enables the model to weigh the importance of each word in a sentence relative to others. This allows LLMs to focus on contextually relevant information when generating responses, thereby producing coherent and context-sensitive output.

2.1.3 Internal functioning

The common understanding of LLMs as programs taking a query and returning an answer is a simplification and not entirely accurate. This input-output view doesn’t

fully represent their complex and probabilistic operation, which will present confusion when we discuss their limitations later on. When a user submits a query, it undergoes four main stages:

- Tokenization: A process in which the query is split into tokens, which are subwords of the original query.
- Embedding: Each token is mapped to a high-dimensional real vector via a learned embedding matrix. The purpose of this learned embedding matrix is exposing syntactic and semantic usage patterns.
- Contextual Encoding The model will then look at the vectors from the previous step and apply the self-attention mechanism, where it determines which tokens are important and to which tokens. The outcome of this mechanism is then turned into a new vector.
- Decoding (Next-Token Prediction) Finally, the model forecasts the most likely next token by turning its final vectors back into a probability list over all tokens and picks one. It repeats this forecasting-and-pick process to form a complete response.

As can be observed of this process, the model entirely relies on statistical patterns from its training corpus, rather than containing a formal method for fact-checking or validation. LLMs assign significant probability mass to tokens or phrases that are plausible in context but unsupported by reality. More on this in section 2.1.4.2.

2.1.4 Challenges

LLMs exhibit remarkable capabilities, however these powerful models are not without their downsides. The relevant flaws central to this thesis are computation limitations and hallucinations.

2.1.4.1 Computational Demands

Running LLMs requires considerable computational resources, with larger models requiring enterprise hardware and server configurations. This often necessitates the use of cloud-based solutions with high-performance hardware. As mentioned, the goal of this thesis is to establish design principles for locally run LLMs. Locally run LLMs introduce uncertainties when combined with running a computationally demanding video game,

as video games are already considered taxing on a computer system. Thereby, the challenge for this thesis is to compare the computational demands of an LLM within this constricted environment.

2.1.4.2 Hallucinations

Hallucinations refer to instances where the model generates inaccurate or fictitious information, deviating from factual knowledge and potentially providing responses that lack a basis in the model's training data [12]. As previously mentioned, the models rely on statistical patterns established during the training phase and lack a formal method of fact-checking or proofing the resulting outcome. This practice is ordinarily problematic as it means users are unable to fully rely on the output of the LLM, however the severity of this problem is amplified in video games. Players expect some degree of accuracy and honesty from the game, relying on information provided.

While the practice of lying or dishonesty to players is not rare, it is customarily applied to contribute to the intended game experience. For instance, a game in which one of the companions of the player lies to the player for the ultimate betrayal in a narrative story. The lies within the game contribute to a plot twist that can set up the final act of the game, an example of this in practice is Star Wars: Jedi Survivor[13]. Unintended false statements, however, are considerably more uncommon. Players perceive intentional false statements in video games as narrative and gameplay effects, while unintentional false statements lack meaning and are mismatched with their mental models [14]. False statements therefore result in worse, or even the breaking of, immersion and therefore undesirable.

2.1.4.3 Mitigation

Mitigation strategies for the hallucination problem have been proposed. The *paper Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* first introduced the idea of Retrieval-Augmented Generation (RAG), which is defined as a methodology that enhances LLMs by retrieving relevant document chunks from external knowledge base through semantic similarity calculation[15] Or, to put plainly, a methodology by which to force the LLM to base its generation on an external knowledge base.

The external knowledge base contains the information that is accessed via information retrieval. Information retrieval involves finding documents in a document collection that contain information relevant to the query [16]. For instance, in a video game in which the player asks information about the town they are currently in, information retrieval

will find the relevant content, such as the town's history, landmarks, and current events, and then provide that information to the LLM to form a factual, coherent and contextual response to the player's question. This process of information retrieval is intended to improve the responses of the LLM. Thereby, the challenge regarding hallucinations is that the locally run LLM models must provide factual information to players or risk a diminished game experience.

While it would be preferable to provide all the external knowledge in a query, this is inefficient from a computational sense as not all information is relevant. We therefore apply the concept of embeddings once more. Both the player query and the documents of the external knowledge base are turned into vectors. The parts of the knowledge base who's embedding lie closest to the query embedding are pulled in as they are deemed relevant to the query. These pertinent details are then appended to the model's input, allowing the LLM to generate accurate, context-aware responses without wasting computational resources on irrelevant information.

2.1.5 Local vs Cloud-based LLMs

In the introduction, the research was scoped to the applicability of local LLMs. This was because cloud-based LLMs bring their challenges that make them less suitable for real time NPC-based interactions.

2.1.5.1 Latency

Latency is a critical challenge for cloud-based LLMs. As highlighted in a recent study[2] forty percent of participants expressed frustration with delays of up to 15 seconds when interacting with cloud-based LLMs. Even minimal delays can disrupt the flow of gameplay, especially for real-time NPC interactions where responses need to be immediate to maintain immersion. At the time of writing, LLM providers are currently not providing low latency responses (¡ 2 seconds). There is a good reason for this lack of supply for low latency responses because their focus is on quality. Furthermore, most users that are using their service are doing it either via a website chat interface or a type of environment less focused on immersion, which is essential for gaming.

2.1.5.2 Pricing

Cloud-based services offer computational power, but often come at a high financial cost. Most commercial LLM providers are not yet profitable [17] and depend on venture capital

to operate, leading to fluctuating or increasing API costs over time. Local deployment, by contrast, utilizes a player's existing gaming hardware, offering a more economical solution for developers.

2.1.5.3 Availability

Cloud-based services are also susceptible to availability issues. OpenAI indicates a 99.57% uptime with Anthropic having a 99.23% uptime, both over a period of 90 days. To put this in perspective, with a 99.50% uptime, Microsoft's Azure, its cloud provider, who also provides the hardware infrastructure for OpenAI, will introduce discounts when the uptime percentage drops under 99.95% as it is considered unacceptable to have such a low uptime percentage [18]. These low levels of reported uptime can result in noticeable disruptions for players during gameplay or not being able to play at all. Furthermore, updates to cloud-based models can require constant adjustments from developers to maintain compatibility, increasing the maintenance burden. While locally run LLMs are not impervious to issues, their issues are associated with the hardware or software of an individual user, not the user-base at large.

2.1.5.4 Scalability

Scalability presents a significant challenge for cloud-based LLMs when applied to video games. Modern games can attract millions of concurrent players, each requiring real-time, complex responses from the system for extended periods (hours). We suggest a hypothetical scenario to illustrate this point more clearly: consider a game like Black Myth: Wukong, which reached a peak of 2,415,714 concurrent players [19]. Supporting LLM-driven NPC interactions for each of these players simultaneously would demand enormous computational resources. Every NPC interaction would require authentication, sending prompts to the cloud, processing it through an LLM, and returning a response with minimal delay. This process would need to occur in real-time for millions of users simultaneously. Cloud providers such as OpenAI and Anthropic lack the infrastructure to manage this level of concurrent, real-time interaction. If a single game release can overwhelm existing systems, the challenge becomes even more pronounced when considering the broader ecosystem. For instance, Steam, a platform limited to PC gaming, averages over ten million concurrent players during peak hours, excluding console platforms entirely. This highlights a critical limitation: cloud-based LLMs cannot yet scale to support widespread, real-time NPC interactions for games with large player bases. Locally run LLMs scale with the number of players as players provide their own computational resources instead of relying on a cloud provider.

2.1.5.5 Scoping

Given the challenges outlined above, this thesis focuses on locally run LLMs as a potential viable solution for enhancing NPC interactions in video games. Local models offer lower latency, greater control over resource allocation, and reduced long-term costs. However, they do come with the downside of worse LLM generation quality due to limited computation resources. Locally run LLMs need to be computationally less intensive and as local machines are not only less powerful but also need to generate an LLM response while simultaneously running a video game.

2.2 NPC Interactions

Non-player characters, NPCs, are characters in a game that are not controlled by a player, but with which players can interact as part of the game [20]. They represent a key application of human-to-computer interaction (HCI) as they allow the player to engage with the game world intuitively by using the player's human capability of social dynamics. It would therefore be incorrect to see them as merely quest-givers or objects that take space. NPCs function as narrative devices, sources of information, and social companions. Players engage with NPCs not just for gameplay progression but also to fulfil social and emotional needs, like those in human-to-human interactions. This behaviour is beneficial to the game design in virtual environments, as proper application often leads to deeper engagement and emotional connections[21].

Therefore, these beneficial social interactions, that players are already exhibiting, can serve as a basis for designing player-NPC interactions. Where previous NPCs were limited in their expression of reciprocation or emulation of these social dynamics due to developers being limited to pre-scripted dialogues, LLMs allow the game to generate appropriate responses to player interactions. Removing a limitation that has been there since the advent of NPCs in games. As we are now capable of generating these responses, we can use these socio-cognitive processes as the basis with which we design LLM-based NPC interactions.

It should therefore be noted that interactions with NPCs should not be considered merely a text-based interaction. They should not be limited to human-readable text, but preferably a structured output that the game can take into a more sophisticated and interactive manner, and should go beyond textual interactions.

2.3 Socio-Cognitive Theories

Socio-cognitive interactions combine two distinct aspects, social and cognitive processes. Social refers to those aspects of mental life that enable and are shaped by social experience. It involves understanding social relationships, interpreting the thoughts, feelings, and intentions of others, and interacting appropriately within social contexts[22]. Cognitive processes are the mental operations involved in acquiring, processing, and storing information. These include perception, memory, reasoning, decision-making, and problem-solving[23]. Based on these two definitions we can define socio-cognitive as the mental processes through which individuals interpret social cues, understand relationships, and engage in interactions that are shaped by both cognitive functions (like reasoning and memory) and social experiences (such as empathy, trust, or cooperation).

In the context of video games, socio-cognitive interactions refer to the process by which players engage with NPCs in a way that mirrors real-world social dynamics. This involves interpreting the intentions, emotions, and goals of NPCs as though they were humans. LLM-based NPC interactions can facilitate these interactions by smoothing out exiting barriers in interactions or even creating previously impossible interactions due to their capacity to generate contextually relevant dialogue and behaviour. By incorporating socio-cognitive theories like the Theory of Mind and distributed cognition, these NPCs can more accurately emulate realistic social interactions, creating a more immersive and intuitive player experience.

Players often see NPCs as sentient beings. This phenomenon arises from a cognitive bias known as anthropomorphism—the attribution of human characteristics, intentions, or behaviours to non-human entities [24]. Players instinctively apply social behaviours to NPCs: they ask questions, seek guidance, form attachments, and sometimes even exhibit empathy or moral decision-making based on their interactions.

This instinctive behaviour comes from social cognition [25], which refers to the mental processes that we use to understand and respond to social cues. Examples are perception, memory, and decision-making, all of which help humans interpret the intentions and actions of other humans. In video games, social cognition leads players to view NPC behaviour as if it were human, even when the characters are abstract, non-human, or simply serve a functional role in the game. This behaviour is beneficial to the game design in virtual environments, as proper application often leads to deeper engagement and emotional connections[21].

2.3.1 Theory of Mind

Theory of Mind (ToM) refers to the ability to understand that other individuals possess their beliefs, intentions, and emotions that may differ from one's own [22]. It is a fundamental aspect of human social interaction, allowing people to predict and interpret the behaviour of others based on their understanding of those differing mental states. In the context of video games, simulating ToM in NPCs enhances the depth and realism of interactions. An LLM-based NPC could demonstrate this capability by adapting its responses based on the player. This can be accomplished by basing its judgement on the player's actions, intentions, past choices, or statements (spoken or written). For example, an NPC might base its responses on the actions of a player and adjust its dialogue or behaviour, giving subtle clues to the player regarding their opinion of these actions. Thereby allowing the NPC to emulate their motivations within the game world. This would make the NPC feel more dynamic, believable, and relatable. Integrating ToM into NPC behaviour intensifies the immersion by allowing players to feel that the world is dynamic and shapes according to changes in it.

2.3.2 Anthropomorphism

Anthropomorphism means attributing human characteristics, intentions, and emotions to non-human entities. This includes extending human-like reasoning, emotions, and behaviours to animals, objects, or artificial agents [22]. Players tend to anthropomorphize NPCs, projecting human-like emotions, reasoning, and intentions onto them. This even happens when the NPCs are clearly artificial, and players are consciously aware of this artificiality. Game developers can leverage this tendency by creating NPCs that exhibit human-like traits, such as consistent emotional reactions or context-sensitive dialogue, regardless of visual appearance. This can lead players to form an emotional bond with NPCs that enhances immersion and engagement. When paired with LLM technology, anthropomorphism becomes even more powerful, as it introduces the ability to generate dynamic dialogue and adaptive behaviour, allowing them to emulate more lifelike, socially aware, and cognizant social interactions. The more players perceive NPCs as socio-cognitive peers, the greater the possibility that the players have a more meaningful or intuitive interaction, thereby strengthening narrative and emotional depth in the gameplay experience.

2.3.3 Distributed Cognition

Distributed Cognition is the theory that cognitive processes are not confined to an individual's mind but are shared across people, tools, and environments [26]. In video games, distributed cognition can manifest through cooperative interactions between players and NPCs. For instance, when solving a puzzle, an LLM-based NPC can offer clues, provide suggestions, or even partake in performing tasks necessary for progression. This creates a sense of comradely or co-dependence, where cognitive responsibilities are shared between the player and the game's artificial agents. Purposefully dividing required information for progression plays into this theory of distributed cognition. By recognizing the game's environment and adapting to changing conditions, LLM-based NPCs can help players achieve shared objectives, and garnering a more immersive and intuitive experience.

2.3.4 Joint Attention

Joint Attention refers to the shared focus of two individuals on an object or event, typically coordinated through gaze, gestures, or verbal cues [22]. In video games, LLM-based NPCs can emulate joint attention by responding to player actions or focusing on specific game elements in tandem with the player. For example, if a player points their character toward a particular object or location, the NPC could acknowledge this through dialogue. Join attention can therefore foster a sense of collaboration and facilitate a shared experience between the player and the NPC. By acknowledging the player's actions in real time, LLM-based NPCs can create more immersive and intuitive interactions that make the world feel more alive.

2.4 Related Work

Before diving into the core of this thesis, it is important to situate this research within the landscape of NPC design and LLMs in the wider game development industry.

2.4.1 Embodied Social Cognition into NPC Design

Foundational frameworks, such as Newell's Bands of Cognition [27], Minsky's Society of Mind [28], and Brooks' Subsumption Architecture [29], established the foundation for modelling cognition in artificial intelligence. These frameworks interpret cognition as either a layered or a modular process that is relies on symbolic information processing.

Within these models, social cognition was primarily treated as an isolated reasoning task consistent with the Computational Theory of Mind [30, 31]. By the early 1990s, however, the limitations of these approaches had become increasingly evident. Brooks' later work, emphasized that internal symbolic models often hindered robust, real-time interaction with the environment[32].

Recent works have identified other limitations in these computational approaches, particularly in the context of interactive systems like NPCs. Deshpande and Magerko [33] advocate for a shift towards embodied social cognition, emphasizing that social understanding emerges through real-time, embodied interactions rather than static mental simulations. Frameworks such as Participatory Sense making [34] and Observable Creative Sense making [35] support the notion that meaning is co-constructed through interaction, rather than pre-programmed.

Integrating embodied social cognition into NPC design allows for more adaptive and context-sensitive interactions with players. NPCs can move beyond static or purely generative dialogue, responding dynamically to player actions and game context. This perspective aligns with the objectives of this thesis, which aims to establish design principles for locally run, LLM-based NPCs that support player immersion and meaningful social experiences.

2.4.2 Player Expectations and NPC interactions

The significance of NPC responsiveness, realism, and social presence has been underscored by research on player expectations. Yin and Xiao's (2024) study on VR interactions identified that players expect lifelike social cues and non-verbal behaviours from NPCs [36]. Findings from this study emphasize the importance of NPCs as affective tools that foster immersion and attachment. Understanding these expectations is crucial for integrating NPC interactions in a way that maintains player engagement. This aligns with the thesis goal of designing NPCs that enhance player experience through context-sensitive and socially aware interactions.

2.4.3 LLMs for NPC interactions

The earliest exploration of LLMs in games began with AI Dungeon [37], built on GPT-2. AI Dungeon bypassed static dialogue trees by using LLMs to generate responses dynamically. Instead of assigning user input to pre-written branches, player prompts were passed to the LLM for context-aware dialogue. This marked a shift from static to

generative interactions, expanding dialogue possibilities. The system's success, supporting over one million users [38], demonstrated the viability of LLM-driven NPCs. This foundational work set the stage for more advanced, interactive NPC architectures.

Since the release of AI Dungeon, LLMS have substantially advanced, giving rise to increasingly sophisticated implementations. In 2024, Treanor et al. introduced *Slice of Life*, which integrates LLM-based dialogue within a social simulation system. This approach grounds prompts in game state data using an existing system for authoring playable social models [39] in combination with an LLM (Google Gemini), to ensure responses are both context-aware and socially coherent. Song (2025) expanded on this with an NPC architecture that spans in-game Unity clients and out-of-game Discord channels [40]. This cross-platform memory sharing enhances NPC persistence and strengthens player attachment. These recent advances demonstrate the expanding role of LLMs in delivering dynamic and socially integrated NPC interactions.

LLMs exhibit strengths in associative tasks, but can face challenges with complex reasoning tasks. Plaat et al. (2024) highlighted Chain-of-Thought prompting as a method for multistep reasoning [41]. While effective for structured tasks, these models often struggle with self-improvement and consistency in reasoning. This limitation affects NPCs' abilities to sustain coherent and context-sensitive dialogue. Scaffolding LLMs within game architectures can help address these challenges by grounding outputs in game state and context. Designing NPCs that leverage LLMs' reasoning strengths while mitigating their weaknesses is critical for immersive gameplay.

2.4.4 LLMs in videogames

LLMs have found applications in various areas beyond NPC interactions, including reinforcement learning [42], interactive storytelling [3], and procedural content generation [4, 43]. These applications demonstrate how LLMs can improve instruction, modify stories based on player decisions, and create organized game content in response to cues. In addition, they illustrate the adaptability of LLMs in handling a variety of game development components, including mechanics and storytelling. However, challenges remain around maintaining coherence, context, and player immersion, particularly when integrating dialogue and dynamic gameplay elements. Understanding these broader uses provides valuable insights for designing NPCs that leverage LLMs effectively. These insights can inform strategies to balance flexibility and consistency, ensuring LLM-driven NPCs contribute meaningfully to the game experience.

2.4.5 Gaps in current research

Although substantial work exists on NPC design, LLM capabilities, and socio-cognitive modelling individually, there remains a significant gap in bridging these perspectives into a coherent design framework specifically for locally run LLM-based NPC interactions. Most existing research focuses on either cloud-based LLMs or isolated aspects of social cognition without addressing the real-time, local, and resource-constrained environments typical of many games.

Chapter 3

Research Objectives

In previous chapters, the need for a socio-cognitive approach to NPC interactions has been established, highlighting how players intuitively engage with NPCs as though they were sentient beings. This chapter presents the core objectives of the research, translating the theoretical foundations into clear, actionable goals.

The focus of this thesis is to explore how locally run LLM-based NPC interactions can be designed using socio-cognitive principles to enhance immersion, intuitiveness, and narrative storytelling. This chapter outlines the research questions, defines the objectives, and sets the scope of the investigation.

3.1 Main Research Question

As outlined in the introduction, the primary research question guiding this thesis is:

> *What design principles can be established to inform the design of locally run, LLM-based NPC interactions in video games, encompassing a spectrum of socio-cognitive interaction methods?*

This question introduces several key constraints for the research:

- **Locally run LLMs:** Focusing on models that operate without relying on cloud infrastructure.
- **NPC interactions in video games:** The study is limited to non-player character interactions rather than broader game mechanics.
- **Socio-cognitive interaction methods:** The thesis draws from socio-cognitive theories to structure realistic player-NPC engagement.

The objective is to establish a set of design principles that developers can apply when creating NPCs driven by LLMs. This chapter also presents sub-research questions that help break down the complexities of the main question.

Research Sub-Questions

To address the main research question effectively, three sub-questions guide the study:

Q1: What technical and design challenges arise when employing local LLMs for NPC interactions in video games, and how can iterative prototypes effectively address these constraints?

