iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82707

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

LLM-Driven NPC Interaction for Enhanced Immersion in Videogame Environments

Viraj Zende[1], Vedant Patil[2], Kanchan Swami[3], Abhishek Veer[4], Disha Wankhede[5]
Department of Computer Engineering, Vishwakarma Institute of Technolgy, Pune, India

Abstract: ***This research aims to incorporate large language models (LLMs) in the real-time gaming experience, adding depth to gameplay mechanics and enhancing player immersion through rich dialogue. In existing videogames, non-player character (NPC) interactions are mostly static due to their scripting and simple state machine logic. In this research, we showcase a sample video game environment populated by three different kinds of NPCs powered by AI: (1) an NPC merchant with the ability to dynamically negotiate prices, (2) a task giver who can create contextually relevant tasks for the player, and (3) a NPC storyteller who can share lore-related stories based on a pre-made knowledge document. All NPCs communicate using predefined prompts and responses generated by a large language model. The implemented system consists of dialogue handling, user interface integration, prompt engineering, and external models communication modules. The results indicate a significant improvement in realism when using AI-powered dialogue compared to statically defined ones.***

Keywords: Artificial intelligence, Game development, Interactive storytelling, Large language models, Natural language processing, NPC dialogue, Procedural content generation, Prompt engineering, Real-time systems, Unity engine.

I. INTRODUCTION

Videogames use non-player characters a lot to tell stories and Non-player characters are heavily used in videogame development to enhance immersion by providing various gameplay features and narrative-related functions. NPCs offer quests, items, and services to players and develop the environment's atmosphere and lore. However, such interaction is usually implemented using pre-designed dialogue trees with fixed rules and pre-scripted sequences. While it is effective in keeping the players in line, it quickly becomes repetitive due to its predictability. The players get used to it after some time, and the immersion effect diminishes. In my opinion, large language models have a chance to fix it. They provide the possibility to interact with NPCs dynamically depending on the player's actions and responses. Although I am still skeptical about how good it can be, it looks promising. In this paper, authors attempt to integrate large language models to provide dynamic responses within the following three NPC archetypes common for RPGs and adventure games. The first one is an economist NPC who takes part in bargaining process, the second one is an on-the-spot quest giver, and the last one is a storyteller. They each perform a different task. The economist is in charge of economic negotiations, the quest generator provides on-the-go quest procedures, and the storyteller generates lore-based narrative from the existing lore database. Thus, this approach is a possible substitute for existing scripts, making them even more versatile, or it may be applied alongside them. Its main advantage is the potential of significantly improving immersion of players due to increased believability of NPC interactions and consistency of the game environment without the need for excessive amount of manually designed content. The authors seem to successfully utilize this idea by using prompts that contain proper formatting and are grounded in game context while maintaining interface integrity. The latter feature seems especially relevant since preserving consistency is an important factor in game design. This approach may be further used in the design of story-telling AI in games.

II. RELATED WORK

In the videogames industry, NPCs have typically relied on fairly simple behavioral strategies ranging from finite state machines to behavior trees or even pre-designed dialogue trees. Some attempts to create more dynamic behavior include such initiatives as Radiant AI in the Oblivion, which attempted to create some semblance of schedules for characters' actions based on rules and reaction to stimuli. Yet it still required all interactions to be predefined manually, thus making realism somewhat questionable. Another interesting example includes the Nemesis system used in Shadow of Mordor, where NPCs formed relationships which evolved dynamically and produced emergent narratives. While the Nemesis system indeed allowed for some form of dynamic interaction between characters, its implementation was ultimately limited by pre-defined scripts crafted by game designers. Those scripts ensured that player experience remained controllable and did not diverge beyond predetermined boundaries, to a certain extent.

There are a number of ways in which procedural narratives can be generated, among which the most prominent are event-driven story graphs, automatic quest generation, and planning-based story development. They enable creation of multiple alternative story paths but require substantial domain modeling effort and cannot process natural language. Therefore, procedurally generated narratives usually vary in structure but retain stylistic unity.

