Document # K00286377-PID
Revision # 1.0

Project Plan

NPC Dialogue System with LLM Integration for Unity

Produced by: Conor Ryan Date: 23/10/2025

Table of Contents

Introduction

The NPC Dialogue System with LLM Integration for Unity project aims to develop a reusable Unity Package which allows non-player characters (NPCs) to deliver dynamically generated responses by using a Large Language Model (LLM). The system combines prewritten player dialogue with AI-generated NPC dialogue, to enhance player immersion and reduce developer work in narrative games.

References

Technologies, U. (n.d.-b). Unity - Manual: UI Toolkit.
https://docs.unity3d.com/6000.1/Documentation/Manual/UIElements.html

Create speech | ElevenLabs Documentation. (n.d.). ElevenLabs Documentation.
https://elevenlabs.io/docs/api-reference/text-to-speech/convert

Technologies, U. (n.d.-b). Unity - Manual: UnityWebRequest.
https://docs.unity3d.com/540/Documentation/Manual/UnityWebRequest.html

Openai.com. (2025). OpenAI Platform.
https://platform.openai.com/docs/api-reference/responses/create.

Project Goals

Business Goals:

- Enhance player immersion with dynamic and context aware dialogue.
- Demonstrate the practical use of generative AI tools in game development.
- Provide a modular and reusable dialogue system for developers to reduce time spent manually scripting.

Technical Goals:

- Integrate an LLM API into Unity to provide real time text generation.
- Allow for dynamic prompt creation based on game scene, context, and NPC traits.
- Implement an offline fallback mode and optional voice generation with ElevenLabs TTS API.

Major Features

- AI-driven NPC dialogue generation via LLMs.
- Contextual prompt building which integrates with Unity game state.
- Unity Editor tools for customizing dialogue templates and NPC personality profiles.
- Offline Fallback mode which uses prewritten text.
- Optional ElvenLabs text-to-speech integration to provide emotional dialogue delivery.

Project Hardware & Software Components

Supported Hardware Devices

- Laptop or PC capable of running Unity 2022+.
- An internet connection to facilitate API communication.

Software Partitions

The responsibilities and architectural status of each software partition are described below.

Partition/Package	Responsibilities	Architecture Status
Core Dialogue System	Manage dialogue state and trigger events	Foundational
Prompt Builder	Collect context data and generate structured prompts	Service
API Connector	Handles responses and requests to the LLM API.	Integration Adapter
Editor Tools	GUI for configuration and NPC personality templates	Tooling
TTS Module	Integrate ElevenLabs TTS to facilitate audio generation	Integration Adapter

Environment

System Environment

- Unity Engine 2022+, LLM API such as OpenAI or OpenRouter, and ElevenLabs API.

Software Development Environment

- Languages: C#, C++, Unity Scripting.
- Tools: Unity Editor, Visual Studio, GitHub for version control.
- APIs: OpenAI or OpenRouter for text generation, ElvenLabs for text-to-speech.

Development Strategies

Development of the project will follow an iterative approach, aiming for incremental builds and feedback:

- Frequent integration and testing.
- Feature-based iteration (research, prototype, integrate, test).
- Weekly progress updates with supervisor.

Project Organization

Common Communications Tools

- Outlook, GitHub Issues, Trello for task tracking.

Hardware/Software Development

- No hardware development required.

Hardware/Software Integration

- Unity Editor integration with LLM/TTS APIs.

Resources

Software Modeling & Programming Effort Estimates

Package	Estimated No. of Classes	Total Effort Estimate *
Core Dialogue	5	3 weeks
Prompt Builder	3	2 weeks
API Connector	2	1 week
Editor Tools	5	3 weeks
TTS Module	2	2 weeks

Total (minimum) effort: 17 classes
11 weeks

Organizational Effort Estimates

The following table summarizes the estimated effort to reach Product Release.

Organisation Group	Total Estimated Effort
Programming/Development	11 weeks
Testing & Evaluation	2 weeks
Documentation	1 week
Total non-hardware Effort:	14 weeks

Software Staff Assignments

Qty.	Skill/Position	Candidate	Assigned?
1	Programmer/Developer	Conor Ryan	Y
1	Academic Supervisor	Eugene Kenny	Y

Capital Needs

- Development PC/Laptop (existing).
- Internet Connection (existing).

Expense Needs

- API subscriptions (OpenAI or OpenRouter, and ElevenLabs).

Software Development Schedule

Use Case/Scenario Estimates

Category 1: No exceptions

1. Player initiates a conversation with an NPC and selects a dialogue option.
2. LLM returns a valid response.
3. Dialogue is displayed.

Category 2: Recoverable exceptions

1. Player initiates a conversation with an NPC and selects a dialogue option.
2. API can't deliver a response due to timeout or latency.
3. Fallback to prewritten offline response.

Category 3: Non-Recoverable exceptions

1. Player initiates a conversation with an NPC and selects a dialogue option.
2. API can't deliver a response due to an invalid key.
3. Disable generative dialogue, fallback to offline mode, and notify the developer.

Category 4: Fatal exceptions

1. Player initiates a conversation with an NPC and selects a dialogue option.
2. API can't deliver a response due to an error.
3. Offline fallback mode fails to deliver a response due to an unknown error.

Iteration Plan

Iteration Definitions

Iteration #0	Iteration #1	Iteration#2
Select tools, plan architecture, and research existing documentation.	Implement the core dialogue manager and create mock data.	Integrate an LLM API and a context-based prompt builder.
Iteration #3	Iteration #4	Testing & Review
Build Unity Editor Tools for dialogue configuration.	ElevenLabs TTS Integration.	Testing, and optimization. Review Usability of the system & achievement of stated goals.

Summary Schedule and Milestones

Iteration	Duration	Start Date	End Date	Milestone
0	2 weeks	Oct 2025	Nov 2025	Research & Planning
1	3 weeks	Nov 2025	Dec 2025	Core Dialogue Manager
2	1 weeks	Jan 2026	Feb 2026	LLM Integration
3	3 weeks	Mar 2026	Mar 2026	Editor Tools
4	2 weeks	Apr 2026	Apr 2026	Testing & Documentation

Product Maintenance

- Later version may include multi LLM support via Open Router, and better response times.

Software Diagnostics and Troubleshooting

- The system will include logging for API responses, error reporting, and debugging within Unity Editor.

Problem Reporting and Correction

- Bugs will be tracked using GitHub Issues, Severity Levels: Low (typo/UI), Medium (API errors), High (System Crashes).

Product Testing & Certification

- All testing will be performed in the Unity Editor. Supervisor will provide quality reviews.

Installation Plan

- Installation will be via Unity Package Manager.

Documentation Plan

End User Documentation

- Developer Documentation (readme and PDF guide) will explain the setup, API configuration, and customization options.

End User Training Plan

- Video guide and PDF Documentation included in final submission.