The background chapter details the challenges associated with cloud-based LLMs, namely scalability, availability, cost, and latency. Therefore, the approach of this thesis is to focus on local LLMs. Local LLMs are not without their challenges, as local machines are more constrained in their computational power, which is likely to limit the capabilities of LLMs. By analysing the issues and challenges within the implementation of local LLMs, we gain insight into the limitations and can prototype solutions accordingly. These issues are unlikely to be unique as they relate to the implementation of local LLMs, and will therefore influence the design principles.

Q2: How can socio-cognitive interaction methodologies be translated into diverse prototype designs for local LLM-based NPCs, and what trade-offs arise regarding player engagement and system complexity?

This research question explores how socio-cognitive theories can be embedded into prototype development. Each prototype will assess distinct levels of interaction complexity. This will result in insights into how to design trade-offs, such as processing power, player engagement, and interaction depth, to impact the gameplay experience. These trade-offs will then influence the resulting design principles, detailing what is a reasonable expectation or consideration when integrating an LLM.

Q3: Which in-game knowledge representations and prompting strategies mitigate hallucinations in local LLM-based NPCs, ensuring coherence and alignment with game lore?

As mentioned in the background, LLMs suffer from hallucinations, in which they generate nonsensical responses, which can be particularly detrimental in a video game setting. Based on this significant issue, a need arises for an efficient and effective strategy to address it. The goal of this research question is to determine how game-specific information can be used to increase the quality of LLM responses. By answering this, we can derive design principles for a more coherent approach to LLM integration.

3.2 Research Deliverables

The following deliverables are intended to be developed during the research:

1. A series of design principles that detail principles to ensure consistency, coherence, and socio-cognitive realism in LLM-based NPC interactions.
2. A cohesive framework in which socio-cognitive concepts are transferred from a series of abstract descriptions to implementations.
3. A series of prototypes that implement the aforementioned framework to evaluate the framework and to provide hands-on experience with which to base the design principles on.
4. An analysis of the technical performance of locally run LLMs in a real-time game environment, focusing on latency and response quality.

5. An evaluation of player immersion, engagement, and interaction quality through observational studies and player feedback across different prototypes.

Scope and Limitations

This research focuses primarily on single-player games, where personalized NPC interactions can significantly enhance immersion. While the framework may be adaptable to multiplayer contexts, those environments are beyond the immediate scope of this study. Additionally, the reliance on locally run LLMs introduces computational limitations, which may restrict the complexity of interactions assessed.

Summary

This chapter outlined the primary goals and research questions. The main objective is to develop a series of design principles for locally run LLM-based NPC interactions in video games. The main research question focuses on integrating socio-cognitive theories into the creation of immersive, context-aware NPC interactions. Three sub-research questions were introduced to explore technical challenges, socio-cognitive design trade-offs, and strategies for mitigating LLM hallucinations. The research approach involves developing and testing prototypes, from which practical design principles will be derived.

Chapter 4

The Socio-Cognitive NPC Interaction Ladder (SCNIL)

To support the structured integration of large language model LLM-based NPC interactions in modern video game design, we have developed the SCNIL (Socio-Cognitive Narrative Interaction Levels) model. It structures these interactions into ten levels based on socio-cognitive principles. Each level builds upon the previous one, gradually enhancing both the depth of engagement between players and NPCs and the complexity of integrating the LLM-based NPC in a video game.

The model outlines ten distinct interaction levels, ranging from basic text-based interactions to multi-modal interactions. These labels are rooted in theoretical foundation previously discussed. The ten levels are designed to ensure that the NPCs not only respond coherently to player input but also emulate quasi-human-like reasoning and adaptive behaviour.

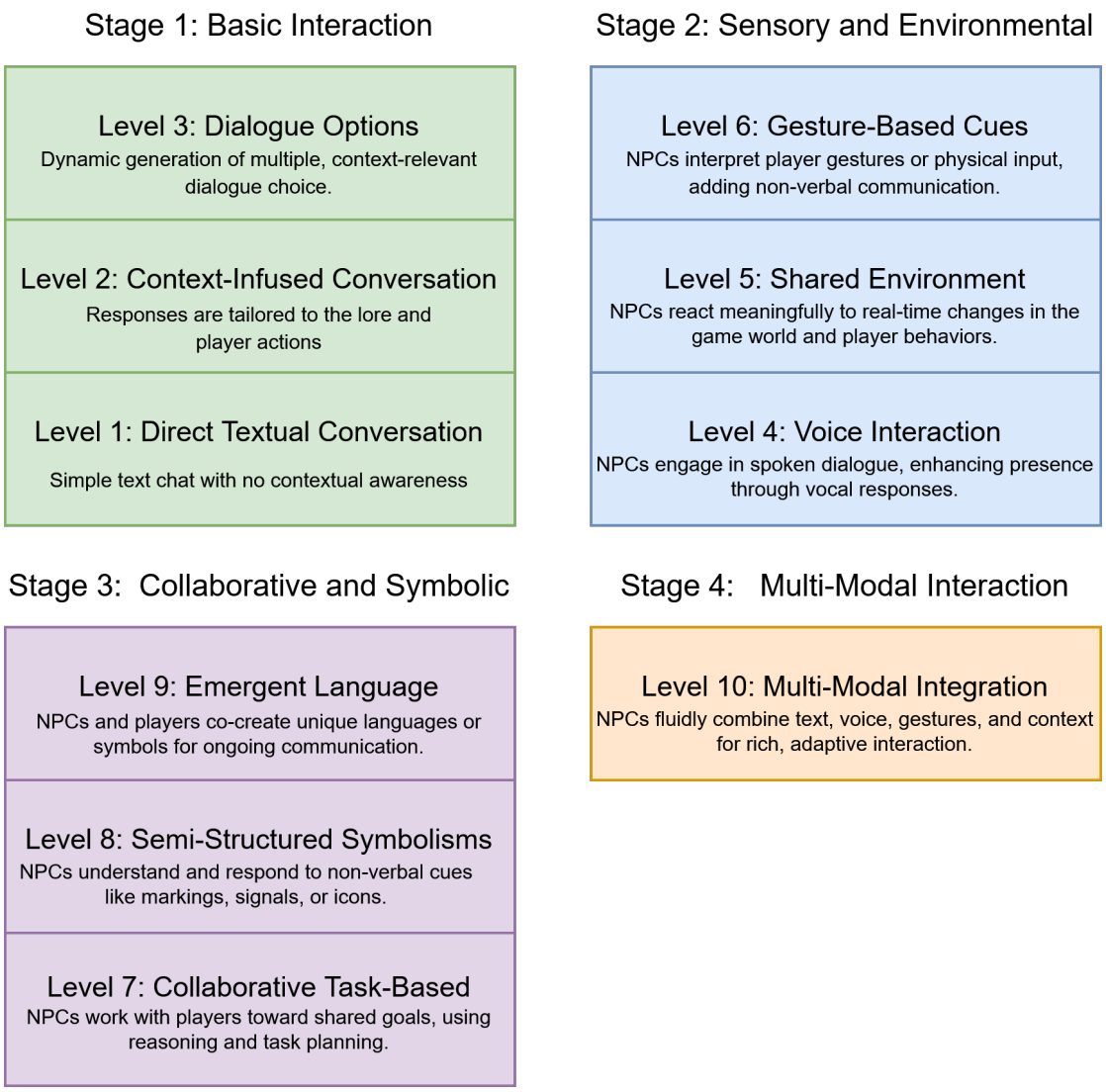

FIGURE 4.1: The ten interaction levels with their corresponding stages

The primary goal of SCNIL is to bridge the gap between traditional pre-scripted NPC interactions and dynamic, context-aware generated responses by LLMs. Existing NPC systems often rely on pre-scripted dialogue trees and rigid behaviour scripts, which limit their ability to respond flexibly to player actions and decisions.

By formalizing a progression of interaction complexity, SCNIL:

- Provides developers with a structured framework for enhancing immersion, narrative depth, and intuitive gameplay.
- Incorporates socio-cognitive principles to align with player expectations for realistic, engaging interactions within a scope of acceptability.

- Balances the technical limitations of locally run LLMs with the need for real-time response and adaptive NPC interactions.

4.1 Theoretical Foundation

SCNIL draws directly from established socio-cognitive theories that explain how players perceive and interact with NPCs:

- **Theory of Mind (ToM):** NPCs emulate an understanding of the player's intentions, adjusting interactions based on prior exchanges or in-game context.
- **Anthropomorphism:** By emulating human-like emotions and behaviours, NPC interactions become more socio-cognitive, encouraging player immersion.
- **Distributed Cognition:** Player-NPC collaborations reflect shared problem-solving dynamics, with both parties contributing to in-game objectives.
- **Joint Attention:** NPCs can consider the player's focus and respond accordingly, creating shared attention on in-game objects or areas of interest.

4.2 Prototypes as Practical Applications of SCNIL

SCNIL is structured around ten prototypes, each representing a level of socio-cognitive interaction and technical complexity. These prototypes are grouped into four stages:

4.2.1 Stage 1 — Basic Interactions (Levels 1–3)

- **Level 1: Direct Textual Conversation** — A straightforward text exchange where players interact with NPCs through unaided dialogue.
- **Level 2: Context-Infused Conversation** — The LLM draws from in-game context (e.g., lore, player progress) to generate more relevant and personalized responses.
- **Level 3: Dialogue Options** — The LLM dynamically generates dialogue choices, offering players multiple conversation paths, relying on conversation prediction and planning.

These levels relate to Social Cognition and Anthropomorphism, emphasizing basic player-NPC relationships and familiar social dynamics.

4.2.2 Stage 2 — Sensory and Environmental Interactions (Levels 4–6)

- **Level 4: Voice Interaction** — Spoken dialogue is introduced, enhancing immersion through auditory engagement.
- **Level 5: Shared Environment** — NPCs respond to environmental changes and player actions within the game world.
- **Level 6: Gesture-Based Cues** — NPCs interpret and respond to player gestures or physical inputs as meaningful interactions.

These levels relate to Joint Attention and Distributed Cognition, focusing on environmental awareness and non-verbal interactions.

4.2.3 Stage 3 — Collaborative and Symbolic Interactions (Levels 7–9)

- **Level 7: Collaborative Task-Based** — NPCs and players share tasks and responsibilities, enabling cooperative problem-solving.
- **Level 8: Semi-Structured Symbolisms (3S)** — NPCs interpret non-verbal symbols or cues (e.g., drawings, environmental signals) to communicate meaning.
- **Level 9: Emergent Language** — Players and NPCs co-develop a shared language or symbol system to support deeper interaction.

These levels focus on Theory of Mind and Distributed Cognition, emphasizing shared meaning-making and dynamic collaboration.

4.2.4 Stage 4 — Multi-Modal Interaction (Level 10)

- **Level 10: Multi-Modal Integration** — This level integrates the previous interaction modes (text, gestures, environmental cues, and voice) into a cohesive system. NPCs dynamically switch between modalities based on context, maximizing socio-cognitive realism and responsiveness.

This final stage represents the culmination of all socio-cognitive theories from Levels 1 through 9.

4.3 Scope of SCNIL

Video games vary widely in their design, goals, and mechanics. While single-player games emphasize narrative immersion and emotional engagement, multiplayer games prioritize balance, competition, and scalability.

SCNIL is best suited for single-player experiences, where personalized NPC interactions enhance immersion and story depth. Story-driven games particularly benefit from SCNIL's emphasis on context-aware, dynamic social interactions.

4.4 Applicability

Although this research focuses on single-player games, the socio-cognitive principles of SCNIL are flexible and adaptable. The framework can be extended to various genres and game structures, providing a theoretical foundation for future research into advanced NPC design.

4.5 Summary

This chapter introduced the Socio-Cognitive NPC Interaction Ladder (SCNIL), a framework for designing increasingly complex and immersive LLM-based NPC interactions. Drawing from socio-cognitive theories such as Theory of Mind, Anthropomorphism, Distributed Cognition, and Joint Attention, SCNIL outlines ten interaction levels that gradually increase narrative depth and gameplay realism.

Each level builds upon the previous, progressing from basic textual exchanges to complex, multi-modal interactions, and offers a structured pathway for designing NPCs that feel believable, responsive, and context-aware.

Chapter 5

Methodology

This chapter details the research methodology used for the integration of locally run LLM-based NPC interactions in video games. It outlines the research design, prototype development process, evaluation criteria, and data collection methods used to assess the SCNIL framework.

A prototype-driven approach was selected to determine the real-world applicability of LLM-based NPC interactions. Socio-cognitive interactions in gameplay are complex, and therefore iterative testing and refinement are essential to evaluate player immersion, gameplay coherence, and technical feasibility.

5.1 Research Design

This research follows an experimental, prototype-based approach to explore LLM-based NPC interactions. Using the SCNIL model as a structured framework, ten prototypes are developed and assessed.

To evaluate the SCNIL model and the developed prototypes, this research adopts a self-testing methodology, where the researcher serves as the primary participant. While this introduces significant bias concerns regarding objectivity of the participant experience, a concerted effort will be made to mitigate these through objective evaluation metrics and predefined test cases. Thereby allowing the study to be transparent and lowering the bar for a reproducibility of the test results. Besides the player experience, we also evaluate the performance testing and compare them to theoretical baselines.

While large-scale user testing would preferable, as they establish a more reliable evidence-based evaluation, that evaluation would be beyond the scope of this thesis by sheer

magnitude. The coordination of evaluating ten prototypes per participant, with an adequate sample size, which requires measuring complex and subjective metrics such as immersion, coherence, and believability are complicated and worthy of their study. Taking proper testing standards, such as controllable environments, repeatability, and systematic evaluation, makes an analysis even more complicated.

At the time of writing, there are also no standardized metrics or methodologies for evaluating LLM-based NPC interactions. While analysing the conversational performance of LLM-based NPCs is not entirely new[44], existing methodologies are not applicable to SCNIL as they are not focused on the socio-cognitive aspects that SCNIL is built upon.

Given these challenges, a self-testing approach provides a controlled, repeatable, and systematic way to evaluate LLM-based NPC interactions without the logistical and computational difficulties of large-scale user testing. Furthermore, any implementation into a video game will be different and will entirely depend on the game in which the SCNIL framework is implemented in. A video game that is not immersive due to game design or mechanic issues will not become immersive by merely implementing SCNIL.

5.2 Prototype Development Process

The study develops ten prototypes, each corresponding to a different SCNIL level. These prototypes are assessed sequentially, with each iteration refining insights into LLM-based NPC behaviour. The prototypes are categorized into four primary stages:

1. **Basic Interactions (Levels 1–3):** Text-based communication, context integration, and dynamic dialogue options.
2. **Sensory and Environmental Interactions (Levels 4–6):** Voice interaction, shared environments, and gesture-based communication.
3. **Collaborative and Symbolic Interactions (Levels 7–9):** Task-based NPC collaboration, emergent language, and semi-structured symbolisms.
4. **Multi-Modal Interaction (Level 10):** Integration of all previous levels into a unified system.

Each prototype is assessed using predefined player inputs, allowing for controlled response analysis.

5.3 Development Tools and Technologies

5.3.1 System Architecture

The system architecture for the SCNIL prototypes consists of key components working together to process player input, generate LLM-based NPC responses, and maintain interaction coherence. The core modules include:

- **Input Processing Module**
  - Captures player input via text, voice, or gestures.
  - Converts non-text input (e.g., speech, gestures) into structured text for LLM processing.
- **NPC Processing Module**
  - Queries the LLM using context-aware prompts.
  - Retrieves relevant in-game knowledge via RAG before generating a response.
  - Retrieves additional current environmental knowledge.
- **Output Generation Module**
  - Converts LLM-generated responses into appropriate formats (e.g., text display, synthesized speech).
  - Manages the NPC animations or environmental interactions based on the response.
- **Performance Monitoring Module**
  - Tracks latency, memory usage, and computational load.
  - Logs all LLM-based NPC interactions for post-analysis.

The following technologies are used to develop and test the prototypes:

5.3.2 Godot 4.4

The prototypes were developed in the Godot game engine. Godot is a free and open source community-driven 2D and 3D game engine [45]. Godot was picked as it was both free and familiar to develop in. The prototypes were originally developed in version 4.3 and were updated to 4.4 as this version became available. The code of the project can be found on GitHub. The prototypes rely on community-made add-ons:

- **PhantomCamera**: A Camera Add-on that allows for simplified 3D camera control available under the MIT licence made by ramokz
- **TerraBrush**: A tool with which to create height maps available under the MIT licence made by spimortdev.
- **Holiday Kit**: A series of free 3D assets available under the Creative Commons CC0 licence made by Kenney.

5.3.3 LLM Integration

The LLM integration was built on-top of *llama.cpp*. *llama.cpp* is a tool that allows developers to interface with LLMs while maintaining state-of-the-art performance on local machines [46]. Two different models were picked for evaluation: TinyLlama and Deepseek-R1.

5.3.3.1 TinyLlama

TinyLlama is an open-source small-scale language model designed to enable end-user applications on mobile devices, and serve as a lightweight platform [47]. TinyLlama also explicitly references its capability to enable real-time dialogue generation in video games [48]. The specific version used was *TinyLlama-1.1B-v1.1* which consists 1.1 billion parameters and *Q4_K_M* quantization. TinyLlama was picked for its fast generation capabilities as this was likely a desirable outcome on computers that will be running a video game simultaneously.

5.3.3.2 Deepseek-R1

DeepSeek-R1 is an open-source LLM model based on the *Chain of Thought* principle capable of generating prompt responses, capable of comparable performance of cutting end closed source LLMs [49]. DeepSeek-R1 also highlights its capacity for complex role-playing[49]. The specific version used was *Deepseek-R1 1.5b*. Deepseek-R1 ran with 1.5 billion parameters with Q5_K_M quantisation. Deepseek-R1 was picked to highlight a heavier LLM to determine if the generation capabilities of smaller models was a limiting factor.

5.3.4 Retrieval-Augmented Generation (RAG):

RAG was implemented to reduce hallucinations by injecting game-specific context into prompts. The RAG was implemented as a stand-alone tool that receives requests from the game via HTTP and passes the appropriate context with the request to the LLM. The data used to provide a reference game lore was an export of the Unofficial Elder Scrolls Pages. This export was picked as it was a large, easily accessible and quality source. The data was embedded using the all-MiniLM-L6-V2 model and embeddings were stored in a Chroma database. *all-MiniLM-L6-V2* uses 384-dimensional vectors, chunk size was set at 500 tokens with a chunk overlap of 100, with the Top-K set at 3. Hybrid match was selected as there is a lot of overlap due to repeated content in game lore. The RAG implementation was done in Python 3.12.

5.3.5 Level-specific tools

For certain levels, specific tools were required. These tools were as followed:

- Level 4: For speech-to-text, we used Whisper by OpenAI.
- Level 4: To convert text to speech, we used the Godot built-in functionality.
- Level 8: For recognising and aiding in the visualisation of symbols, we used *Q* Super-Quick Recognizer.

5.3.6 Pipeline

The pipeline linked all of these tools together using HTTP POST requests. HTTP was picked over other methods, such as a command line interface or web sockets. This was

primarily to allow fine-tuning of the pipeline, swapping other tools or other maintenance without requiring the entire pipeline to be redeveloped. For instance, *Ollama*, an alternative LLM interfacing tool, was initially picked instead of *llama.cpp*. *Ollama* caused repeated issues with AMD ROCm, for hardware acceleration, and was therefore replaced. Figure 5.1 showcases the order of the pipeline.

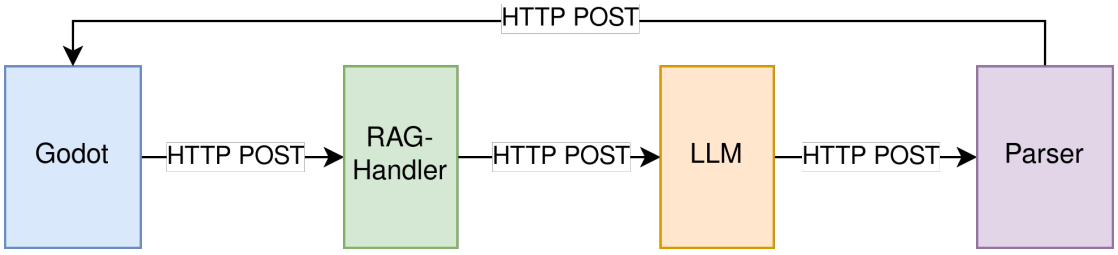

FIGURE 5.1: Pipeline order

5.4 Evaluation Criteria and Metrics

Since no external participants are involved, NPC interactions are evaluated using quantitative and automated measures:

5.4.1 Independent Variable

- **SCNIL Level:** interaction complexity.