Recent advances in large language models (LLMs) have revolutionized approaches to designing game dialogue systems. Architectures such as GPT-series or LLaMA show impressive results in open-ended text generation, reasoning within given contexts, persona modeling, and few-shot learning capabilities. Previous researches employed such models in conversation systems, storywriting tools, educational tutors, and narrative simulation engines. However, the application of LLMs in real-time game environments creates additional constraints, such as necessity to guarantee prompt consistency, reliable output, grounding in verified game information, and logical gameplay continuity.

Hybrid solutions combining traditional game AI with natural language generation using LLMs have become increasingly popular recently. Such game AI relies on conventional rules implemented in game for basic functionality but uses large language models (LLMs) to generate engaging conversations and narratives. Hybrid approach strives to preserve control over game design while offering dynamic and varied interactions to players. Following this approach, this study utilizes techniques of structuring prompts and grounded retrievals.

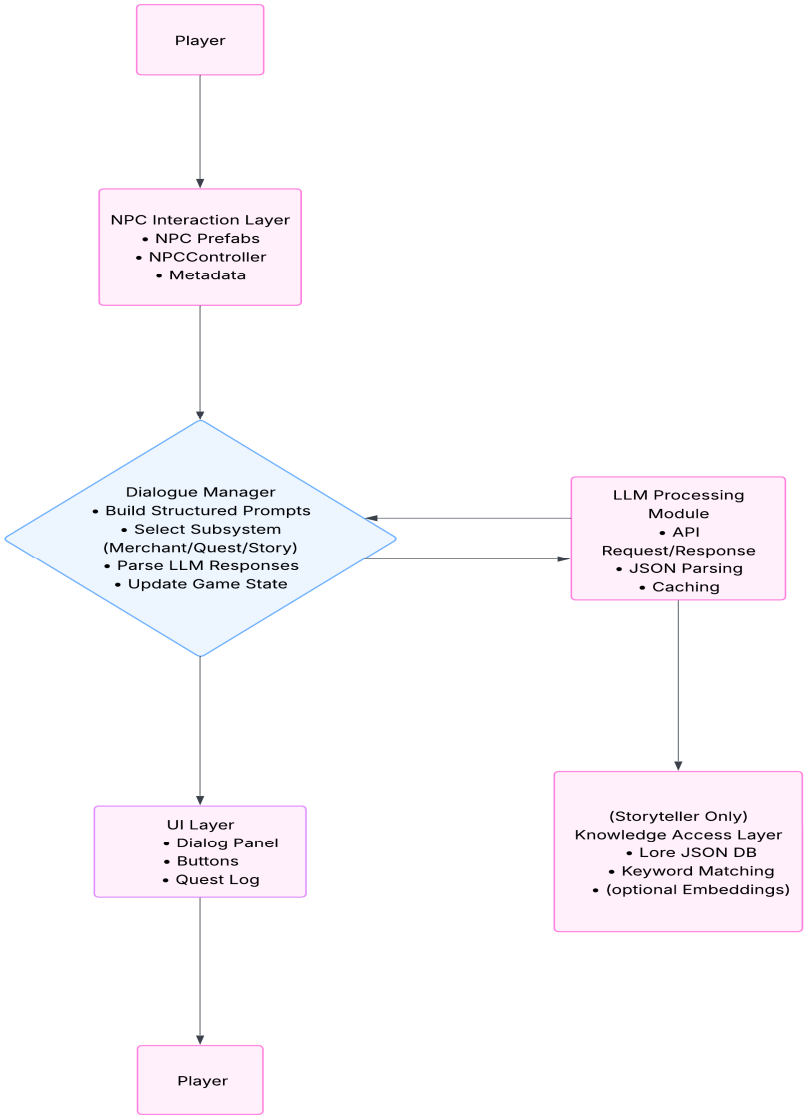


Fig. 1. System Architecture Overview

III. SYSTEM ARCHITECHTURE

The suggested architecture is modular in nature and enables dynamic interactions but still adheres to game mechanics that permit incorporating the LLM-based dialogue system into the gaming environment created using Unity. The five major elements of the architecture include the NPC Interaction Layer, Dialogue Management System, LLM Processing Module, Knowledge Access Layer, and User Interface Layer. When combined, these layers allow for interactions between the player and NPCs in real-time and contextualized settings.

