Title of Project	Dynamic NPC Dialogue System with LLM Integration
Name of Student	Conor Ryan
Student Number	K00286377
Date	06/10/2025
Word Count	1387

I declare that no element of this assignment has been plagiarised:

Student Signature	Conor Ryan

1 Introduction

Dialogue trees play a massive role in how players interact with and perceive a game world. In most narrative driven games, conversations with non-player characters are used to build up the story, expose character motivations, and flesh out the world to make it feel alive. However, as much as games have become more realistic and visually advanced throughout the years, the systems used for dialogue in games has mostly remained the same. In most games these days, every individual line of dialogue must be manually written by a developer, and the players dialogue options are limited to a small number of scripted responses. This results in games becoming predictable and repetitive as the player spends more time in game.

This project, "Dynamic NPC Dialogue System with LLM Integration," will focus on investigating how a Large Language Model (LLM) could be used to make game dialogue more varied and interesting. The idea of the project is to create a Unity system which allows non-player characters to respond to the player in a varied but natural way, using a Large Language Model to generate lines in real time. The player will still be presented with a choice of predefined dialogue options to keep the story intact, but the NPCs response will be generated dynamically using an AI prompt which provides context such as the current scene, story, and the NPCs personality type.

When the player triggers a conversation in game, this system will collect contextual data from the game and use it to create a structured prompt which is sent to an LLM through an API. The LLM in turn will generate a unique dialogue response which will be displayed on screen to the player. As potential stretch goal, this project will also experiment with the ElvenLabs Text-to-Speech API, which could enhance the NPCs response with natural sounding voices with emotion.

The goal is to make dialogue feel reactive and less repetitive, while still letting the developer guide the story in a consistent manner. This system could save game writers time, reduce the need for complex dialogue trees, and improve the players immersion in the game. It also branches into a new area of game design, where tools like LLMs are being used to create more dynamic content for games.

1.1 Problem Definition and Background

1.1.1 Aim and Research Questions

This project aims to deliver a reusable Unity Package which allows generation of dynamic NPC dialogue using a Large Language Model, creating flexible, context aware NPC responses while retaining developer control over the game's narrative. See below for research questions this project aims to address:

1. How can information about NPC personality, story progression, and game state be used to create structured prompts for an LLM?
2. How can a developer retain narrative control while still allowing the LLM to create dynamic and realistic responses?
3. What's the best way to design the tools interface so that other developers can customize and use the system easily?
4. How can the system deal with issues such as latency or loss of connection during gameplay?
5. How will voice generation work with the system and what benefit does it offer in comparison to text only dialogue?

1.1.2 Background and Context

Contemporary dialogue systems in games are designed using branching dialogue trees, in this system every individual line of dialogue is predefined and linked together with conditions or player choices. This method gives game writers full control over the game's narrative, but it is time consuming and hard to scale. Each new player choice means more branches, more writing, and more testing. This results in many games reusing the same lines across multiple characters, or including a small number of lines per character, which can affect the players immersion.

The recent and rapid development of Large Language Models like GPT or Grok has provided an opportunity for creating game dialogue in real time. Such models can generate text which sounds both natural and context aware when given the right prompt. Integrating these models into game engines like Unity comes with several challenges like managing performance, prompt format, and ensuring generated dialogue is consistent with the game's scenes and lore.

Unity is ideal for this kind of experiment due to its flexibility. It provides necessary tools like 'UnityWebRequest' which makes it simple to connect to APIs, and the UI Toolkit which can display dialogue text on screen. The project will use these tools to build a system that can be reused and adapted by other developers.

1.1.3 Problem Definition

The problem this project aims to solve is the predictable and static nature of non-player character dialogue in most modern games. Even triple A games with a massive amount of resources struggle with repetitive conversations and limited responses, which can pull the player out of the game. Current dialogue systems have several constraints:

- **Repetitiveness:** Non-player characters repeat the same lines, often when it doesn't make sense.
- **Lack of Reactivity:** NPCs don't often respond to player behaviour or changing game states.
- **Reduced Replay Value:** When the player has seen all dialogue, conversations become much less interesting.
- **High Creation Cost:** Each individual line of dialogue must be prewritten and tested.
- **Limited Emotional Delivery:** In lower budget titles, dialogue can feel flat with no variation in tone or voice.