5.4.2 Dependent Variables

- **Response Coherence:** Measured by logical consistency and adherence to in-game lore. The response coherence is graded on a series of three possibilities, pass, lacking or incoherent. Pass implies that a response is considered appropriate, lacking implies that the response shows signs of logical consistency and/or adherence to in game lore, but fails to convince while incoherent implies that it fails this consistency and adherence standard.
- **Memory Retention:** Assessed using predefined memory recall tasks.
- **Computational Performance:** Evaluated through:
  - Latency
  - Memory usage
  - Processing load

- **Tokens Per Second (TPS):** A metric with which to measure the speed of an LLM. Tokens are parts of a word at a ratio of 100 tokens being approximately 75 words [50]. There are four levels of ratings towards the TPS measure based on the reading speed of the average adult in English for fictional words [51].
  - < 3 TPS (≈ 133 wpm): *Insufficient*
  - 3–5.3 TPS ($\approx$ 134–238 wpm): *Passable*
  - 5.3–8.9 TPS ($\approx$ 239–400 wpm): *Acceptable*
  - > 8.9 TPS (> 400 wpm): *Perceivably instant*

5.4.3 External Factors

We note LLM randomness and world-complexity as covariates.

5.4.4 Automated Response Logging

- All LLM-generated NPC responses are stored in logs including input prompts, output text, and generation time.

5.4.5 System Performance Testing

- CPU/GPU load, memory consumption, and latency are tracked across SCNIL levels.
- The feasibility of real-time execution on common local hardware is also assessed.

5.5 Ethical Considerations

Although this study does not involve external participants, ethical guidelines are followed:

- **Transparency in Bias and Limitations:** Acknowledging subjectivity in self-testing.
- **Replicability:** All prompts and logs are made available to enable reproducibility.

5.6 Limitations of the Methodology

Despite offering a structured and repeatable approach, this methodology has several limitations:

- **Self-testing Bias:** Results may reflect subjective interpretation, despite objective metrics.
- **Lack of Player Diversity:** No insights into varied player behaviour or psychology.
- **Game-Specific Factors:** SCNIL effectiveness may vary across game types and mechanics.
- **Scalability Assessment:** A more demanding game environment may be necessary to test LLM capacity boundaries.

To mitigate these issues, the study emphasizes quantitative analysis and full transparency for future replication.

5.7 Summary

This chapter outlined the experimental, prototype-based methodology used to assess LLM-based NPC interactions. Ten prototypes were developed based on SCNIL levels, ranging from simple textual interaction to complex multi-modal engagement. The research relies on self-testing, predefined prompts, and automated data collection(Tokens per second). The next chapter presents the results and their implications for deriving design principles.

Chapter 6

Prototyping and Experimental Setup

The previous chapter detailed the methodological framework and evaluation criteria. This chapter builds upon that by detailing the technical implementation of the SCNIL prototypes and the experimental environment. As mentioned in the methodology chapter, large-scale user testing is not feasible within the scope of this study. All prototypes are assessed using a controlled and self-tested methodology.

6.1 Prototyping Approach

Developing LLM-based NPCs introduces challenges in coherence, responsiveness, and memory retention. Instead of creating a single, complex LLM-based NPC, we chose to develop an LLM-based NPC for each of the layers within the SCNIL model. This ensures both a quality LLM-based NPC integration and practical feasibility. The express purpose of the prototypes is to develop a minimal example for each level. Each game will implement LLMs differently as they require varying levels of RAG, game integration and have different performance expectations. Therefore, these minimal examples will be developed not to create a studio-quality implementation, but to create a generalised experience that can result in design principles that are widely applicable.

6.2 SCNIL Prototype levels

The SCNIL model is structured into ten levels, with each level representing an increasing complex socio-cognitive interaction or technical implementation. These levels are categorized into four stages. In a video game, the level of interaction between a player and an NPC is different, requiring different levels to express these interactions. Companion NPCs in the game spend more time with the player, whilst a player might not even interact with other NPCs. The following section will describe each stage.

6.2.1 Stage 1: Basic Interactions (1–3)

This stage represents the fundamental capabilities of LLM-based NPCs, putting the emphasis on text-based interactions without any additional environmental awareness or multimodal input.

- **Level 1: Direct Textual Conversation.** Baseline free-form chat with no game context—serves to assess raw LLM performance (relates to anthropomorphism).
- **Level 2: Context-Infused Conversation**
- **Level 2: Context-Infused Conversation.** The LLM-based NPC has access to game-specific context (player progress, world lore, relationships) via RAG, enabling responses that infer player intent (aligned with Theory of Mind).
- **Level 3: Structured Dialogue Options.** Instead of free-form chat, the LLM-based NPC presents a set of response choices, reducing unpredictability of user input while retaining generative depth. Player selections influence subsequent options, invoking Theory of Mind by inferring intent and guiding conversational strategy.

  The basic interactions stage focuses on, as the name implies, basic LLM functionality in a game setting, including response coherence, memory integration, and structured interaction and lacks game world integration.

6.2.2 Stage 2: Sensory & Environmental Interactions (4–6)

This stage introduces multimodal capabilities, allowing the LLM-based NPC to process spoken language, gestures, and environmental awareness to create a more socio-cognitive complex interaction. Furthermore, the technical implementations become significantly more complex than the previous stage.

- **Level 4: Voice Interaction.** The LLM-based NPC processes player speech via speech-to-text, generates text responses, and delivers them through text-to-speech, enabling voice-driven dialogue. This mode engages Joint-Attention theory by fostering shared focus between player and NPC.
- **Level 5: Shared Environment Awareness**
  At this level, the LLM-based NPC reacts to environmental changes, including object interactions, locations, and in-game events. This allows for context-sensitive dialogue, where the LLM-based NPC can dynamically acknowledge its surroundings. Level 4 already started the alignment with Joint Attention, but level 5 goes further, as it now also requires the NPC to acknowledge external objects that both the player and NPC are aware of, resulting in a sense of shared space leading to a more meaningful interaction.
- **Level 6: Gesture-Based Cues**
  Level six enables LLM-based NPCs to interpret and respond to non-verbal player input. Examples include pointing at objects, nods/shakes for yes/no answers, or emote-based communication. This is directly tied to Joint Attention and Distributed Cognition, as gesture-based cues introduce non-verbal communication and shared intent understanding, mimicking real-world human interactions.

This stage focuses on testing multimodal interactions and LLM-based NPC awareness of game environments, moving beyond pure text-based communication.

6.2.3 Stage 3: Collaborative & Symbolic Interactions (7–9)

This stage introduces shared problem-solving, symbolic communication, and cooperative decision-making, requiring the LLM-based NPC to interpret player intent dynamically.

- **Level 7: Task-Based Collaboration**
  LLM-based NPCs assist in gameplay mechanics, such as solving puzzles, exploring areas, or strategizing. This requires the LLM to track player objectives, prioritize tasks, and adapt its suggestions based on game progress. This aligns with Distributed Cognition, where knowledge and decision-making are no longer confined to the player but instead distributed across player-NPC collaboration, mimicking real-world cooperative problem-solving.
- **Level 8: Semi-Structured Symbolisms (3S)**
  The LLM-based NPC gains the ability to interpret abstract symbols or semi-structured communication forms. The player might draw a basic symbol,

and the LLM-based NPC must attempt to interpret its meaning in the game world. This is grounded in Anthropomorphism and Distributed Cognition, as NPCs are now emulating a human-like symbolic reasoning processes.

- **Level 9: Emergent Language**
  In this level, the player and LLM-based NPC co-develop a shared lexicon, the LLM-based NPC learns non-standard terms introduced by the player. This dynamic vocabulary-building mirrors real-world language emergence, where two individuals create a personalized communication style over time. This is aligned with Theory of Mind, as the LLM-based NPC infers the meaning of symbols in real-time, and Distributed Cognition, as language construction is a collaborative process.

This stage focuses on evaluating LLM-based NPCs in cooperative gameplay, abstract reasoning, and negotiation of meaning.

6.2.4 Stage 4: Multi-Modal Interaction (10)

The final SCNIL stage integrates all previous interaction types into a single, cohesive LLM-based NPC system, where the LLM-based NPC can fluidly switch between text, speech, environmental awareness, gestures, and emergent communication methods.

- **Level 10: Integrated Socio-Cognitive NPCs**
  LLM-based NPCs at this level seamlessly combine all prior capabilities, dynamically selecting the most appropriate interaction mode based on context. For example, an NPC might react to an object in the game world (Level 5), respond verbally (Level 4), and use shared symbolic references developed over time (Level 9). For this level, the applicable socio-cognitive theories all are all in play.

This stage focuses on evaluating comprehensive, multimodal LLM-based NPC interaction, ensuring natural transitions between different communication styles.

6.3 Development and Testing Process

Each prototype was developed incrementally, with each level being assessed individually before higher levels were developed. This ensured:

1. Technical feasibility: preventing performance bottlenecks before adding complexity.

2. Coherent progression: testing if the foundational interactions where dependable before integrating multimodal capabilities.
3. Comparative evaluation: measuring if higher SCNIL levels improve interaction quality over simpler implementations.

The next section will detail the implementation of each prototype, including game engine integration, LLM selection, and system architecture.

6.4 Experimental Setup and Testing Conditions

6.4.1 Testing Environment

The experiments were conducted in a controlled game simulation environment, where the SCNIL LLM-based NPC interacts with the player under a set predefined condition. Each test session follows a structured sequence to assess the LLM-based NPC's ability to:

1. Process different input modalities (text, speech, gestures, environment).
2. Maintain coherence and memory across interactions.
3. Respond dynamically to environmental changes for the later levels.
4. Adapt behaviour based on emergent communication patterns for the later levels.

Each SCNIL level is assessed independently.

6.4.2 Hardware

The following hardware will be used for testing:

Type	**Name**	**Notes**
CPU	AMD RYZEN 7 7800X3D	45W-TDP mode - iGPU disabled
RAM	Corsair Vengeance 64 GB	5600MT/s – 2 DIMM
GPU	NVIDIA GeForce RTX 5070 Ti	16 GB – GDDR7 – CUDA enabled
SSD	Kingston SKC3000D2048G	7 GB/s Rated Write – Read — 2048 GB (1.9 TB)

6.4.3 Structured Test Cases

To ensure consistent evaluation across different SCNIL levels, a series of predefined test cases was created. All the test cases are judged based on the previously mentioned 3-point criteria: Pass, lacking or incoherent. These test cases are as follows:

- **Response Accuracy** – Does the NPC correctly interpret and respond to the player's input?
- **Memory Retention** – Can the NPC recall past interactions and apply them meaningfully?
- **Environmental Awareness** – Can the NPC recognize and reference objects in the world?
- **Adaptive Interaction** – Does the NPC choose the most contextually appropriate interaction mode?

6.4.4 Data Logging and Collection

All LLM-based NPC interactions are automatically logged in an external JSON file to ensure objective evaluation and eliminate reliance on subjective impressions. The logging system records:

- **Player Input**: The exact text, speech, or gesture used.
- **NPC Response**: The LLM-generated output in raw form.
- **Processing Time**: The time taken from player input to NPC response.
- **Resource Usage**: CPU, memory, and GPU load during processing.

6.4.5 Reference hardware

Before we begin with the experimentation of the prototypes, we first perform a benchmark to determine the computation hardware available. This was done to give a baseline of comparison and to detail what the LLMs are capable of in an unrestricted environment. The benchmark is done with the built-in benchmark tool in llama.cpp (llama-bench.exe) Both models, TinyLlama and Deepseek, will be benchmarked on both the CPU and GPU. The metrics that will be recorded are the prompt processing and text generation.

6.4.6 Challenges and Mitigation Strategies

6.4.6.1 Computational Constraints

LLMs for real-time NPC interactions require significant computational resources. The later stage levels of SCNIL, particularly those involving multimodal input processing, increase the computational demand. Latency may also affect player experience, making real-time interaction difficult. There are two steps being taken to mitigate these issues as best as possible:

- Using optimized, quantized models to reduce memory footprint.
- Conducting profiling to determine the optimal batch sizes and token limits for faster inference.

6.4.6.2 Maintaining Long-Term Memory and Context

Higher SCNIL levels require LLM-based NPCs to recall past interactions, learned vocabulary, and environmental context over extended gameplay sessions. LLMs, however, have a limited context window, meaning earlier interactions may be forgotten. There are three ways to address potential issues:

- Explore the possibility to store interaction logs in an external memory system (e.g., vector databases or in-game event logs).
- Use summarization techniques to condense previous interactions while retaining key details.
- Implement a state tracking mechanism where LLM-based NPCs maintain a structured representation of past conversations and actions.

6.4.6.3 Balancing Flexibility with Game Design Constraints

While LLMs enable dynamic, free-form NPC interactions, excessive flexibility may conflict with game design intentions. Developers typically require some level of control to ensure interactions remain within narrative and gameplay constraints. There are three ways to address potential issues:

- Use guided prompting techniques to keep responses aligned with game mechanics.
- Allow designers to set interaction boundaries by defining permissible dialogue structures.
- Incorporate fallback scripted responses for cases where the LLM-generated content deviates too far from the intended experience.

6.4.7 Summary

This chapter outlined the development and evaluation of the SCNIL prototypes using a controlled self-testing methodology. Since large-scale user testing was not feasible, predefined prompts, automated response logging, and performance benchmarking were used to evaluate the performance. The SCNIL model was divided into four stages, progressing from basic text-based interactions to complex

multi-modal communication. Each prototype was developed incrementally, integrating an LLM, retrieval-augmented generation, and speech-to-text processing. Structured test cases evaluated the response accuracy, memory retention, and the environmental awareness. All LLM-based NPC interaction data was automatically logged for analysis. Key challenges, including computational constraints and memory limitations, were addressed through model optimization and external memory storage.

Chapter 7

Prototypes

This chapter describes how each level was developed, what the player will experience and what the flow of the program is with a corresponding flowchart.

7.1 Level 1: Direct Textual Conversation

This is the most literal and basic implementation of an LLM. It defines the interaction merely via an unaided textual conversation, setting the player as the full controller of the interaction. This means, a player asks a message and the LLM response is directly transferred to the player. This also means that the quality of the prompt can vary and that the level of integration in the game world is minimal. The direct textual conversation should therefore be viewed as merely an evaluation of the prompt pipeline because it lacks any game-specific knowledge or integration. The demo features the player walking up to the LLM-based NPC, typing a message, and receiving a response. The model requires no memory retention, as each response is generated independently. It looks as followed:

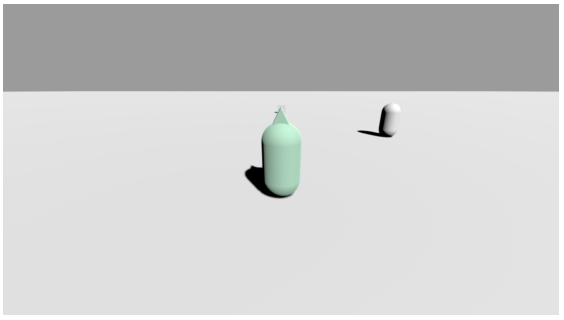

FIGURE 7.1: Step 1: The start of the game, the player wanders around

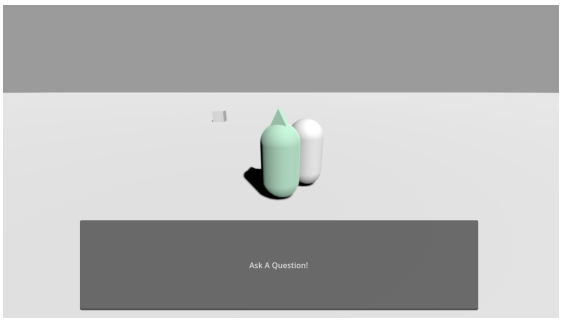

FIGURE 7.2: Step 2: Upon approaching the NPC, a text prompt appears

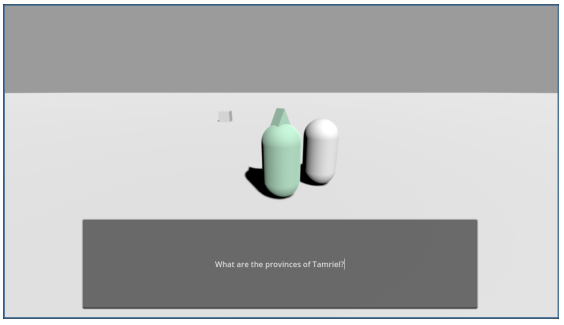

FIGURE 7.3: Step 3: The player submits a query

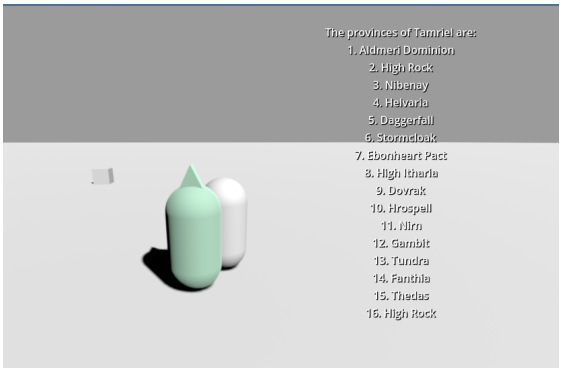

FIGURE 7.4: Step 4: Response

Note that in the example, the LLM returns sixteen regions. The actual lore contains nine, thereby making this an example of the limitations of the evaluation. In this first level, the game loop shown in figure 7.5 can be seen.

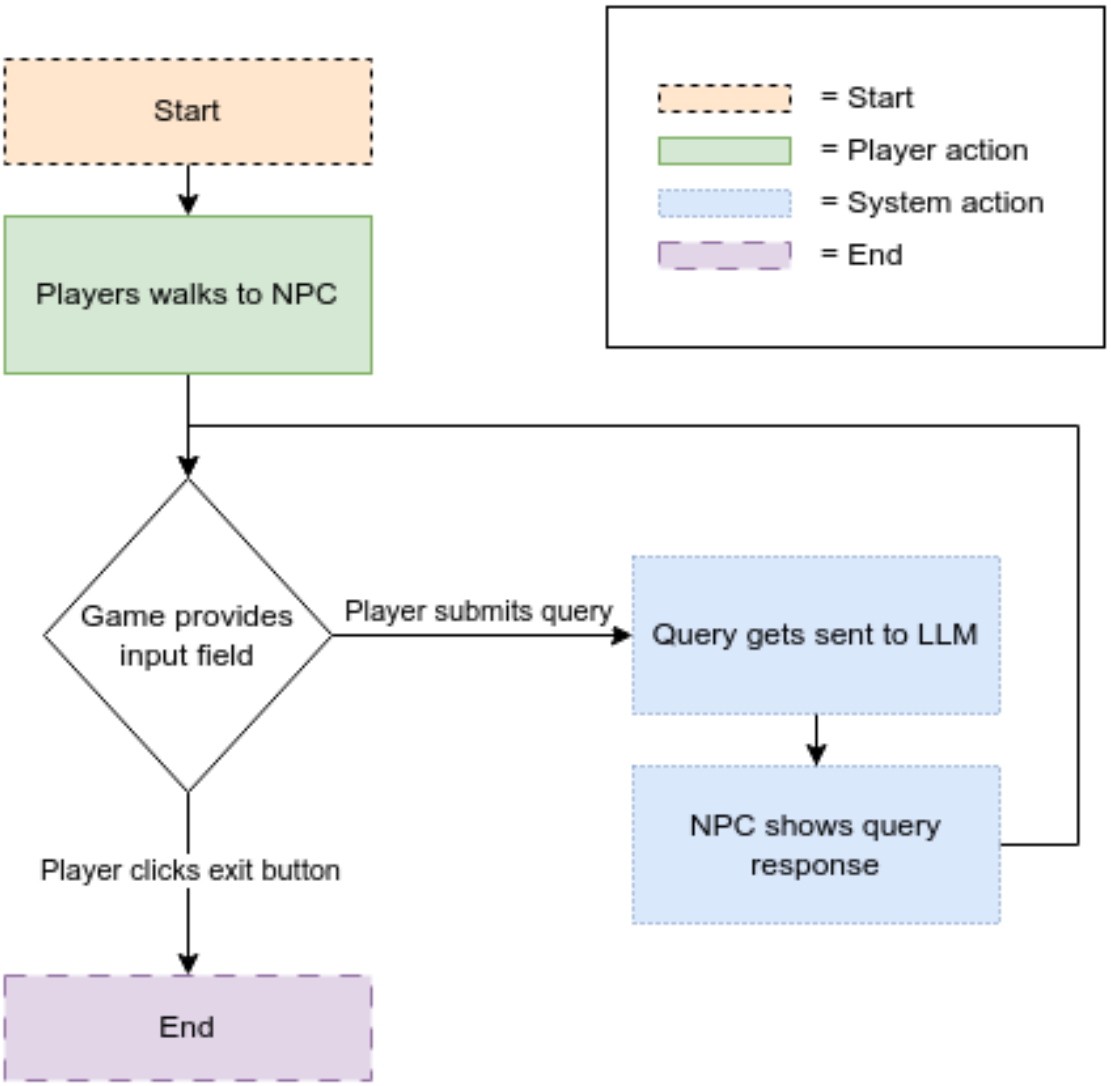

FIGURE 7.5: Level 1: Game-loop

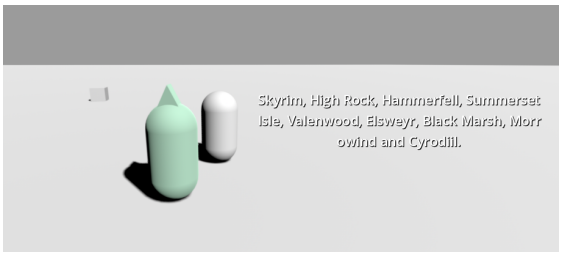

FIGURE 7.6: RAG-based Response

7.2 Level 2: Context-Infused Conversation

Much like the direct textual conversation, the interaction entirely consists of text, but it differentiates by what the LLM bases their knowledge on. This was achieved by implementing RAG. A context-infused conversation allows the LLM to access game specific information when computing the output of a prompt. To store all the relevant information, an in-memory database was developed that is read and sent to the LLM when prompting. The demo is like the previous level, the players walk up to an LLM-based NPC, asks a question, but contrary to the previous level, the memory retention ensures a conversation arises between the player and the LLM-based NPC. The difference between the previous level and the current level can be clearly seen in figure 7.6, as in this example, the correct provinces are shown based upon the RAG database.