A. NPC Interaction Layer

A Unity prefab incorporates personality profiles, interaction triggers, and character attributes to depict each NPC. The NPCController component identifies the type of dialogue processing required by designating one of three functional roles: storyteller, quest-giver, or merchant. NPC metadata—including personality traits, roles, base item prices, and associated keywords—is transmitted to the Dialogue Manager to guide the creation of prompts when a player initiates a conversation.

B. Dialogue Management System

All communication between players and NPCs is centrally managed by the dialogue manager. Upon receiving input from the user, it generates organized prompts based on the NPC's role and directs the request to the appropriate dialogue-handling procedure. For example, quest-givers expect outputs across multiple fields such as TITLE, DESC, OBJ, and REWARD, while interactions with merchants require organized pricing information. Additionally, this module analyzes LLM responses, adjusts game variables (including quest lists and dynamic pricing), and evaluates the effect of generated content on the game state.

C. *LLM Processing Module*

This module uses an HTTP client to connect to the external large language model. The system is capable of supporting live API calls for real-time inference of the model and mock responses, which are used for offline testing. The client sets up temperature and token parameters, structures requests using user and system prompts, and provides the output text for further processing. In the course of iterative testing, light caching is used to minimize the number of repeated calls. The design is flexible enough to accommodate permanent NPC memory or conversation history if needed, despite the fact that the prototype uses a stateless request architecture.

D. Knowledge Access Layer

For the purpose of grounded storytelling, the system has a retrieval module that maps player queries to the corresponding lore passages in JSON format. A simple keyword-matching algorithm is used to determine the most suitable lore piece to be fed to the LLM, thereby ensuring consistency of responses from the storyteller with the game world. This module can be expanded to embedding-based retrieval.

E. User Interface Layer

The UI Layer offers an interactive dialog box, quest log, and role buttons (bargain, help, story). The UI Layer accepts the parsed results from the LLM Dialogue Manager and displays them to the player. The UI components are also used to capture the player's text input and send events to the interaction pipeline. This layer ensures that the interaction is clear and consistent for all NPC roles.

F. Data Flow Summary

When the player chooses to interact, the following sequence occurs in the system:

1) The NPC Interaction Layer determines the NPC type.
2) The Dialogue Manager processes the input and builds a formatted query.
3) The LLM Processing Module produces a response.
4) The Dialogue Manager interprets the output and changes the game state accordingly (price changes, quest generation, story generation).
5) The User Interface Layer shows the output to the player.
6) The Knowledge Access Layer is called only for storyteller NPCs to fetch the appropriate knowledge.

This design is beneficial in that it provides a clear understanding of the responsibility of each component, is easily expandable to accommodate other types of NPCs, and works correctly even when using the probabilistic output of the LLM.

IV. METHODOLOGY

The approach to developing and testing the proposed AI-based videogame interaction system is broken down into three large components: implementation strategy, prompt engineering and model design, and evaluation procedure. This allows for the technical development of the system to be balanced with the testing of the effects of the LLM-based dialogue system on immersion.

A. Implementation Strategy

The prototype was developed using the Unity game engine with C# scripts handling player input, NPC actions, and data exchange between modules. Each type of NPC—merchant, quest giver, and storyteller—was modeled as a Unity prefab with metadata attributes like role, personality, base price values, and corresponding keywords. The master controller that sent user inquiries to the relevant functional pipeline was the Dialogue Manager.