By using prewritten player dialogue with LLM generated non-player character responses, this project aims to correct these issues while keeping the story consistent.

1.1.4 Scope of the Project

The project will aim to create a Unity based dialogue system with these key features:

- Integration with a Large Language Model API to generate text in real time.
- Gathering of game context data like scene, character traits, and game state to build prompts.
- Tools which allow other developers to define dialogue templates and parameters in the Unity Editor.
- An offline fallback mode with prewritten dialogue to safeguard against connection loss and API issues.

Several stretch goals have been identified for the project these include:

- ElvenLabs Text-to-Speech API integration for voice generation.
- Support for different Large Language Models i.e. GPT, Grok, Gemini provided by OpenRouter a unified API for a multitude of LLMs.
- Support for basic conversation memory to allow responses to remain consistent in long form interactions.
- Customizable character personality templates which influence tone and vocabu-lary.

1.1.5 Intended Audience and Stakeholders

The project is aimed mainly at game developers, particularly those who work with Unity and want to add dynamic and reactive dialogue to their games without the need to write thousands of dialogue lines manually. It will also be useful to players who would benefit from more varied and immersive conversations.

1.1.6 Approach and Methodology

The development of this project will follow an iterative process, with frequent testing and feedback from supervisors at each stage. See below for an explanation of each stage:

1. **Research Phase:** Review and learn about existing dialogue systems and methods to integrate AI.
2. **Design Phase:** Create software architecture diagrams, visualize data flow, and define structure for prompts.
3. **Implementation Phase:** Build the dialogue system, prompt generator, and API communication scripts.
4. **Testing Phase:** Undergo tests to evaluate dialogue quality and responsiveness of the system.
5. **Evaluation and Refinement:** Analyse performance impact, make improvements where necessary and document findings.

1.1.7 Project Plan

Lorem Ipsum

1.1.8 Assumptions

The main assumptions for the project to work are the following:

- A reliable internet connection for API calls.
- Access to functional API keys for the selected LLM and TTS services.
- Unity 2022 or later as the main engine.
- Latency in responses will be manageable within gameplay.

1.1.9 Expected Outcomes

By the end of this project, the main goal is to deliver a working Unity package which allows developers to easily integrate AI-driven non-player character dialogue into their games. The system should create believable and varied which react to game state, while being easy to configure using the Unity editor.

If the project is successful, it will demonstrate a practical use for modern AI technology in game content generation, while also reducing the workload involved with writing dialogue for games, and enhancing the player's experience.

2 Literature Review and Research

2.1 Personalized NPC Dialogue and Character Consistency

Recent studies show that there is a growing interest in enhancing how NPCs communicate with players in video games. Particularly in games which rely on heavy story elements and character interactions. Traditionally dialogue systems in games are dependent on branching dialogue trees, these structures allow developers to keep a consistent narrative structure but are limited in how reactive they can be. They also struggle to properly reflect player actions or fast changing game states, which leads to repetitive dialogue and predictable behaviours.

Lie et al. (2025) designed a narrative framework to enhance NPC dialogue by making it more personal and consistent which each character's background and personality [1]. Their study emphasized a major challenge with modern NPC dialogue. That large language models can generate text which seems natural but tends to drift away from the characters intended voice or knowledge base, they describe this problem as "character hallucination". To combat this problem their framework uses a combination of static knowledge about a given character, like personality traits, backstory, or world view, with dynamic knowledge provided by the players previous interactions.

Their framework utilizes multiple interesting methods which help guide the model into more believable character behaviour. For instance, it saves short-term memories in a knowledge graph, which is like an editable structure which the system can update as new interactions occur in the game. They store long-term memories in a vector database, which makes it efficient to retrieve older data without causing slowdowns. The framework also uses a method called AMR (Abstract Meaning Representation) parsing to interpret the players inputs more reliably. Instead of using a raw text prompt, AMR breaks the input down into a logical structure cutting away unnecessary detail and increasing the chances of the NPC responding accurately.

Lie et al.'s study demonstrates that NPC consistency is dependent on more than a well-structured prompt. Their shows that NPC responses become more believable when the system can utilize NPC personality, short- and long-term memory, and structured knowledge in way that stays within the boundaries of the game world. While this is a more complex approach, it highlights the necessity of controlling how the LLM interprets context. A topic which is becoming increasingly relevant in modern game development. For my own project, the concepts around structured data, memory, and character constraints directly relate to creating NPCs which respond in coherent way when using an API to generate dialogue.