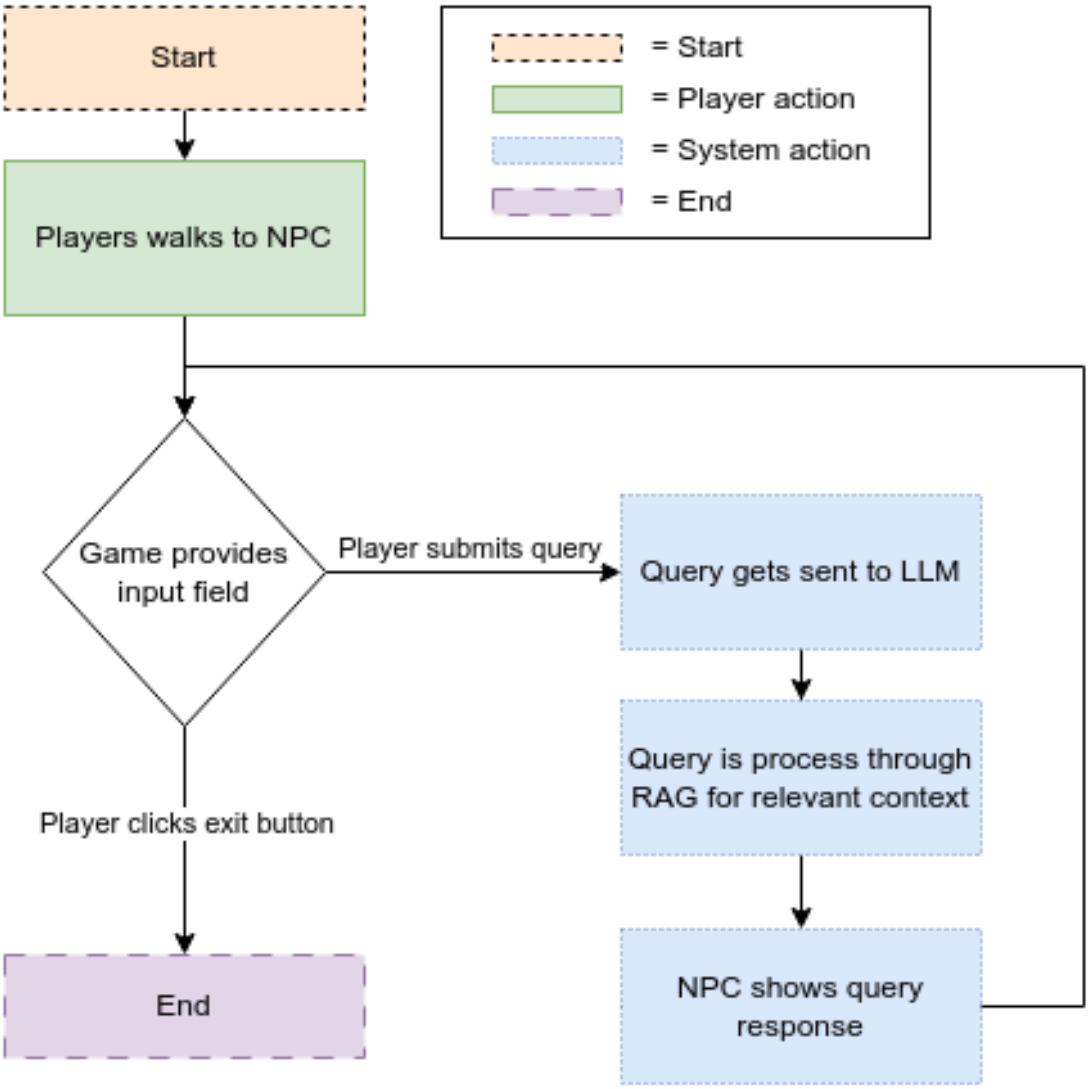

FIGURE 7.7: Level 2: Game-loop

7.3 Level 3: Dialogue Options

In this level, the LLM provides a series of dialogue options that the player can pick. These dialogue options steer the conversation and allow the player to guide the course of the conversation. This implies a more advanced level of integration in the game wherein a certain level of options is generated based on the experience of the player, effectively predicting avenues of conversation with the player. Important is that the game developer does not create pre-programmed outputs but provides a frame of reference that the LLM will use to base its suggestions on. The advantage of this LLM interaction is that conversations can be more tailored by the player input and narrative storytelling. With the disadvantages being that the LLM can suggest dialogue options that the player had no intention of asking. Like the

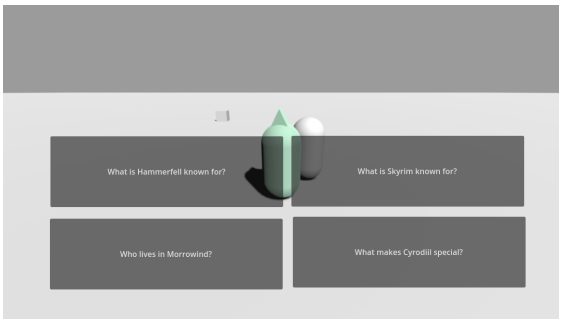

FIGURE 7.8: An example of the dialogue options

previous two levels, the player walks up to an LLM-based NPC, gets prompted for a series of possibilities, selects the relevant one, and continues in a conversation.

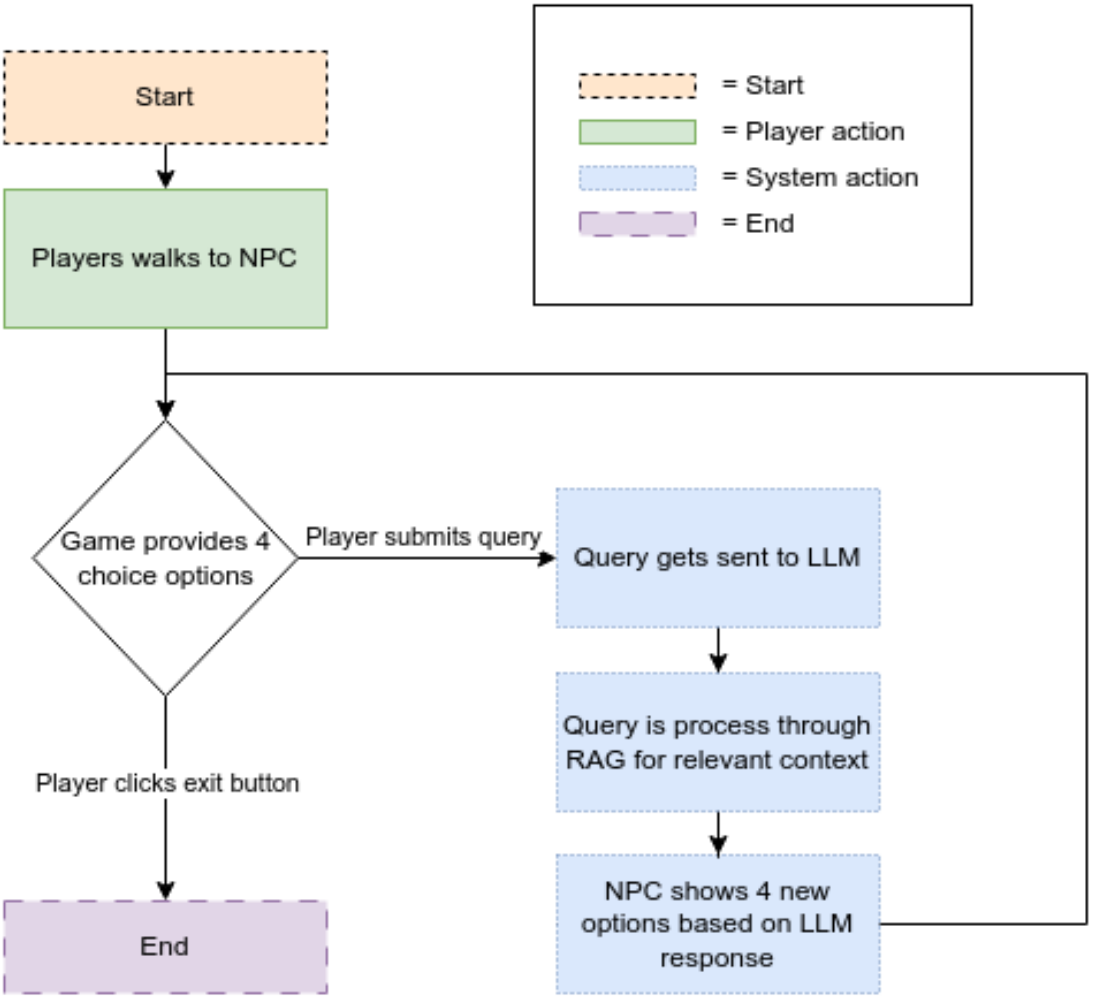

FIGURE 7.9: Level 3: Game-loop

7.4 Level 4: Voice Interaction

While the process of generating a voice interaction might seem technically easier than one of a dialogue option, it requires a more challenging integration with the game world. Contrary to earlier levels, the pipeline no longer stops at the generation of a response in a textual message, but is now also concerned with the shape and form of that message. Speech and writing are two different mediums of communication. In the demo, the player walks up to the LLM-based NPC, presses the record button, asks their question, waits for the response, listens to the response, and replies. In this example, prompts would be similar to level 2.

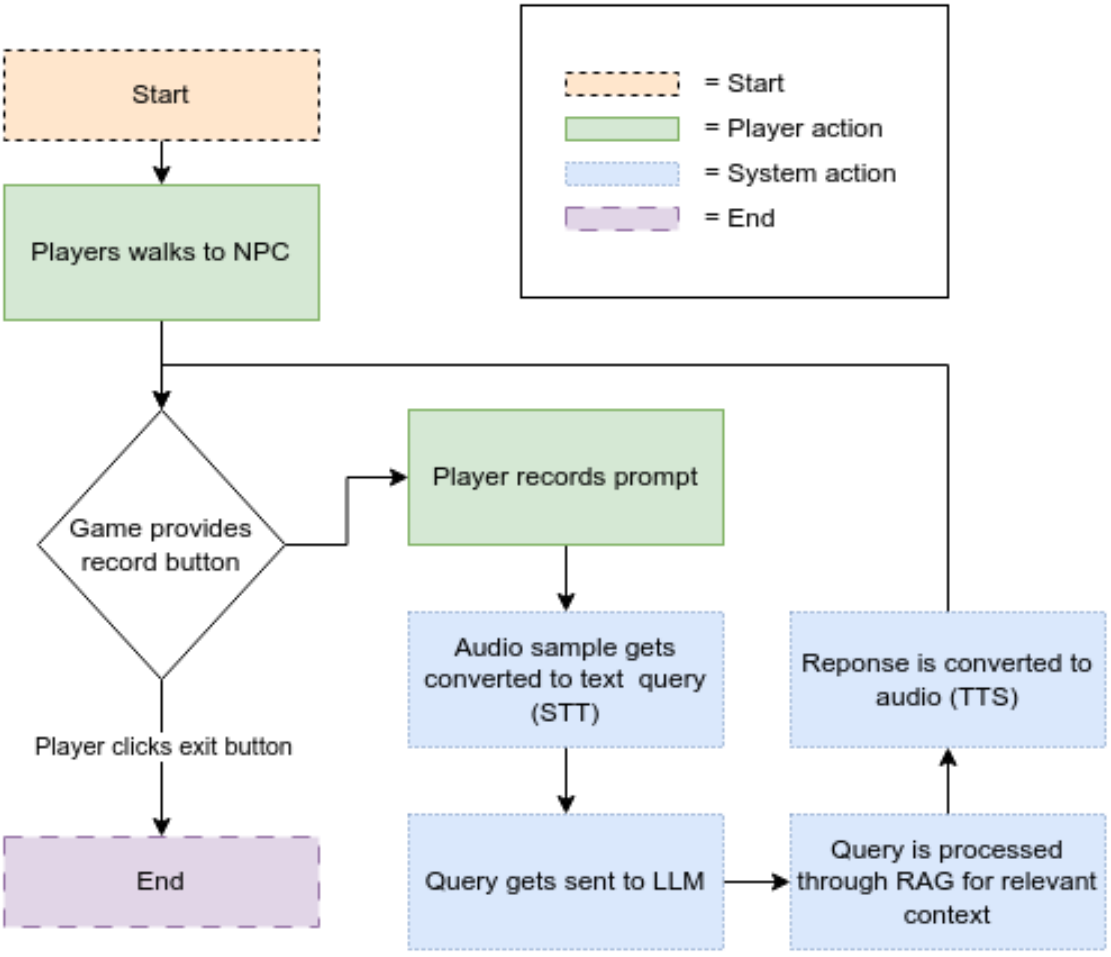

FIGURE 7.10: Level 4: Game-loop

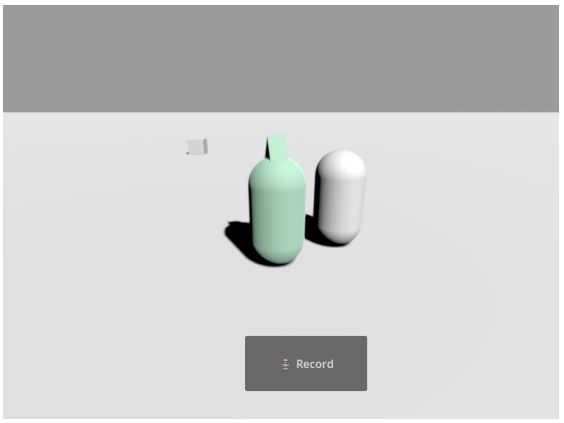

FIGURE 7.11: When approaching an NPC, the player is prompted to start the recording

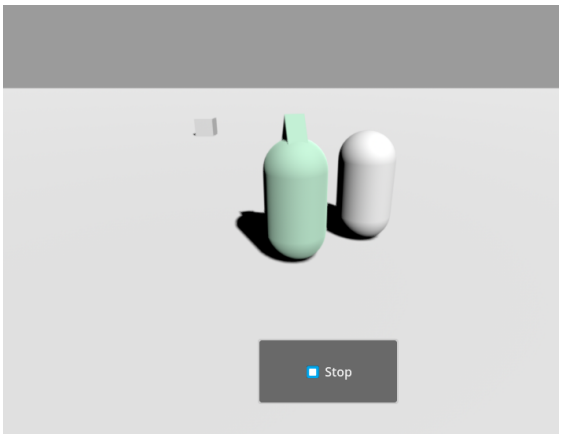

FIGURE 7.12: Once the player is recording a message, they then get the option to stop it, at which point it will be processed by the LLM

7.5 Level 5: Shared Environment

In the shared environment, the LLM-based NPC gains an understanding of the environment. This also means that environmental changes are logged to ensure a temporal memory. In this demo, the environment is filled with a series of coloured cubes, the player can ask the LLM-based NPCs about the cubes. The prompt will then be appended with the environment information of the videogame. When the world changes, the LLM-based NPC will reflect these world changes.

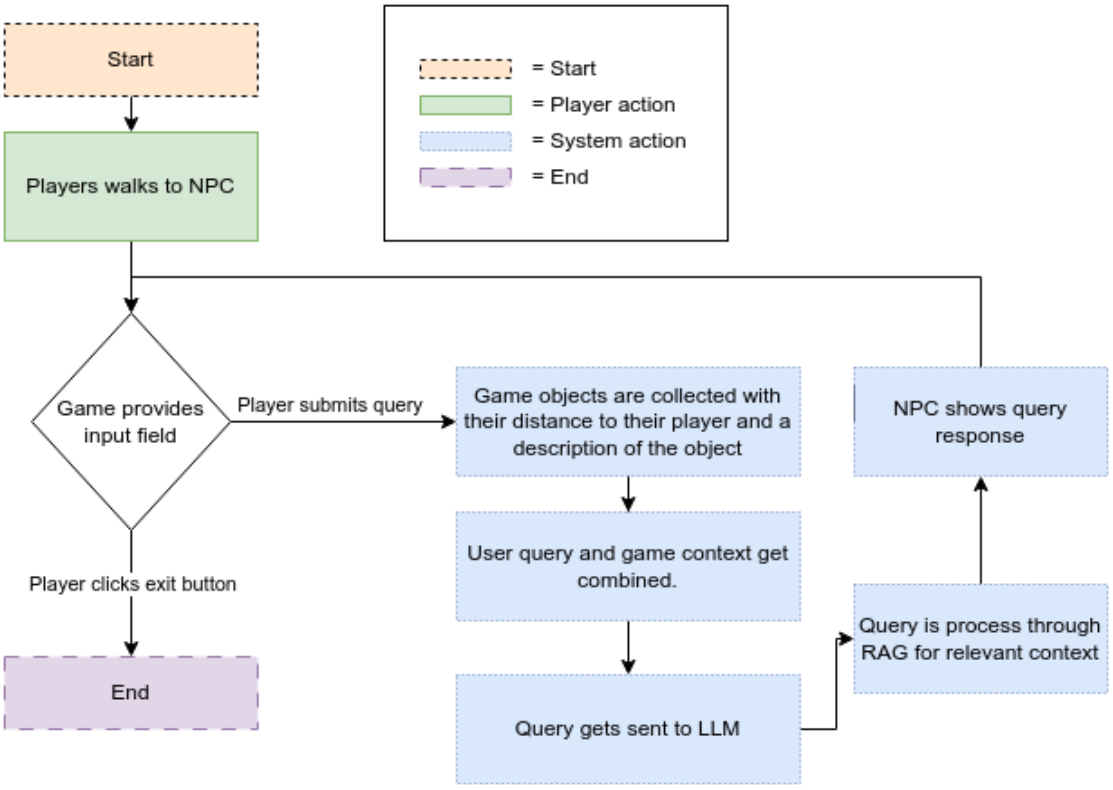

FIGURE 7.13: Level 5: Game-loop

7.6 Level 6: Gesture-Based Cues

In the gesture-based cues, the player gets to choose a series of buttons that emulate gestures on the player character that the LLM is provided with when processing a prompt. In the demo, the player walks up to the LLM-based NPC, asks a question while using the buttons to provide non-verbal inputs.