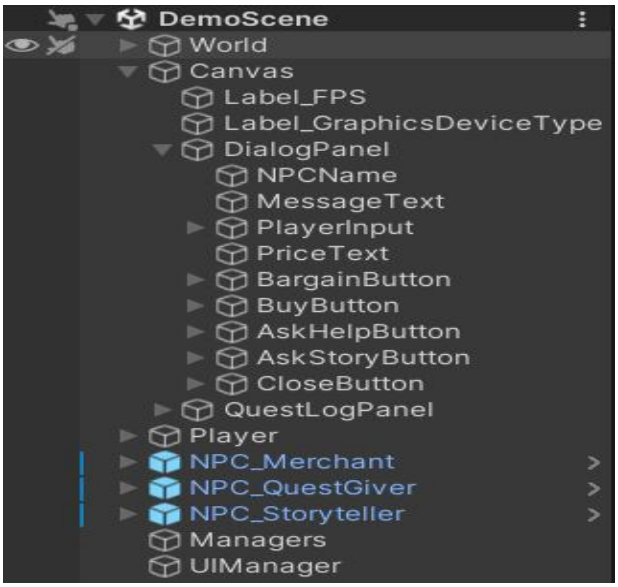


Fig. 2. Unity Hierarchy

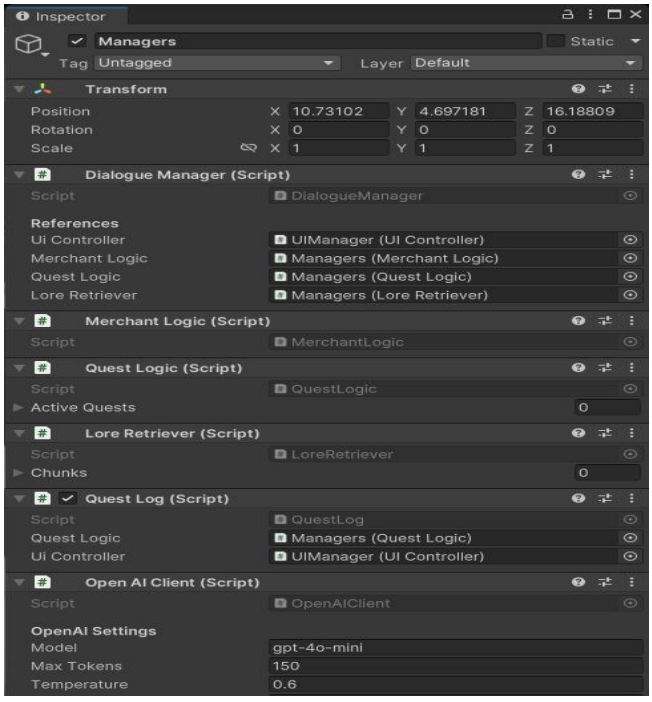


Fig. 3. Managers GameObject

The choice of two client types was done for flexibility purposes. In order to test the system logic without the need for API requests, a dummy client for a fake LLM was created. After the interaction flow was tested successfully, the client logic was moved to one that allowed submitting structured requests to an external large language model via HTTP requests. The responses were always delivered as JSON objects, which were processed deterministically and then applied to the game state according to the system design.

A caching algorithm was introduced to minimize the number of API calls. Additionally, due to the modularity of the system, it was possible to replace any part of the system quickly, e.g., the lore retrieval system or NPC logic without changing the entire architecture.

B. Prompt Engineering and Model Design

The prompt templates were designed depending on the requirements for different NPC types. The prompt template for the merchant involved the use of a limited prompt in order to produce structured output (e.g., a multiplier in front of the "PRICE:" label) to update the pricing of the items. For quest givers, a multi-field output prompt was used with TITLE, DESC, OBJ, and REWARD fields to facilitate easy conversion into game objects. The storytellers employed the idea of prompting with the lore chosen in advance, thereby grounding the story.

The temperature and token limit for the prompts were optimized depending on the required complexity. For prompts requiring a structured and reliable response, lower temperatures (0.2-0.4) were used. Higher temperatures (0.6-0.8) were applied for prompts for storytelling purposes. Keywords were used to map player queries to relevant lore chunks, but the design allows for using embeddings in the future.