2.2 LLM Capabilities & Limitations

With the recent development of LLM models like GPT, many developers are questioning how they might replace or enhance dialogue systems. It's clear that LLMs can generate natural sounding responses based on small pieces of context, making them suitable for games which need more reactive characters. But they also come with risks, particularly when used in real time.

A 2024 research paper studied the performance of LLMs when they are used to role-play NPCs in video games [2]. This study performed a comparative evaluation of multiple models in various aspects such as character consistency, responsiveness, and ability to follow role instructions. The results show that while LLMs can generate engaging dialogue, they usually fail to stay consistent and in character in long form conversations. Some models changed tone mid conversation or began providing information which the character would not know, which is a major problem for narrative driven games.

The study demonstrated the importance using well structured prompts in combination with properly defined personas. Most models performed better with detailed instructions and example responses, indicating that the LLMs reliability depends on how developers design the prompt structure. Additionally, the study states that built in safety features and refusal behaviours inherent in modern LLMs sometimes interfere with gameplay. For example, an NPC may refuse to answer a simple question, which it mistakenly interpreted as inappropriate. Limitations like these make it clear that developers should implement fallback responses and filtering layers to maintain reliability.

OpenAI's GPT-4 Technical Report (2023) supports these observations in demonstrating the known limitations of LLMs [3]. Even though GPT-4, was significantly more capable than older versions, it still maintained issues like hallucination, limited memory over longer conversations, and random illogical reasoning. Their report states that the model's performance varies depending on a prompt's length, structure, and context window size. For video games these factors are extremely important because bad responses can break the players immersion.

These studies show that LLMs are powerful tools for generating dynamic dialogue, but they must be used carefully. Video games need consistency, controlled information, and predictable behaviour, which also happen to be LLMs biggest weaknesses. This reaffirms my approach to include an offline fallback mode, structured prompt templates, and limits to randomness in the LLM's output settings. It also means that developers should avoid using LLMs for long-term conversations unless they implement a system to track game context externally.

2.3 Context, Memory, and Behaviour Control in LLM-driven NPCs

A major topic which reoccurs in modern studies is the importance of controlling how the LLM interprets context. When creating NPC dialogue, the consistency is reliant on not only the model used, but also how developers can guide its behaviour. This is particularly important for games, where the NPCs personality, traits, and world knowledge need to remain stable.

The 2024 NPC assessment study shows similar results to the previously discussed studies, that LLMs performed much better when given strong personality descriptions and clear constraints [2]. This suits the approach I will use for this project, where structured prompts provide details about personality traits, scene information, and the NPCs emotional tone. The study also showed that more reliable responses are achieved when the LLM is told what it should avoid doing, like references to real world facts which contradict game lore.

Lie et al. (2025) emphasized that memory has an influence on the characters behaviour [1]. Instead of using the model's internal context window, which resets each time for API based calls, their system stores important information separately and feeds it back into the prompts when necessary. This method can help characters behave as if they remember earlier interactions, which is essential for immersion. Even though my project will use a simpler version of memory, the concept of storing context externally lines up with the industry's best practices and helps mitigate inconsistent dialogue.

The GPT-4 report solidifies the idea that an external context structure is essential [3]. Because the model can't track long histories or self-correct hallucinations, giving clear constraints and filtering outputs is a necessity for any system which relies on consistent behaviour. This is particularly important when an LLM needs to interact with prewritten dialogue. The LLM must stay on topic and avoid contradicting earlier events, otherwise the conversation will feel disconnected from the game.

These studies demonstrate that proper NPC dialogue generation needs a balance between dynamism and controlled context. Developers must carefully guide the LLM to maintain personality, memory, and consistency. This project will utilize these concepts by combining structured prompts, personality templates, and fallback modes to generate NPC dialogue that is reactive without breaking the games context.

2.4 References

[1] X. Liu, Z. Xie, and S. Jiang, "Personalized Non-Player Characters: A Framework for Character-Consistent Dialogue Generation," AI, 2025.
[2] Author(s), "Assessing Large Language Models for Role-Playing Chat-based NPC Agents in Games," 2024.
[3] OpenAI, GPT-4 Technical Report, 2023.