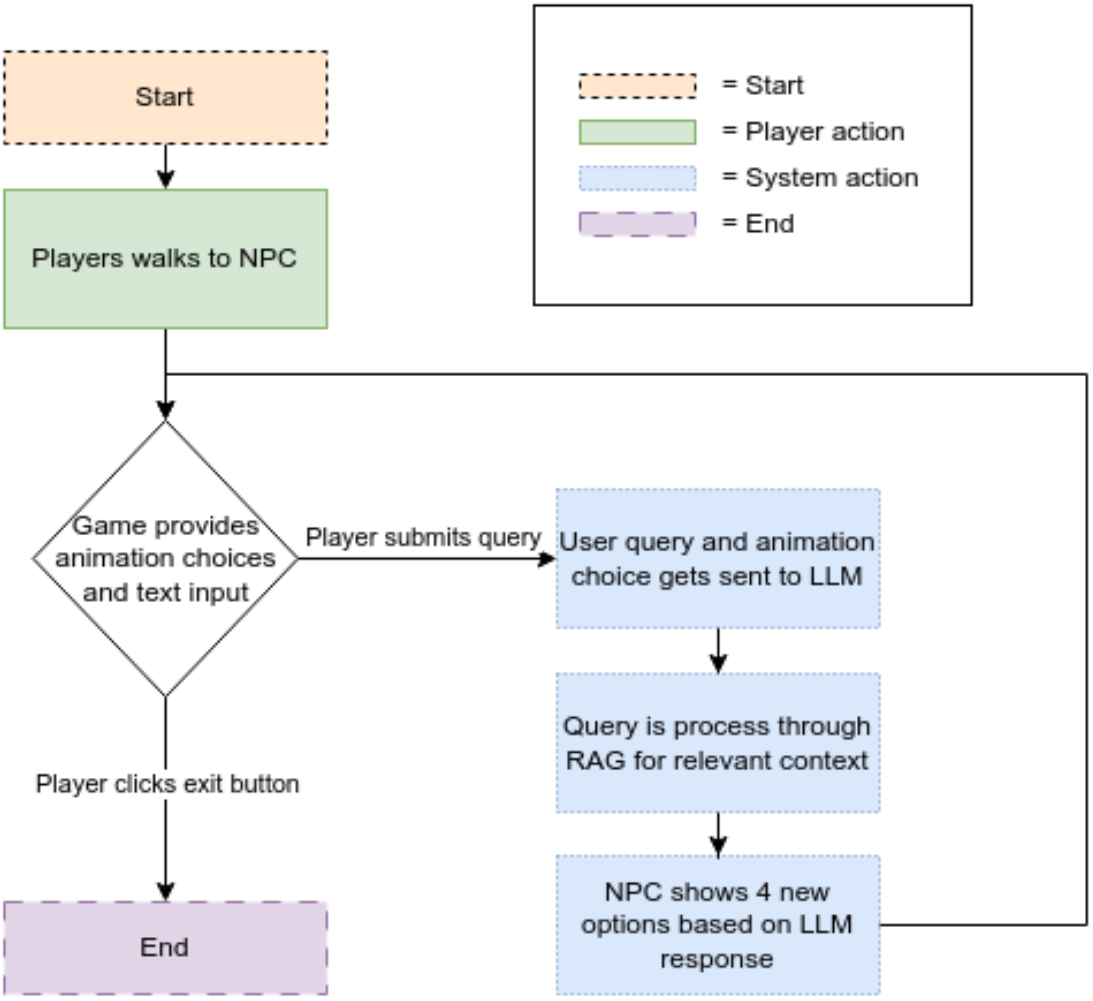

FIGURE 7.14: Level 6: Game-loop

7.7 Level 7: Collaborative Task-Based

In level 7, the LLM-based NPC and the player are intended to operate in collaboration. This means that the player and the LLM-based each get their half of the challenge and must then solve it by working together. This method of collaborative tasks can be of particular value in narrative sections. Co-constructivism has been shown to be applicable in educational settings [52], which are present in video games as tutorials or other instructional moments. Learning can be done on an individual basis, but it can also be considered a social activity and therefore falling in the socio-cognitive field.

In the demo, there is a basic survival setting in which the player and the LLM-based NPC split up need to decide the ideal location of their base camp. The player and the NPC split up and gather information about their half of the island. After exploring, they will reunite, and exchange the knowledge and discuss the ideal location to put down their base camp. This demo reinforces the idea of collaboration, as exploring the entire island would be too complex. Furthermore, from a game design perspective, it allows the game developer to subtly introduce relevant information via the LLM-based NPC. For example, it might be preferable to suggest that a base camp near the ocean might be preferable due to its proximity to the fish.

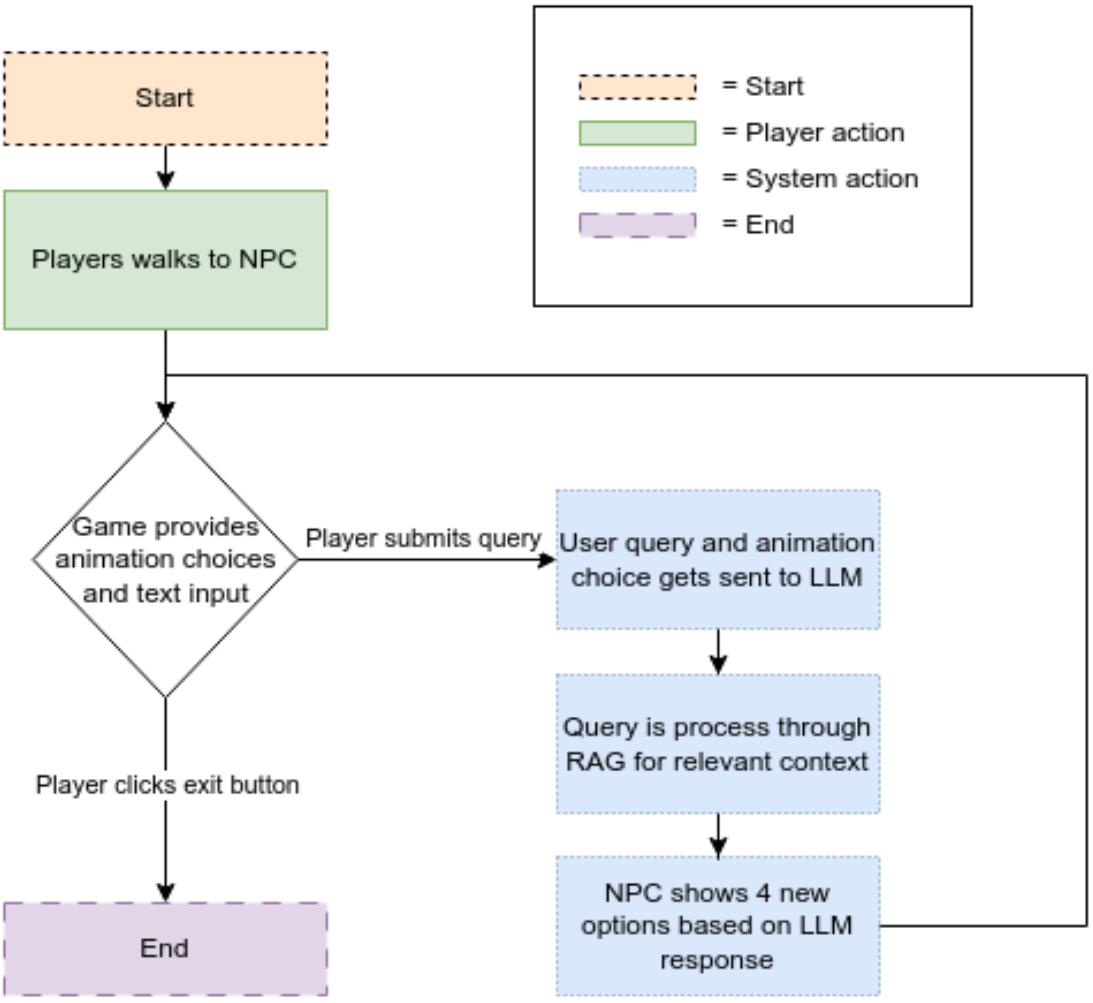

FIGURE 7.15: Level 7: Game-loop

FIGURE 7.16: Example of a drawing made within the tool. A single drawing is made. The player can submit a drawing when completed.

7.8 Level 8: Semi-Structured Symbolisms (3S)

The idea behind 3S is to transpose this idea of how communication is done from a direct message, meaning from the player to the LLM-based NPC in an easily accessible format such as text or even audio that is then converted to text. The demo features a game mechanic in which the player and the NPC do not converse in the same language, so they use drawings to visualize the message. For instance, the player teaches the LLM-based NPC to bake bread, by drawing a series of steps in order. The LLM-based NPC will then try to translate these steps to a series of actions in the video game. The player and the LLM-based NPC go back and forth and refine the symbolisms. It is the task of the LLM to interpret these symbolisms into a coherent input and then convert that input into a series of goals that the game can support via an automated planning system, in this case a purposeful action programming system or more commonly referred to as GOAP.

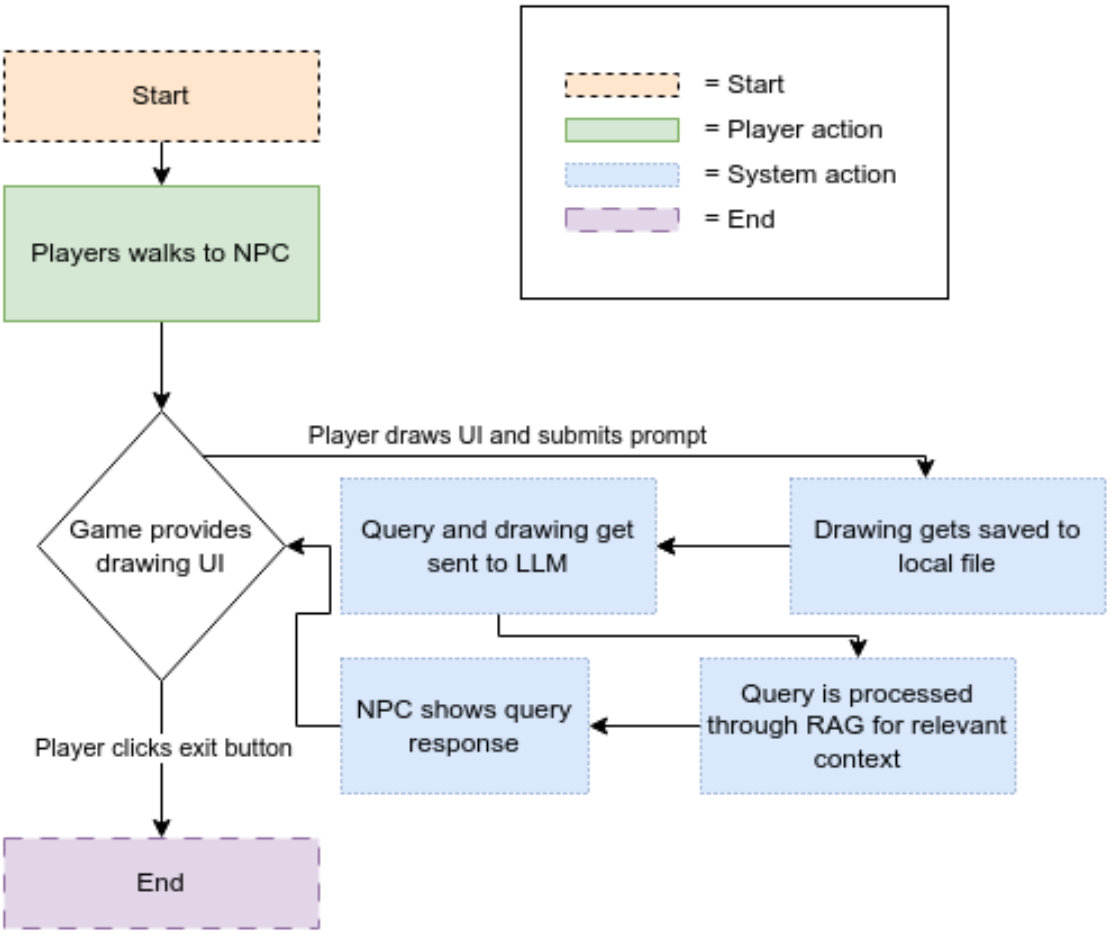

FIGURE 7.17: Level 8: Game-loop

7.9 Level 9: Emergent Language

An emergent language indicates a deeper aspect of socio-cognitive interaction, namely negotiation of meaning, resulting in the development of a shared dictionary. In this demo, the player and LLM-based NPC walk through a forest, generating a dynamic vocabulary system by pointing at objects where the LLM-based NPC learns novel words or phrases based on repeated interactions. Similar to level 5, objects in the world are used to create a context. For instance, if a player points at a tree and names that tree a certain word, this word will then apply to that object. The LLM will then try to use this word in its speech, thereby creating an emergent language with the player. The player and LLM will then attempt to have conversations based on in-world objects and words. The LLM can also suggest words for certain objects, thereby reinforcing the emergent property of the language.

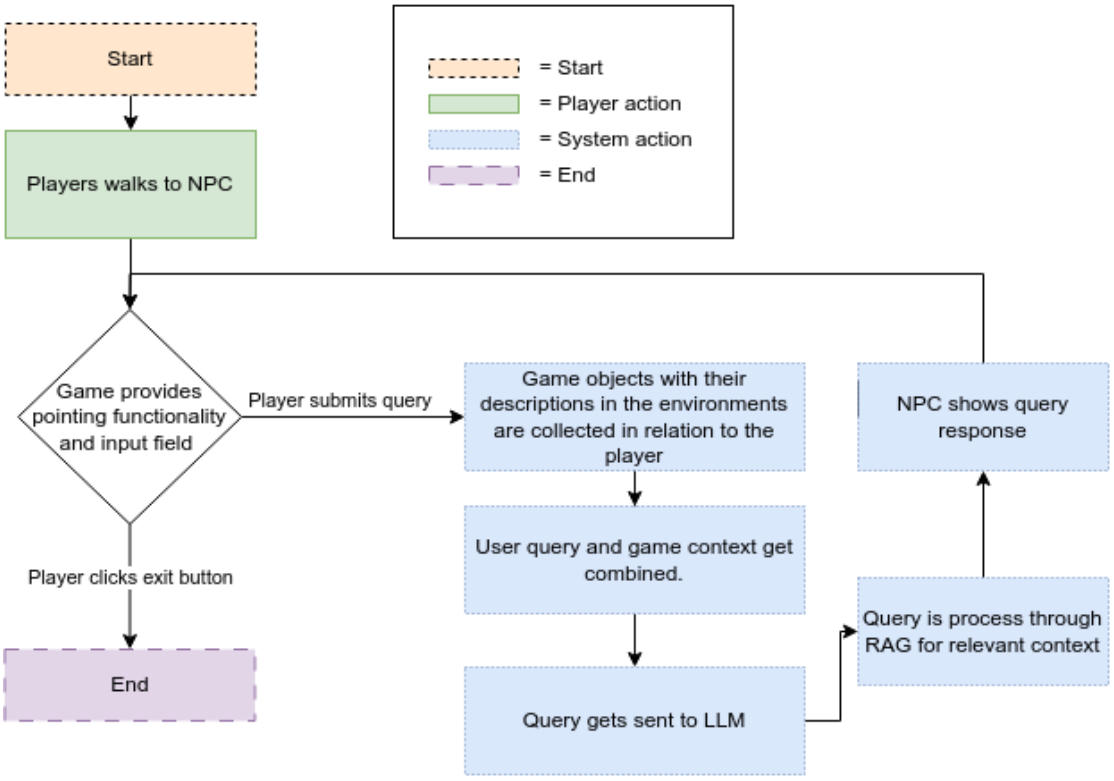

FIGURE 7.18: Level 9: Game-loop

7.10 Level 10: Multi-Modal Collaboration

The highest level of the model is an amalgamation of the previous levels. Multi-Modal collaboration would be the application of several layers for a singular interaction. For instance, level 8 introduced semi-structured symbolisms, but the gestures introduced in level 6 or even voice interactions in level 4, might aid in the development of a level 9 emergent language. In other words, an NPC capable of interpreting multiple socio-cognitive inputs.

In this demo, the player, and the LLM-based NPC work together to prepare for an upcoming expedition. The player can use spoken commands, text input, or gestures to communicate objectives. The LLM-based NPC recalls previous interactions, referencing past conversations, learned vocabulary (Level 9), and environmental conditions (Level 5) to offer suggestions. For example, if the player gestures toward a river, the NPC might recall their emergent term for "water" and suggest gathering supplies before crossing. If the player asks verbally about nearby shelters, the NPC integrates spatial awareness and guides them to a known safe location. The NPC's ability to dynamically switch between interaction styles ensures that every conversation feels adaptive and contextually rich. Furthermore, this environment will be more computationally demanding. Involving more roaming NPCs, thereby simulating a more practical game.

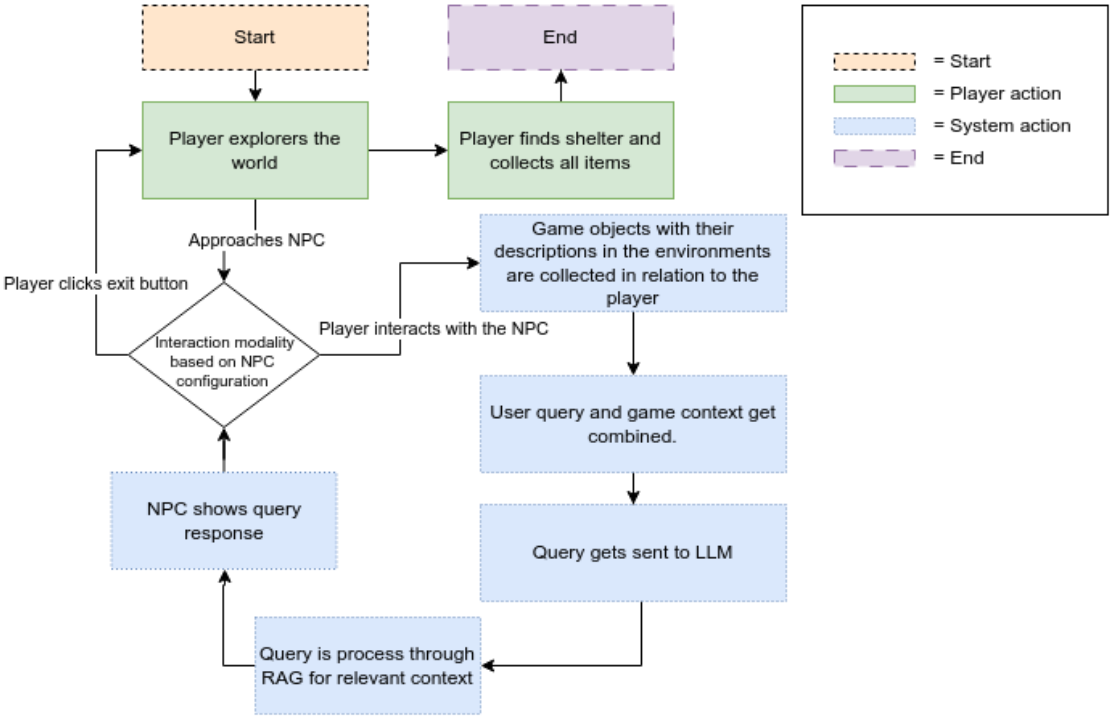

FIGURE 7.19: Level 10: Game-loop

Chapter 8

Results

8.1 Environment Analysis

The environment analysis provides insight into the hardware performance of the machine used for local testing. This is to provide a baseline of performance with which to base the results on.

SCNIL Prototypes

8.2 Level 1: Direct Textual Conversation

8.2.1 Response Quality

The LLM provided grammatically correct and well-structured responses, but lacked game-specific context, leading to generic replies. Without memory or retrieval augmentation, the NPC frequently forgot prior interactions, resulting in inconsistent responses when asked follow-up questions.

Hardware	Model	Type	TPS
CPU	TinyLlama	Prompt processing	435.24 ± 11.35
CPU	TinyLlama	Text generation	77.47 ± 0.41
CPU	DeepSeek	Prompt processing	139.60 ± 3.37
CPU	DeepSeek	Text generation	41.76 ± 1.40
GPU	TinyLlama	Prompt processing	30426.91 ± 943.31
GPU	TinyLlama	Text generation	492.95 ± 22.22
GPU	DeepSeek	Prompt processing	22580.40 ± 1052.57
GPU	DeepSeek	Text generation	309.37 ± 13.79

Table 8.1: Results of llama.cpp benchmark

Example Failure Case: The player asked the NPC its name, and after receiving a response, asked again later. The NPC provided a different name, highlighting its lack of memory.

8.2.2 Response Speed

The TinyLlama was capable of generating tokens at an average rate of 114 TPS. DeepSeek was comparatively slower, with a rate of 33 TPS. Both are thereby labelled as perceivably instant.

8.2.3 Failure Cases & Limitations

- No memory retention: The NPC is incapable of remembering past interactions. Follow-ups are impossible without direct referencing.
- Hallucinations: The NPC exhibits hallucinations by giving misleading and outright responses.

8.3 Level 2: Context-infused Conversation

8.3.1 Response Quality & Context Awareness

The LLM was capable of successfully incorporating game-specific knowledge such as world lore, characters, and the backstory of the LLM. Follow-up questions were possible, and the LLM was capable of emulating distinctive characteristics such as angry, sad, or annoyed.

8.3.2 Performance & Latency

The RAG overhead caused a significant performance impact during the initial few prompts. With the first prompt resulting in a TPS of 2.1, falling within the range of insufficient, and the second resulting in a TPS of 4.9, falling within the range of passable. By the third prompt, the response was set at 120 TPS, falling within the range of perceivably instant.

8.3.3 Failure Cases & Limitations

- Bias/Censorship breaking immersion: The DeepSeek model exhibits a degree of censorship that disrupts immersion. For example, prompts instructing LLM-based NPCs to adopt specific character roles or avoid referencing to

themselves as language models are often overridden by refusals or default responses designed to steer away from sensitive topics.

- Context Limitations: Due to context window constraints, the NPC forgot earlier details from extended conversations, requiring the player to repeat information.
- Increased Computational Cost: The need to retrieve and process additional data led to a higher system load, increasing both response latency and VRAM usage.

8.4 Level 3: Dialogue Options

8.4.1 Response Quality & Narrative Structure

The introduction of structured dialogue options significantly improved narrative coherence while reducing hallucinations and inconsistent responses. The LLM successfully generated context-aware choices, allowing players to shape conversations while maintaining structured narrative progression.

Example Success Case: When discussing a past quest, the LLM-generated options correctly reflected the player's past choices, allowing for meaningful branching dialogue.

Example Failure Case: In some instances, the LLM generated options that were too similar, providing redundant choices rather than a diverse range of responses.

Example Failure Case: The LLM could get stuck in a certain line of questioning or present several options that were equally enticing without being able to return to previous interactions.

8.4.2 Performance & Latency

Generating multiple dialogue options required more processing time, leading to a noticeable increase in inference delay. Instead of generating one response, the LLM had to create 3–5 possible responses. However, the responses were generally shorter than the open-ended questions, thereby limiting the total inferencing time.

8.4.3 Failure Cases & Limitations

- Limited Response Diversity: Some generated options were too similar, offering little real choice.

- Overly Generic Options: Occasionally, all available responses were vague or redundant, reducing engagement.
- Ending Choice: Ending a conversation was abrupt. Either the LLM is tasked with generating an end-condition, which essentially keeps the player as a 'hostage', or the player can end it but without a socially acceptable ending to a conversation.

8.5 Level 4: Voice Interaction

8.5.1 Response Quality & Naturalism

The introduction of voice interaction allows NPCs to respond through natural speech synthesis instead of text-based responses alone. Players can communicate via spoken input, which is then converted into text using a speech-to-text system before being processed by the LLM.

Success Cases:

- Context-aware dialogue: NPC responses remained coherent when integrating game lore and dynamic retrieval mechanisms.
- Naturalistic speech patterns: By incorporating slight speech delays and verbal fillers, responses felt more conversational and less robotic.

Failure Cases:

- Speech recognition errors: Due to ambient noise or varied accents, the speech-to-text system occasionally misinterpreted words, leading to irrelevant or incoherent responses.
- Response formatting issues: Since spoken dialogue lacks clear punctuation, responses sometimes lack intonation or structure, making them harder to understand.
- Latency in response time: The need to transcribe speech, process it, generate a response, and convert it back to speech introduced noticeable delays in NPC reactions.

8.5.2 Performance & Latency

Unlike previous levels, where text-based interaction allowed for rapid exchanges, voice processing introduced an additional computational layer: Speech-to-text delay, LLM response generation, and Text-to-speech synthesis.

8.5.3 Failure Cases & Limitations

- Accents and speech variability reduce effective use for different players. Furthermore, background noise, such as emergency vehicles driving by, can ruin a prompt.
- Synthesis of sentences for spoken word requires different syntax and is less strict in syntax.
- Lack of emotional expression: While text-to-speech provided intelligible speech, the speech lacked any expressive variation, making NPCs sound robotic.

Key Takeaways: Voice interaction can improve immersion but can introduce errors and misinterpretations. Latency from speech-to-text and text-to-speech impacted real-time responsiveness, requiring optimization. A possible solution is shorter would be to prompt shorter messages. For instance, replying questions are yes or no is likely to have fewer errors and reduce latency of the speech-to-text pipeline.

8.6 Level 5: Shared Environment

8.6.1 Response Quality & Environmental Adaptation

The integration of shared environment awareness allowed LLM-based NPCs to recognize and respond to in-game environmental changes. Unlike previous levels where interactions were limited to direct communication, NPCs now considered objects, locations, and dynamic world elements when generating responses.

Success Cases:

- Context-sensitive dialogue: NPCs dynamically referenced surrounding objects, weather conditions, and recent game events in their responses, making interactions feel more grounded.
- Spatial awareness: NPCs could acknowledge the player's physical position in the game world, reacting differently if the player was nearby or interacting from a distance.
- Memory of world changes: NPCs retain knowledge of modifications to the game world, such as moved objects, recent battles, or completed quests, allowing for more persistent and meaningful conversations.

Failure Cases:

- Inconsistent tracking of objects: While NPCs correctly referenced certain environmental changes, there were instances where they failed to register object interactions or incorrectly recalled past states.
- Delayed response updates: Due to the reliance on retrieval mechanisms, NPCs sometimes took longer to acknowledge recent changes in the world, causing occasional mismatches between player actions and NPC responses.
- Multidimensional Spatial Awareness: NPCs require complex and detailed input to determine what the spatial layout is to make intelligent representations.

8.6.2 Performance & Latency

Adding environment awareness did not increase the computational time significantly. The added information was added via a tracking schema of the environment that could be expressed in less than 1000 tokens. Multidimensional spatial awareness can increase this token count significantly.

8.6.3 Failure Cases & Limitations

- Multidimensional Spatial Awareness: NPCs require complex and detailed input to determine what the spatial layout is to make intelligent representations.
- Inconsistencies and Hallucinations: Contrary to the previous levels, in this level inconsistencies cause confusion and unintended game loops in which the player will attempt to verify environmental factors before realizing they are hallucinations. This causes unintended and undesirable game loops.

8.7 Level 6: Gesture-based Cues

8.7.1 Response Quality & Non-Verbal Communication

The integration of gesture-based cues allowed players to interact with NPCs using non-verbal inputs such as pointing, nodding, or using predefined gestures.

Success Cases:

- Intuitive interaction: Players could point at objects, and NPCs responded appropriately by acknowledging, describing, or interacting with the pointed object.
- Non-verbal confirmations: Nodding and shaking the head were correctly interpreted by NPCs, allowing for quick yes/no interactions without requiring verbal input.

- Multi-modal dialogue: Players could combine gestures with speech or text, allowing for more nuanced conversations.

Failure Cases:

- Ambiguity in interpretation: In some cases, NPCs misinterpreted gestures due to unclear intent, especially when multiple interactive objects were nearby.
- Over-reliance on explicit gestures: NPCs primarily responded to directly mapped gestures, struggling with more abstract or indirect movements that required additional reasoning.

8.7.2 Performance & Latency

Adding gesture-based interactions required real-time detection and processing of non-verbal inputs. Furthermore, the NPC needed animation synchronization. NPCs responding to gestures with appropriate body language required real-time animation blending, sometimes causing minor delays in rendering complex interactions. The latency was negligible as the necessary information can be stripped down to minimal data.

8.7.3 Failure Cases & Limitations

- Recognition inconsistency: Gesture detection was less reliable in fast-paced interactions, where quick movements were sometimes missed or misclassified.
- Limited expression range: The system worked well for simple gestures (pointing, nodding, waving) but struggled with more complex or culturally specific gestures.
- The LLM was incapable of handling conflicting information: For instance, if the player referred to an object pointed in the general direction, the LLM was incapable of inferring the intended object, thereby generating an inaccurate response.

8.8 Level 7: Collaborative Task-based

8.8.1 Response Quality & Cooperative Engagement

The introduction of collaborative task-based interactions allowed NPCs to engage in cooperative problem-solving with the player. Unlike previous levels where NPCs primarily responded to player input, this level enabled NPCs to actively participate in shared objectives, such as solving puzzles, exploring environments, or planning

strategies. This significantly increased NPC agency, making them feel more like teammates rather than passive dialogue partners.

Success Cases:

- Shared problem-solving: NPCs effectively assisted in solving multistep puzzles by recognizing player actions, providing suggestions, and adapting their responses based on the task's progress.
- Dynamic task planning: NPCs adjust their behaviour depending on the player's decisions, such as suggesting alternative routes in navigation-based challenges or adjusting plans when encountering obstacles.
- Real-time coordination: NPCs successfully performed task-switching, responding to changing priorities within the game world.

Failure Cases:

- Over-reliance on predefined solutions: NPCs performed well when tasks followed an expected path, but if players attempted an unconventional approach, the NPCs often failed to adapt dynamically.

8.8.2 Performance & Latency

Introducing real-time task collaboration required continuous environment scanning, decision-making updates, and memory retrieval, which increased computational demand.

8.8.3 Failure Cases & Limitations

- The two chosen LLMs lacked the necessary complexity capable of this type of multimodal problem-solving. This led to the need, for the first time, for a more complex model.
- Inability to handle creative problem-solving: When players deviated from expected solutions, NPCs were unable to adapt, instead repeating scripted responses.

8.9 Level 8: Semi-Structured Symbolisms

8.9.1 Response Quality & Symbolic Interpretation

The introduction of semi-structured symbolisms allowed NPCs to interpret and respond to non-textual forms of communication, such as symbols, drawings, or

visual patterns. Unlike previous levels, where interactions were either text-based, voice-based, or gesture-based, this level required NPCs to decipher meaning from abstract representations and translate them into actionable responses.

Success Cases:

- Player-driven communication: Players successfully used symbols to convey basic commands, directions, or abstract concepts, and NPCs correctly inferred their intended meaning.
- Iterative meaning refinement: NPCs were able to engage in back-and-forth exchanges, adjusting their interpretation based on additional symbols or clarifications from the player.

Failure Cases:

- Ambiguity in symbol interpretation: Some symbols had multiple possible meanings, causing NPCs to misinterpret player intent.
- Lack of context adaptability: NPCs often failed to recognize symbols when they were presented in unconventional formats or when they deviated slightly from expected shapes.
- Lack of chronology: NPCs were unable to reason the relationship between certain objects, for instance, in an example with a coconut tree, the NPC was unable to infer that the coconut came from the tree.

8.9.2 Performance & Latency

Processing symbolic inputs required real-time recognition, classification, and interpretation, adding extra computational overhead compared to purely textual interactions.

8.9.3 Failure Cases & Limitations

- Limited emergent understanding: NPCs struggled with abstract or layered symbolic meanings, requiring predefined categories for interpretation.
- The interactions with NPCs put the player in a teacher role: This is not bad from a socio-cognitive perspective, but combined with the previous level, should the game designer want the NPC to have more autonomy and teach the player certain symbolisms, this proved to be incredibly challenging.

8.10 Level 9: Emergent Language

8.10.1 Response Quality & Adaptive Communication

The introduction of emergent language allowed NPCs and players to co-develop a shared lexicon over time, enabling more dynamic and personalized interactions. Unlike previous levels, where NPCs interpreted predefined symbols or gestures, this level required NPCs to learn, remember, and apply new terms or expressions introduced by the player.

Success Cases:

- Gradual vocabulary formation: NPCs successfully associated words, phrases, or symbols with new meanings through iterative interactions, making conversations feel more organic.
- Context-aware adaptation: NPCs adjusted their understanding of emergent words based on previous interactions, allowing for progressive refinement of shared language.

Failure Cases:

- Inconsistent word recall: Some NPCs forgot or altered previously learned meanings, leading to confusion or contradiction in later interactions.
- Limited abstraction: NPCs struggled with abstract or metaphorical meanings, often assigning overly literal interpretations to new terms.
- Difficulty with syntax adaptation: While NPCs could recognize new words, they struggled with grammar rules, leading to awkward phrasing or sentence structure issues.
- Multilanguage. Switching between multiple languages resulted in consistent experiences. For instance, a word for a tree in English, converted to Chinese[1] caused issues in its translation or purpose. This likely implies that the LLM is just memorising the words, not actually abstracting them to transferable contexts.

8.10.2 Performance & Latency

Real-time word learning, memory storage, and adaptive language generation significantly increased processing demands compared to previous levels. Long-term memory management, storing and recalling multiple user-defined words, increased VRAM and memory footprint, particularly in extended gameplay sessions.

[1] Chinese (Mandarin) was picked as a second language for evaluation, as DeepSeek-R1 only supports Chinese and English

8.10.3 Failure Cases & Limitations

- Memory drift: Over long interactions, some previously defined words shifted meaning due to context window limitations or even disappeared.
- Lack of multi-layered meaning: NPCs struggled to recognize synonyms or adjust meaning dynamically based on context.
- Rigid sentence integration: While NPCs learned individual words, they failed to fluidly integrate them into natural conversation, sometimes resulting in robotic or unnatural phrasing.
- LLMs were incapable of requesting additional information from the player: For instance, letting the player put it in certain contexts or conditions. The LLM is incapable of validating its hypothesis.

8.11 Level 10: Multi-Modal Collaboration

8.11.1 Response Quality & Integrated Interaction

The introduction of multi-modal collaboration allowed NPCs to seamlessly combine multiple interaction modes, such as text, speech, gestures, environment awareness, and emergent language, into a single cohesive communication system. Unlike previous levels, which focused on isolated interaction methods, this level aimed to create fluid and adaptive NPC responses by dynamically selecting the most contextually appropriate mode based on the player's input.

Success Cases:

- Seamless input switching: NPCs successfully interpreted and responded to mixed inputs, such as spoken commands paired with gestures or text input combined with environmental cues.
- Adaptive responses: NPCs adjusted their communication mode based on context, choosing text for precise information, speech for immersive dialogue, and gestures for non-verbal cues.
- Modality transfer: The NPCs were rarely capable of transferring a piece of information that was supplied in one modality to deliver in a different modality.

Failure Cases:

- Inconsistent integration: Some input combinations caused NPC confusion, such as when gestures and spoken input conveyed conflicting intentions.

- Over-processing delays: Simultaneously handling multiple input types increased response times, making real-time interactions feel slower compared to previous levels.

8.11.2 Performance & Latency

Processing multiple simultaneous inputs, cross-referencing previous interactions, and adapting response formats introduced significant computational demands. Synchronization issues arose, ensuring coherent NPC reactions across multiple modalities, which led to animation and speech timing mismatches, requiring additional processing to correct delays.

8.11.3 Failure Cases & Limitations

- Processing conflicts: NPCs struggled to prioritize inputs when multiple modalities contradicted each other, leading to unexpected responses.
- Limited dynamic learning: While NPCs retained information across modalities, they lacked the ability to adjust interaction preferences over time based on player habits.

Chapter 9

Design principles

Having discussed the methodology, the prototypes, and the results, we can now present the design principles. In this chapter, we discuss who these guidelines are meant for, what considerations need to be made, the reason for this consideration and to what level of the SCNIL model they are applicable to. The principles are not ordered in any way. This chapter ends with a table summarising the principles and how they relate to the SCNIL levels.

Principles Table

#	**Principle**	**Key Focus**	**Levels**
1	Words are cheap, tokens are not	Evaluate when to use LLMs vs. scripted logic	1–3, 7
2	Game Over, Your NPC has talked too much	Manage dialogue length and pacing	1–3, 10
3	Talk is cheap, Context is priceless	Ensure relevant context integration for NPCs	2–10
4	Lag kills conversations too	Maintain real-time performance	1–6
5	Give NPCs time to read the lore too	Simulate believable information spread	All levels
6	If it feels like a gimmick, it probably is	Avoid unnecessary modalities	4–10
7	Narrative is the driver, not the passenger	Align interactions with narrative logic	5–10
8	One modality to rule them all? No.	Provide flexible modality options	2–10

9	Friction makes for believable fiction	Introduce NPC autonomy and resistance	2–10
10	Principles guide, but creativity decides	Encourage experimentation and flexibility	All levels

TABLE 9.1: Overview of Design Principles and Their SCNIL Applicability

9.1 Principle 1: Words Are Cheap, Tokens Are Not

Not every NPC interaction requires an LLM, not even every socio-cognitive interaction requires an LLM. While LLMs enable dynamic, generative responses, they are not a one-size-fits-all solution. Pre-scripted dialogue, decision trees, or traditional AI systems can often provide a faster, more coherent, more predictable, and less computationally intensive solution. Using an LLM in a scenario where its capacity for generative responses and behaviour are not clearly beneficial will result in a suboptimal experience. LLMs, at the time of writing, are a trade-off. They can allow developers to do this previously done but introduce a host of challenges. If something can be done using classical methods, then an LLM is not the solution.

Guidelines:

- Ask yourself: Does this interaction or moment need dynamic generation? If an NPC provides fixed, objective-centred information, then pre-scripted dialogue may be preferable.
- Evaluate player expectations: some NPCs (e.g., merchants, store vendors or tutorials) benefit from predefined and structured responses, while others (e.g., lore-heavy characters, companions, or gameplay moments) may need adaptive dialogue or generated interactions.
- Consider the computational resources: LLM responses are more computationally intensive than scripted alternatives in terms of latency, memory, and compute.

SCNIL Applicability: Relevant across all SCNIL levels, especially Levels 1–3 (Basic Interactions) and Level 7 (Task-Based Collaboration), where the decision to use an LLM versus a pre-scripted system directly impacts design.

9.2 Principle 2: Game Over, Your NPC Has Talked Too Much

LLMs naturally generate open-ended responses, leading to interactions that can continue indefinitely unless these are specifically managed. In a game setting, prolonged conversations may disrupt pacing, reduce player agency, or create unnecessary delays in gameplay. For instance, if you are having a final conversation with an NPC before a boss battle, having a lengthy discussion about the weather will ruin any sense of suspense, but by default, the LLM will encourage this behaviour.

Implementation Guidelines:

- Define the expectations for a specific interaction. This means that when a player approaches an NPC at a specific time, the game context used for generating a response needs a boundary for acceptable expectations. When a player visits a town for the first time, LLM-based NPCs might engage in longer conversations, whereas in the previously mentioned boss fight example, the conversation should be short.
- Define when a specific interaction is appropriate. If a player just interacted with an LLM-based NPC, decide if this appropriate. What would the reaction of the LLM-based NPC be and why. For instance, a hard-coded response, so not by the LLM, could be that the NPC is busy. Deciding if an interaction is appropriate lays the foundation for player expectations in the game and the level of immersion.
- Implement contextual cut-off triggers. To aid the expectations for specific interactions, when there is a specific goal of an interaction, implement triggers that cut off the interactions when the goal has been achieved.
- Implement natural ending cues. Consider social etiquette when ending a conversation by an LLM-based NPC to when ending a conversation, this means excusing themselves, redirecting the player or even letting the NPC interact with another NPC. Particularly, the last one turned out to be a more natural transition. The goal with this is to cement the fact that an interaction is done, and the player needs to move on. There should be no ambiguity if an interaction is still occurring.
- From a player's perspective, ending a conversation is also a challenge. While in a dialogue state, wherein a series of choices are presented to a character, it is clear when it ends, LLM-based NPCs do not have this luxury. A player can continue the conversation and struggle to find a suitable conclusion. These

conversational dynamics are being actively studied and lack a conclusive answer [53]. Therefore, do not shy away from providing the player with an immediate exit or subtly drop hints that the conversation needs to conclude. Depending on culture and player preferences, they are unlikely to enjoy the feeling of the LLM-based NPCs consistently ending the conversation with them, but them never doing the same the other way around.

SCNIL Applicability: Relevant to all levels, especially Levels 1–3 (Basic Interactions) and Level 10 (Multi-Modal Conversations), where prolonged dialogue could impact pacing.

9.3 Principle 3: Talk is Cheap, Context is Priceless

LLMs are powerful, but they are not a one-size-fits-all solution for LLM-based NPC interactions. Without the proper constraints, integration, and measured oversight, LLM-based NPCs can generate inaccurate, irrelevant, and immersion-breaking responses. Unlike the classical AI-driven NPCs, LLMs do not inherently track game state, player progress, or have any world awareness. Monitoring and the presentation of the world must be explicitly provided through structured prompts, retrieval mechanisms, and controlled access to in-game data. This adds another level of complexity to the game that needs to be implemented, reflected, and assessed. A quality LLM-based NPC functions as a component of a larger system, alongside scripted dialogue, game logic, and a dedicated world-state tracker to create coherent and meaningful interactions.

Implementation Guidelines:

- Control Input & Context: Use retrieval-augmented generation (RAG) or structured memory to ensure LLM-based NPCs only have access to relevant game data rather than relying on generic model training.
- Monitor NPC Awareness: Track what the LLM-based NPC should and should not know based on player progress, quests, and world events. Each NPC needs checks and monitoring to activate the appropriate part of the lore.
- Guardrails are essential: Use guided prompting, system messages, or fallback scripted responses to prevent the LLM from generating inconsistent or off-topic dialogue.
- Combine with Traditional Systems: Use LLMs alongside pre-scripted interactions where necessary, rather than replacing all NPC dialogue with generative text.

SCNIL Applicability: Relevant at all SCNIL levels, especially Levels 2–10, where context integration, game state tracking, and memory retention become essential for meaningful LLM-based NPC interactions.

9.4 Principle 4: Lag Kills Conversations Too

Real-time responsiveness is crucial for maintaining immersion in gameplay, although instantaneous responses are not desirable. LLMs in games must balance fast response times with realistic conversational pacing. In real-world interactions, pauses, delays, and processing time contribute to believability and pacing. LLM-based NPCs that respond too quickly can feel unnatural, while those that take too long can frustrate players. The challenge is optimizing response speed while also blending in subtle delays where appropriate to enhance realism.

Implementation Guidelines:

- Optimize for Speed: Ensure LLM inference is low latency for real-time interactions, especially in combat or action sequences.
- Introduce Natural Delays: Implement short pauses or animations (NPCs "thinking", reacting before speaking or vocally expressing thought difficulty) to make conversations feel more natural.

SCNIL Applicability: Applies across Levels 1–6, particularly in voice interactions, text-based dialogue, and multimodal cues where pacing is key.

9.5 Principle 5: Give NPCs Time to Read the Lore Too

The information flow within the game world should make temporal sense. If an NPC learns new information, it may take time to process it, communicate it, or spread it to other LLM-based NPCs. Similarly, when a player issues a command or asks a complex question, a short delay can signal that the NPC is "thinking" rather than instantly generating a response. As the socio-cognitive capacity of the LLMs increases, and the player starts anthropomorphising on a deeper level, players become more aware and cognisant of the information propagation dynamic.

Implementation Guidelines:

- Control NPC knowledge updates: NPCs (not just LLM-based NPCs) should learn new information at a natural pace, rather than instantly reacting to world changes.

- Simulate information spread: Introduce delays in knowledge dissemination, where some NPCs learn about events before others. For instance, if a big battle took place in one town, and the player rushes across the border, they are unlikely to have heard about it, but when the player wakes up in the morning, the news has spread.
- Use in-game time as a factor: Certain events may only be recognized after a set period, requiring in-game days or interactions before an NPC acknowledges them.

SCNIL Applicability: Relevant to all levels.

9.6 Principle 6: If It Feels Like a Gimmick, It Probably Is

Different interaction modalities, such as text, speech, gestures, and all environmental cues offer unique and immersive methods of interaction, but not every modality is the correct choice for every interaction. While gesture-based cues can feel intuitive, they may be cumbersome in practice, requiring unnecessary effort from the player. The key is to choose the appropriate modality based on usability, player comfort, narrative goals, and game flow rather than forcing an interaction style for immersion alone.

Implementation Guidelines:

- Prioritize ease of use: If a modality introduces friction or requires extra effort without a meaningful advantage, a simpler input method may be preferable. Do not add gimmicks for the sake of adding gimmicks.
- Context matters: Gestures may be useful for symbolic interactions but feel excessive for simple dialogue exchanges. Voice input may be great for role-playing moments but impractical in noisy environments.
- Test for fatigue: If a modality requires repeated physical actions (e.g., pointing, nodding, drawing symbols), consider whether it remains enjoyable over extended play sessions.

SCNIL Applicability: Relevant to Levels 4–10, particularly Level 6 (Gesture-Based Cues) and Level 10 (Multi-Modal Interactions), where choosing the right interaction mode is key to usability.

9.7 Principle 7: Narrative is the Driver, Not the Passenger

When designing LLM-based NPC interactions, both the narrative and player experience should be the determining factors for the choice of interaction modality, not the other way around. Starting with a modality-first approach such as a task-based interaction can lead to incoherent and gimmicky interactions that break immersion rather than enhancing it. Since socio-cognitive engagement is central to how players interact with the LLM-based NPCs, every input method should feel purposeful, intuitive, and narratively justified.

If the interaction feels forced or improper, it risks breaking immersion. Players should not feel like they are engaging with technology, even though they are. Instead, they should feel like they are participating in a living, cognizant world.

Implementation Guidelines:

- Narrative first, technology later: Start by defining what the player needs to feel, learn, or accomplish, then decide the method with which to achieve this followed by assigning the appropriate modality to support that goal.
- Keep interactions seamless: If an interaction method calls attention to itself instead of enhancing immersion, reconsider its implementation. Iterative testing is encouraged.
- Evaluate necessity: Does a specific modality add to the player's sense of presence in the world, or does it distract from it? Does the application of the modality feel natural? For instance, the example in which the NPC and the player cannot talk to each other naturally introduces Level 8's semi-structured symbolism. In a situation where they can talk to each other, it is improper.
- Ensure modality consistency: If a player learns to interact with a modality in a certain way, try to transfer this application similarly. Limit the diverse ways in which a player is asked to do something similar. For instance, if the player uses a specific type of grammar for the development of an emergent language (Level 9), extract this grammar and apply it when the player is developing a new language. While it might seem beneficial to create new experiences every time, they can quickly limit the expressiveness of the player and the level of engagement.

SCNIL Applicability: Relevant to Levels 5–10, where interactions extend beyond basic text and require deeper integration into gameplay and narrative design.

9.8 Principle 8: One Modality to Rule Them All? No.

Players will have different preferences when it comes to LLM-based NPC interactions. Some may find voice input engaging, while others might prefer the precision and speed of text. Since player agency and immersion are key to socio-cognitive engagement, forcing a single modality can alienate parts of the player base or decrease agency and immersion.

A multimodal approach allows players to choose how they engage with LLM-based NPCs, enhancing the accessibility, personalization, immersion, and thereby, hopefully enjoyment. By offering multiple interaction methods (text, voice, gestures, or contextual menu selections), games can accommodate diverse player preferences while maintaining a cohesive experience.

Implementation Guidelines:

- Allow seamless switching between modalities: Players can use voice when they want immersion, text when they need precision, or gestures when it feels natural. Allow them to choose and make sure they know they have the choice.
- Avoid modality exclusivity: Do not design interactions that require only voice, only text, or only gestures unless absolutely necessary. A player who is used to voice and suddenly must use text will experience a decreased modality accessibility. Furthermore, if a modality is often inaccessible in the game (for instance, if voice is rarely usable), consider not integrating the modality at all.
- Consider accessibility needs: Ensuring players with different abilities can choose the most comfortable interaction method.
- Design interactions to be robust across multiple inputs so that no single modality feels "tacked on" or inferior to others.

SCNIL Applicability: Relevant to Levels 2–10, particularly in voice interactions (Level 4), gesture-based cues (Level 6), and multi-modal collaboration (Level 10).

9.9 Principle 9: Friction Makes for Believable Fiction

In human interactions, individuals don't always agree, they have different priorities, they can have misunderstandings, and their own personal motivations. This kind of friction is a key part of socio-cognitive engagement, making interactions feel dynamic and believable. If an LLM-based NPC only exists to please the player,

it risks feeling robotic and artificial. LLM-based NPCs need autonomy to make conversations meaningful.

Friction is already a key part of narrative design, but it requires even more attention when developing LLM-based NPCs. LLMs are inherently designed to generate agreeable responses, making them less likely to challenge, resist, or outright reject player requests unless explicitly constrained. Without a system that reinforces autonomous goals, LLM-based NPC interactions can become flat, predictable, transactional, and meaningless.

Implementation Guidelines:

- Define NPC goals and constraints: NPCs should have personal objectives, biases, or limitations that sometimes conflict with the player's desires.
- Introduce negotiation mechanics: Instead of NPCs instantly complying with requests, they should ask for something in return, express doubt, or refuse outright if it goes against their interests.
- Balance cooperation and resistance: NPCs should still be helpful when appropriate but should not act like mindless assistants—they should engage in persuasion, hesitation, or questioning.
- Acquiescence is okay: LLM-based NPCs can change their mind at times. It is important for them to grow and even take the priorities of the player as their own goals and constraints. However, this needs to happen sporadically—the world does not revolve around the player, but they can leave their mark on it.
- Use memory to reinforce personality: If an LLM-based NPC previously had a disagreement with the player, their responses should reflect that past friction.

SCNIL Applicability: Essential for Levels 2–10, especially in collaborative task-based interactions (Level 7), emergent language negotiation (Level 9), and multimodal interactions (Level 10) where NPC autonomy matters most.

9.10 Principle 10: Principles Guide, But Creativity Decides

The design principles were created to help with the development of consistent, immersive, and coherent LLM-based NPC interactions. Innovation comes from pushing boundaries and breaking conventions. The field of LLM-driven NPCs is still in its early stages, and purpose-built models for games do not yet exist. This means that current best practices are temporary solutions, not absolute laws.

Developers should feel empowered to ignore, adapt, or completely rethink these principles in pursuit of something new, unconventional, or experimental. Many of gaming's most revolutionary mechanics and storytelling methods came from defying expectations, and the same will likely be true for LLM-based NPCs.

Implementation Guidelines:

- Adapt existing principles: If a principle does not fit your vision, try to understand why this principle was made, keep that assumption or cause in mind, and redefine it.
- Prototype freely: The only way to discover what works is to try and fail. Unexpected results may lead to novel game mechanics.
- Embrace LLMs as an evolving tool: What is impractical today may become feasible in the near future as AI models improve.
- Focus on creative impact: If breaking a principle leads to something fun, immersive, or narratively compelling, then it was worth it.
- Force a certain play style: There are few design principles that advocate for player agency and autonomy. If you want to develop a game to suit a narrative design that forces the player to interact in a highly specific way, do it. There is likely to be a niche of people that will enjoy this experience regardless of the opinion of the mass market.

SCNIL Applicability: Relevant to all levels.

Chapter 10

Discussion

This chapter discusses the research findings, the challenges we experienced during the prototyping phase and insights we have gained for LLMs within videogames.

10.1 Local LLM Limitations

In the background chapter, we detailed two challenges. One of the challenges described was that of local LLMs, which have disadvantages when it comes to computational limitations. We therefore take the time to assess what limitations we have experienced, their significance, and viable solutions.

The largest limitation we experienced was the context window of the LLM and providing the appropriate contextual information. The challenge with an LLM is not in interfacing with the LLM, but that they require a significant amount of contextual information to provide a quality response. The task of the developer is therefore to provide the infrastructure with which to monitor, store, and maintain the contextual information. This is as much art as engineering. The context window of an LLM is the amount of text that the model can consider or "remember" at any one time[54].

In a video game setting, it is likely that the context window is smaller than the total information at play. This context window size issue is also prevalent when it comes to cloud-based LLMs, albeit less significantly. There are cloud-based LLMs with context windows in the millions of tokens[55], compared to the maximum size of one hundred and twenty-eight thousand we attempted during testing.

This lack of context window size necessitates summarizing or condensing information. This process too can become complicated and requires a careful selection of what information is relevant and required. Unfortunately, this process can bring

its issues, such as reliability. Furthermore, it also introduces overhead in the development cycle.

10.2 Hallucinations

In the background chapter, we detailed how hallucinations can be a challenge when working with LLMs. We detailed how Retrieval Augmented Generation (RAG) is a mitigation tool by which context is supplied to prompts to ground the response in a factual basis which was implemented during testing.

RAG was able to force the LLM-based NPC to generate responses based on the appropriate context. RAG was not without its flaws, as it increases the delay between player interaction and LLM-based NPC response. The data-dump of the Unofficial Elder Scrolls Pages used saw a decrease in performance between 30% and up to 159%, depending on the query. While this is a substantial increase, with GPU-based inferencing, performance was still an order of magnitude beyond the threshold for perceivably instant. Existing research on this topic has shown that an optimized RAG is likely to be anywhere between 20–30%, a typical RAG between 50–80%, and a worst-case being over 100% [56].

This is, however, a statistic with questionable value for real-world gaming applications. In a commercial game, this type of external knowledge base can be optimized for search queries by either grouping information to simplify search queries, applying caching to reduce search duration, and other RAG improvement methodologies. Video games allow more RAG optimization as the source material, made by the game studio, can be optimized from the moment the lore is first typed. Thereby making it possibly even more efficient.

This might give the impression that RAG implementations are a simple problem to solve. Unfortunately, implementing RAG on a large-scale game will be a complicated process. This is because RAG suffers from scalability issues.

The scalability, referring to the application of LLM-based NPCs in games that contain extensive game lore, extensive player interactions, and narrative progression, is a major issue that game developers must carefully evaluate. For the NPC to have accurate information, the game needs to keep extensive track of the relevant information at the time of an NPC interaction. While information such as the location of the NPC and the weather are relatively simple data points to access, narrative challenges are a different story.

For example, if a player talks to certain NPCs in a town and then refers to an event that only some NPCs should know about, the retrieval system must determine whether the LLM-based NPC should be aware of this event.

In a traditional RPG, this is managed through scripted flags, ensuring that NPCs only reference events the player has witnessed or directly influenced, or simply put, a game developer programs NPCs to explicitly mention something, thereby creating a limited number of possibilities. However, in an LLM-based system using RAG, knowledge retrieval is not inherently restricted, meaning an LLM-based NPC could reference spoilers, private conversations, or events outside their logical awareness. This feature of segmented knowledge makes RAG more difficult, as the knowledge base needs to be continuously updated. Updating the knowledge base is not an effortless process, as it requires either creating new document embeddings or programming the game in a way that certain in-game flags generate a certain context that is supplied during the prompt. Both of these possibilities require these narrative decisions to be known at the time of development, which is not a guarantee in current game development pipelines.

Information segmentation must also account for temporal context. For example, if an event occurs in one city and the player flees to another, NPCs should not immediately be aware of it. Instead, information should spread gradually over time, reflecting how news and rumors naturally travel. Without this, NPCs may either appear omniscient, knowing details they should have no access to, or isolated from the world, acting as if events outside their immediate surroundings never happened, both of which can break immersion.

In the context of socio-cognitive interactions, this becomes even more critical. The principle of distributed cognition suggests that knowledge is not centralized but shared across individuals and environments. For NPCs to feel real and cognizant, they must exhibit realistic expectations regarding knowledge propagation, acknowledging what they should know, when they should know it, and how they learned it.

RAG can significantly improve NPC responsiveness and contextual accuracy. Unfortunately, RAG also introduces a complex layer of knowledge management that must be carefully designed. LLM-based NPCs must exhibit realistic, context-aware interactions, which demands a structured approach to knowledge segmentation and dynamic updates on a temporal basis. Without these considerations, NPCs risk breaking immersion, either by knowing too much or too little. As a video game increases in size, the knowledge management will scale exponentially. Addressing these scalability issues will be essential for making LLM-driven NPCs a viable and seamless part of future game worlds.

10.3 Agency vs Narrative

One of the most severe shortcomings discovered during experimentation was ensuring narrative consistency while maintaining player agency. Unlike traditional dialogue, where responses are tightly controlled, LLMs generate responses dynamically, making it difficult to enforce story constraints. This often leads to what we refer to as *narrative drift*, where LLM-based NPCs unintentionally reveal spoilers, contradict lore, or break quest logic by responding too flexibly to player input. Essentially rewriting the story at the moment.

It is our belief that this issue originates from an LLM-model issue. LLMs are designed to be cooperative and user-driven, which implies that they try to accommodate player input as best as they can, rather than enforce defined limitations. If a player claims that an NPC is their long-lost sibling, for example, the LLM might accept this as truth, even if it contradicts established storylines. While this might appear as strange behaviour for a player, this lack of boundaries means that this phenomenon can happen accidentally.

To clarify, this is a different problem than the hallucination problem. Hallucinations originate from a lack of factual basis, thereby generating them to fill a prompt. Narrative drift originates from a model designed to accommodate the player. It ignores the factual basis in favour of providing a response that is more user-friendly.

To prevent this issue, strict narrative controls and response filtering are needed to mitigate that NPCs stay within the game's intended story structure. However, this might not be enough to prevent it from happening outright.

10.4 LLM-Based NPCs: A Different Game Entirely

One can take a glance online and find many articles that claim that LLM-based NPCs are right around the corner[57] [58]. This is a narrative worthy of scrutiny. The background research, the creation of the prototypes, and the creation of the design principles leave me with the impression that the task of creating LLM-based NPCs is being underestimated. In this chapter, we detailed issues with RAG and narrative drift.

LLM-based NPCs are not "better" than traditional methods or version 2; they represent an entirely different design paradigm. For many interactions, pre-scripted dialogue or decision trees may still be more reliable, efficient, and narratively consistent. LLMs introduce dynamic, generative interactions that require new methods of control, memory management, and narrative structuring. This shift

does not mean that LLMs will replace traditional methods; rather, it suggests that LLM-based NPCs will require their field of study, tools, and methodologies, separate from both general-purpose LLMs and conventional game AI.

Given these challenges, the future of LLM-based NPCs is unlikely to involve general-purpose models shoehorned into games, but rather purpose-built models trained specifically for interactive storytelling and game dialogue. Until such models exist, developers must carefully weigh the immersion benefits of generative NPCs against the technical and design costs they introduce.

LLMs may be a powerful tool, but without careful integration and research, they risk becoming an over-engineered solution to a problem already solved more efficiently with traditional design techniques. LLM-based NPCs are not an evolution of existing game AI; they are an entirely distinct approach, requiring new tools, new thinking, and ultimately, a new field of study.

10.5 Alternatives to LLM-based NPCs

While the paper is clearly oriented towards LLM-based NPCs, the implementations have garnered some insights in the development of alternative approaches to create immersive and dynamic interactions with NPCs.

NPC conversations that are limited in scope and purpose, as even LLM-based NPC interactions will be, in which the developers desire a dynamic input, can be developed without LLMs. The natural language processing (NLP) field contains plenty of techniques capable of sentence semantic recognition. These are techniques in which the meaning of a sentence is extracted. This can then be matched to an appropriate response. The advantage of this approach is that the challenges involved with LLM response generation do not need to be considered. While this approach is less flexible compared to processing input, it is entirely acceptable to teach the player the appropriate syntax for asking a question and allow that syntax to become part of a developing skill throughout the gameplay.

10.6 The Goal of LLMs Within Video Games

In researching LLM-based NPCs, we naturally ventured out of that singular application and took a glance at other LLM-based integrations in video games. These were focused on dynamic story generation and procedural content generation. These types of projects were often referred to and mentioned in previously online articles detailing the future of LLM integration.

Interestingly enough, we came across minimal research from game developers indicating why a dynamically driven story is desirable. A game story that branches out based on the interest of the player is a great pitch idea, but when dissected, it raises glaring questions. Who is the target audience for such a game? What is the age rating of the game? What structure does a story take? How long is the game? Why would someone want to play this game besides novelty?

Graphic design is essential for player retention[59], and narrative design is an integral part of video games[60]. Thereby conflicting with the narrative that an LLM-based solution would be ideal. Even within the scope of LLM-based NPC interactions, narrative goals reigned supreme in deciding gameplay and immersion, with all other components of the game acting in a supportive fashion. This raises the question of what LLMs will contribute to video games in a meaningful and impactful sense. The technology is undoubtedly capable of new, previously unattainable experiences, but is fraught with speculation and aspirations that either short-sighted or impractical.

Chapter 11

Future Work

11.1 Conversational LLM

In the experimenting phase when converting speech to text and generating a response based on voice input, it became clear that the LLMs that are currently available make underwhelming conversational partners. They seek an objective from the prompt and try to answer it, but they rarely turn a series of questions into a conversation. LLMs rarely showed the ability to foresee or anticipate future questions. As described in the previous chapter, LLMs have been shown to be very user-centric, which can become a significant issue when there are narrative goals to achieve in a story.

It might therefore be prudent to develop a more conversationally oriented LLM that, instead of using literature or other forms of texts meant to be read, is instead trained on texts of which the source is conversational in usage, such as audio or movies.

11.2 Micro-prompting

A socio-cognitive interaction not addressed in the current iteration of the SCNIL is that of ongoing reactions to interactions. Facial expressions are an important tool for expressing and recognizing emotions [61]. A possible avenue of expression these facial expressions while a reaction is occurring is using micro-expressions. Micro-expressions are brief and involuntary facial expressions that occur when people are trying to hide their true feelings or conceal their emotions [62].

An example of this behaviour would be talking to someone and seeing their responses as the message is being presented. For instance, teachers often start a sentence with a certain intention or goal but change the course of the creation of

the sentence based on the response of their students. This can be seen on the face of a student, for example being confused when hearing the first half of the sentence, expressed in a raised eyebrow. The teacher is then likely to clarify the meaning of their thought, in which the second part of the sentence might invoke a sense of understanding, expressed in a hum from the student who indicates understanding.

These micro-expressions, or in the case of playing a role within the socio-cognitive application of the LLM, micro-interactions, are not implemented in the SCNIL. Implementing these types of interactions requires an alternative implementation in which prompts are processed while they are being created. The practice and implementation of this would be different, as the LLM is classifying the prompt based on a series of micro-expressions. Micro-expressions last less than a one fifth of a second [63]. To perform to the degree of such a low-latency scenario, it might be that simpler and thereby faster LLMs need to be made for this type of classification, as the responses need to be generated faster than the user can finish a prompt.

From a cursory glance of the available LLMs, none came forward capable of supplying the required level of speed. A future study could develop this type of LLM or some other type of architecture capable of rapidly providing responses. Furthermore, it is not necessary to implement this via an LLM, as other methods of neural language processing might be more applicable.

11.3 Socio-cognitive Interaction Limitations

The thesis is based on an assumption which is worth criticizing and discussing. The assumption made is that everybody is capable of and willing to indulge in the same level or degree of socio-cognitive interactions. Meaning that in an ideal world, players are willing to engage with the video game through methods of interactions more socio-cognitive in nature and have the capacity to do so.

The willingness of a player to engage in a video game containing socio-cognitive interactions is not something worth exploring within this thesis, as the primary goal of video games is entertainment by choice. Implying that customers have the consideration to play a game or not, leading to a natural filtering of players with those who are not interested in this type of gameplay whilst drawing in others. Not all those who play video games enjoy horror games, and thereby the classification that a video game is a horror game will both dissuade and persuade. Whether there is a market for customers interested in these types of socio-cognitive interactions is also beyond the scope of this thesis.

Contrary to the willingness of players, the question regarding the ability to engage in a certain socio-cognitive manner is of vital importance when developing a video game. Factors such as age have shown to be contributing factors[64] towards a decreased level of social cognitive capacity[65]. Cultural differences also have a significant impact on socio-cognitive perspective. Furthermore, there are other factors that can inhibit the capacity of a player to engage in a certain socio-cognitive manner[66]. This is to say that it is short-sighted to assume that all players can perform in the expected manner a game designer might desire.

To address this, a future study can investigate this aspect of socio-cognitive video game interactions. This study could then hopefully identify underlying issues with the SCNIL or the design principles, resulting in a more refined manner. It would be preferable to perform a study after LLM-based socio-cognitive interactions have become more mainstream, as it is the opinion and experience of the author that novelty in video games is excellent in masking underlying issues, but that these eventually become abundantly clear.

11.4 Large-scale User Testing

The primary limitation of this study was the lack of large-scale user testing. A future study that performs a large-scale user study could evaluate the effectiveness of different SCNIL levels in real-world gaming environments. This would allow for a vast quantity of qualitative feedback on immersion, engagement, and narrative coherence. Other than the qualitative feedback, quantitative data on player behaviour and interaction patterns could also be collected.

Large-scale testing could provide empirical validation of the SCNIL model and quantify the impact on player experience that could help refine the design principles for LLM-based NPC interactions.

Conducting large-scale testing with LLM-based NPCs is a sophisticated endeavour. A key challenge is how to factor in the variability in player interactions, as participants may approach NPCs with diverse expectations, communication styles, and immersion. While limiting participants to SCNIL-driven LLM-based NPC interactions under controlled conditions might alleviate variability, they will introduce the bias of the researchers into the results. It is therefore the belief of the author that an iterative process wherein each level is examined rather than one large study encompassing all levels would be beneficial. Benchmarking LLMs is often done by predefined tests that might not represent the method of interaction performed by players, therefore refining the methodology with which to judge these types of LLM integrations should take precedence.

11.5 Multi-NPC

This thesis scoped itself to experimentation with a single NPC, as this was a more manageable constraint with clearer goals and expectations. The possibility of multiple NPCs interacting with one another with the player is an interesting dynamic worth exploring, as it opens up theories on group dynamics. A future study could take this as a point of exploration and determine the limits that can occur in the narrative cohesion between multiple NPCs.

Chapter 12

Conclusion

The integration of LLM-based NPCs presents both novel prospects and significant obstacles for game development. SCNIL, a framework that categorizes NPC interactions across ten levels, is the culmination of this thesis's exploration of the technical, design, and socio-cognitive complexities of employing locally run LLMs to create interactive NPCs. By developing and testing prototypes at each level, this research has identified key design challenges, trade-offs, and opportunities for implementing LLM-driven NPCs.

This chapter addresses the research questions by reflecting on the findings and synthesizing them into a set of actionable design principles that will guide future development in this evolving field. Before we address the main research question, we will answer the sub-research questions.

1. What technical and design challenges arise when employing local LLMs for NPC interactions in video games, and how can iterative prototypes effectively address these constraints?

The primary challenge with LLM-based NPCs is not simply generating responses, but structuring interactions in a way that ensures coherence, immersion, and efficiency. Unlike traditional game AI, LLMs require continuous access to relevant game context, which introduces overhead in memory management, retrieval strategies, and computational resources. Even at lower SCNIL levels, maintaining coherent, relevant, and timely responses turned out to be a non-trivial task, requiring:

- Structured knowledge retrieval mechanisms (e.g., RAG) to prevent hallucinations and ensure accuracy.
- Memory management systems to track player interactions and maintain NPC consistency.

– Optimized models to balance real-time inference speeds with computational constraints.

The iterative prototyping process revealed that while locally run LLMs remove cloud-based latency and pricing concerns, they still pose significant scalability challenges, requiring further refinement of contextual processing and dialogue filtering techniques from the video game. These findings indicate that LLM-based NPCs need a robust backend infrastructure, not just a conversational model, to function effectively. It's important to remember that both local LLMs and cloud-based LLMs may have the same problems with getting good answers. While it's possible to make LLMs less computationally intensive from the standpoint of memory and inferencing, it's not the limiting factor at the time of writing.

2. How can socio-cognitive interaction methodologies be translated into diverse prototype designs for local LLM-based NPCs, and what trade-offs arise regarding player engagement and system complexity?

The SCNIL model provided a structured way to translate socio-cognitive principles into video game interactions, ranging from simple text exchanges to advanced multimodal collaboration. The prototype findings found that higher levels of interaction complexity required significantly more structured information management, as LLMs do not naturally track context over time.

The main trade-off was between open-ended generative flexibility and structured game design. While dynamic, player-driven dialogue enhances immersion, it also creates narrative unpredictability, potential inconsistencies, and player confusion. Significant insights gained from the prototypes include:

– Text-based interactions (Levels 1–3) are relatively straightforward but require game-specific constraints to remain useful.

– Multimodal interactions (Levels 4–6) introduce usability challenges, as not all input methods feel natural or improve gameplay.

– Collaborative and emergent interactions (Levels 7–9) enhance immersion but require clear boundaries to prevent NPCs from feeling either too rigid or too chaotic.

The findings indicate that LLMs work best when they enhance socio-cognitive engagement rather than replace structured game mechanics.

3. Which in-game knowledge representations and prompting strategies mitigate hallucinations in local LLM-based NPCs, ensuring coherence and alignment with game lore?

LLM-based NPCs are prone to hallucinations, generating inaccurate or misleading responses that conflict with established game lore. The study explored multiple techniques to mitigate this issue, including:

- Retrieval-Augmented Generation (RAG): Providing NPCs with real-time access to curated game-specific data.
- System-level constraints: Restricting NPCs to predefined world knowledge to prevent AI-driven improvisation that disrupts immersion.
- Memory and tracking mechanisms: Ensuring that NPCs retain useful information while avoiding unnecessary long-term storage issues.

Findings suggest that hallucination mitigation remains an ongoing challenge, and future work should explore purpose-built game LLMs with better contextual grounding capabilities.

What design principles can be established to inform the design of locally run, LLM-based NPC interactions in video games, encompassing a spectrum of socio-cognitive interaction methods?

From the findings, several key design principles emerged to guide the development of LLM-based NPCs:

- **Context is key:** LLM-based NPCs require a robust context-management system to ensure coherent and relevant responses.
- **Balance generative flexibility with structure:** NPCs should exhibit adaptive behaviours, but within a well-defined design framework to avoid narrative disruptions.
- **Scalability is a major constraint:** Local LLM-based NPCs demand significant computational resources, making optimization strategies essential.
- **Modality should serve gameplay, not dictate it:** Not all input methods improve gameplay, and some can be more cumbersome than immersive.
- **LLM-based NPCs should express friction and autonomy:** Instead of acting as passive assistants, they should have independent goals and motivations to reinforce realism.

- **Experimentation is key:** LLM-based NPCs are still a developing technology, and game developers should feel encouraged to challenge existing principles and experiment with new approaches.

These principles offer a foundation for game developers seeking to integrate LLM-based NPCs while maintaining gameplay coherence, technical feasibility, and socio-cognitive engagement.

Final Thoughts & Future Directions

This thesis demonstrates that LLM-based NPCs hold immense potential for creating richer, more interactive game worlds, but also highlights significant design and technical challenges that must be addressed. The field is still in its early stages, and purpose-built game LLMs will likely be required to fully realize scalable, immersive NPC interactions.

While LLM-driven NPCs are not yet a solved problem, the findings of this study provide a structured framework, tested insights, and actionable design principles to help bridge the gap between current limitations and future innovations in game NPCs. The potential for more dynamic, responsive, and socially aware game worlds makes further research in this field both valuable and necessary. The journey toward intelligent, responsive NPCs is just beginning, and it happens, one prompt at a time.

Bibliography

[1] Roberto Gallotta, Graham Todd, Marvin Zammit, Sam Earle, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. Large language models and games: A survey and roadmap. *IEEE Transactions on Games*, page 1–18, 2024.

[2] Xiangyu Peng, Jessica Quaye, Sudha Rao, Weijia Xu, Portia Botchway, Chris Brockett, Nebojsa Jojic, Gabriel DesGarennes, Ken Lobb, Michael Xu, Jorge Leandro, Claire Jin, and Bill Dolan. Player-driven emergence in llm-driven game narrative, 2024.

[3] Dekun Wu, Haochen Shi, Zhiyuan Sun, and Bang Liu. Deciphering digital detectives: Understanding llm behaviors and capabilities in multi-agent mystery games. *arXiv preprint arXiv:2312.00746*, 2023.

[4] Xinyu Mao, Wanli Yu, Kazunori D Yamada, and Michael R. Zielewski. Procedural content generation via generative artificial intelligence, 2024.

[5] Alessandro Marincioni, Myriana Miltiadous, Katerina Zacharia, Rick Heemskerk, Georgios Doukeris, Mike Preuss, and Giulio Barbero. The effect of llm-based npc emotional states on player emotions: An analysis of interactive game play. In *2024 IEEE Conference on Games (CoG)*, pages 1–6, 2024.

[6] Sudha Rao, Weijia Xu, Michael Xu, Jorge Leandro, Ken Lobb, Gabriel DesGarennes, Chris Brockett, and Bill Dolan. Collaborative quest completion with llm-driven non-player characters in minecraft, 2024.

[7] Filippo Chiarello, Vito Giordano, Irene Spada, Simone Barandoni, and Gualtiero Fantoni. Future applications of generative large language models: A data-driven case study on chatgpt. *Technovation*, 133:103002, 2024.

[8] Mingjin Zhang, Jiannong Cao, Xiaoming Shen, and Zeyang Cui. Edgeshard: Efficient llm inference via collaborative edge computing, 2024.

[9] Gad Benram. Understanding the cost of large language models (llms), February 2024.

[10] The AllBusiness.com Team. Prompt (ai prompt). *TIME*, April 2025. Accessed: 2025-04-14.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[12] Gabrijela Perković, Antun Drobnjak, and Ivica Botički. Hallucinations in llms: Understanding and addressing challenges. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 2084–2088, 2024.

[13] George Foster. How star wars jedi: Survivor pulled off the greatest betrayal of 2023, December 2023. Accessed: 2025-04-14.

[14] Michael Yin, Emi Wang, Chuoxi Ng, and Robert Xiao. Lies, deceit, and hallucinations: Player perception and expectations regarding trust and deception in games. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[16] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[17] Mike Isaac. Openai courts investors as it seeks trillions of dollars to build a.i. superintelligence. https://www.nytimes.com/2024/09/27/technology/openai-chatgpt-investors-funding.html, September 2024. The New York Times.

[18] Microsoft Corporation. Microsoft volume licensing service level agreement for microsoft online services (worldwide english, april 1, 2025). *Microsoft Word Document*, April 2025. Accessed: 2025-04-14.

[19] Andrew King. Kingdom come: Deliverance 2 proves that niche is now mainstream, March 2025. Accessed: 2025-04-01.

[20] Martin Sas and Victoria Hendrickx. It's alive! the rise of generative non-player characters in video games. *CiTiP Blog*, May 2024.

[21] Andrey Krekhov, Sebastian Cmentowski, Katharina Emmerich, and Jens Krüger. Beyond human: Animals as an escape from stereotype avatars in virtual reality games. *arXiv preprint arXiv:1907.07466*, 2019.

[22] Robert A. Wilson and Frank C. Keil, editors. *The MIT Encyclopedia of the Cognitive Sciences (MITECS)*. The MIT Press, Cambridge, MA, 1999.

[23] Jay Friedenberg and Gordon Silverman. *Cognitive Science: An Introduction to the Study of the Mind*. Sage Publications Ltd, Thousand Oaks, CA, 2006. Paperback.

[24] Esmeralda G. Urquiza-Haas and Kurt Kotrschal. The mind behind anthropomorphic thinking: attribution of mental states to other species. *Animal Behaviour*, 109:167–176, 2015.

[25] Nicholas Epley. A mind like mine: The exceptionally ordinary underpinnings of anthropomorphism. *Journal of the Association for Consumer Research*, 3(4):591–598, 2018.

[26] Paul Thagard. *Mind: Introduction to Cognitive Science*. MIT Press, Cambridge, MA, US, 2nd edition, 2005. Also available in paperback: ISBN 0-262-70109-X.

[27] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, 1994.

[28] Marvin Minsky. *The Society of Mind*. Simon and Schuster, 1988.

[29] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[30] Jerry A. Fodor. *The Language of Thought*. Harvard University Press, 1975.

[31] Zenon W. Pylyshyn. Computational models and empirical constraints. *Cognition*, 28(1-2):3–71, 1988.

[32] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1):139–159, 1991.

[33] Manoj Deshpande and Brian Magerko. Embracing embodied social cognition in ai: Moving away from computational theory of mind. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024.

[34] Ezequiel Di Paolo and Hanne De Jaegher. Participatory sense-making. *Phenomenology and the Cognitive Sciences*, 6(4):485–507, 2007.

[35] Manoj Deshpande, Milka Trajkova, Andrea Knowlton, and Brian Magerko. Observable creative sense-making: A method for quantifying improvisational co-creative interaction. In *Proceedings of the ACM SIGCHI Conference on Creativity and Cognition*, 2023.

[36] Michael Yin and Robert Xiao. Press a or wave: User expectations for npc interactions and nonverbal behaviour in virtual reality. *Proceedings of the ACM on Human-Computer Interaction*, 8:1–25, 10 2024.

[37] Tarpley Hitt. Meet the mormon college student behind the viral artificial intelligence game AI Dungeon, 2019. Accessed: 2025-05-12.

[38] Nick Walton. How we scaled ai dungeon 2 to support over 1,000,000 users, 2020. Accessed: 2025-05-12.

[39] Joshua McCoy, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. Comme il faut: A system for authoring playable social models. 01 2011.

[40] Li Song. Llm-driven npcs: Cross-platform dialogue system for games and social platforms, 2025.

[41] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, Jul 2024. Preprint.

[42] Stanford CS224N, Xuanzi Chen, Zhengjia Huang, and Ryan Li. Words and wins: Enhancing game play with llm fine-tuning by rl.

[43] Chengpeng Hu, Yunlong Zhao, and Jialin Liu. Game generation via large language models, 2024.

[44] Samuel Rhys Cox and Wei Tsang Ooi. Conversational interactions with npcs in llm-driven gaming: Guidelines from a content analysis of player feedback. In Asbjørn Følstad, Theo Araujo, Symeon Papadopoulos, Effie L.-C. Law, Ewa Luger, Morten Goodwin, Sebastian Hobert, and Petter Bae Brandtzaeg, editors, *Chatbot Research and Design*, pages 167–184, Cham, 2024. Springer Nature Switzerland.

[45] Juan Linietsky, Ariel Manzur, and the Godot Engine community. Introduction. https://docs.godotengine.org/en/stable/about/introduction.html. Accessed: April 1, 2025.

[46] Georgi Gerganov and ggml-org contributors. llama.cpp: Llm inference in c/c++. https://github.com/ggml-org/llama.cpp, 2023. Accessed: 2025-04-15.

[47] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

[48] J Zhang. Tinyllama. https://github.com/jzhang38/TinyLlama, 2024.

[49] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren,

Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[50] OpenAI. Tokens, 2025. Accessed: 2025-04-01.

[51] Marc Brysbaert. How many words do we read per minute? a review and meta-analysis of reading rate. *Journal of Memory and Language*, 109:104047, 2019.

[52] Kurt Reusser and Christine Pauli. Co-constructivism in educational theory and practice. In James D. Wright, editor, *International Encyclopedia of the Social Behavioral Sciences (Second Edition)*, pages 913–917. Elsevier, Oxford, second edition edition, 2015.

[53] Juliana Schroeder. Ending conversation is a fraught endeavor. *Trends in Cognitive Sciences*, 25(7):552–553, 2021.

[54] Dave Bergmann. What is a context window?, 2024. Accessed: 2025-04-01.

[55] Google AI. Long context $\|geminiapi, 2025. Accessed : 2025 - 04 - 01.$

[56] Robert Lakatos, Peter Pollner, Andras Hajdu, and Tamas Joo. Investigating the performance of retrieval-augmented generation and fine-tuning for the development of ai-driven knowledge-based systems, 2024.

[57] Martin Sas and Victoria Hendrickx. It's alive! the rise of generative non-player characters in video games, 2024. Accessed: 2025-04-01.

[58] GoodAI. Ai people: Announcing the next evolution of gaming ai npcs, 2024. Accessed: 2025-04-01.

[59] Mine Okur, Raif Kızıl, and Elif Atamaz. The art of graphic design in video games: beyond the visual. *Revista Amazonia Investiga*, 2024.

[60] Anastasiia Poliakova and Kateryna Lut. Narrative in video games' verbal mode. *Scientific Journal of Polonia University*, 2023.

[61] Paul Ekman. Facial expression and emotion. *American psychologist*, 48(4):384, 1993.

[62] Thuong-Khanh Tran, Quang-Nhat Vo, Xiaopeng Hong, Xiaobai Li, and Guoying Zhao. Micro-expression spotting: A new benchmark, 2020.

[63] Wen-Jing Yan, Qi Wu, Jing Liang, Yu-Hsin Chen, and Xiaolan Fu. How fast are the leaked facial expressions: The duration of micro-expressions. *Journal of nonverbal behavior*, 37:217–230, 2013.

[64] Lucas J. Hamilton, Amy N. Gourley, and Anne C. Krendl. They cannot, they will not, or we are asking the wrong questions: Re-examining age-related decline in social cognition. *Frontiers in Psychology*, 13, 2022.

[65] Mei-Liang Chen and Chieh-Peng Lin. Assessing the effects of cultural intelligence on team knowledge sharing from a socio-cognitive perspective. *Human Resource Management*, 52(5):675–695, 2013.

[66] Helen Tager-Flusberg. Evaluating the theory-of-mind hypothesis of autism. *Current Directions in Psychological Science*, 16(6):311–315, 2007.