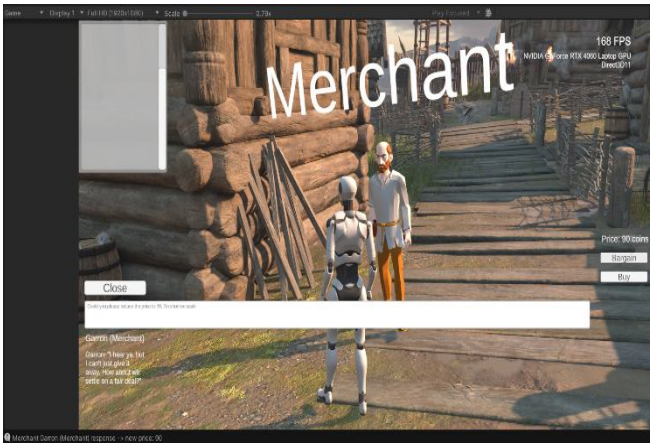
Throughout the development process, different safety measures were developed, including fallback rules for handling invalid prompts and deterministic overrides for stable gameplay.

C. Evaluation Procedure

To assess the performance of the designed system, it was necessary to check the effectiveness of the method for improving immersion and creating dynamic and contextualized dialogue. In other words, it is crucial to examine the responsiveness, coherence, and credibility of AI dialogue. Examples of bargaining conversations with the merchant, procedural quests with givers, and lore-related messages from storytellers were used for the evaluation. This qualitative assessment shows the differences in dialogue generated by AI versus script-based dialogue.

Feedback on the system was acquired informally from peer reviewers who tried out the prototype. Participants were supposed to provide feedback on clarity, creativity, and immersion compared to scripted NPC conversations. The results showed that AI dialogue improves the immersive experience because the behavior of NPCs changes in accordance with player actions.

V. RESULTS AND OBSERVATIONS



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

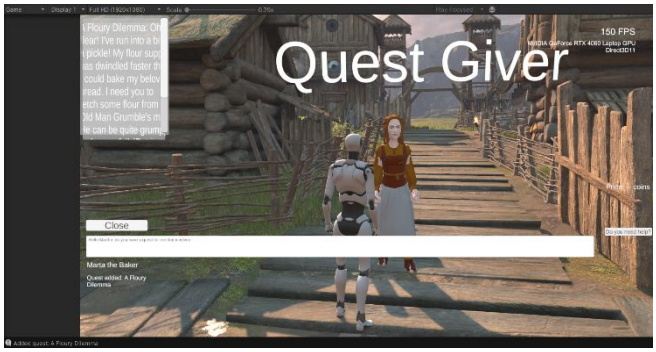


Fig. 4-5-6. Screenshots of NPC Interactions

Testing of the prototype has been done using both direct testing of the system and user feedback to evaluate the quality, consistency, and immersive nature of the gameplay interactions with the help of the language model. It is evident from the results that with the use of structured prompts, contextual data, and grounding methods, it becomes possible to create realistic conversations with different NPCs in various game scenarios.

A. Results from Merchant Interaction

It was possible for the merchant character NPC to effectively implement the price multiplier and conversation response depending on the player's negotiation attempt. The structured prompt assisted in ensuring that the model will consistently generate an appropriate numeric multiplier.

Sample Dialogue:

Customer: "I only have 60 gold pieces.

Can you sell me the health potion for something less?"

Vendor (LLM Generated):

"Looks like you're a good bargainer, traveller. Perhaps we

Can come to some sort of deal."

PRICE: 0.75

The multiplier lowered the price of 100 coins to 75 coins, showing the reliability of the connection between dialogue creation and the game's mechanics.

Observations:

The price changes happened in a deterministic manner because of the strict output formatting.

The users felt that the conversations were more "human" than the fixed merchant responses.

The users tried some unconventional bargaining methods, and the system handled them well because of the flexibility of the LLM.

B. Quest Generation Results

The NPC quest giver generated unique quests based on player queries, with the output following the structured fields as defined. This ensured smooth conversion of AI-generated content to quest objects in the game.

Sample Generated Quest:

TITLE: Missing Flour

DESC: Our shipment of flour never arrived this morning. The roads have been unsafe lately.

OBJ: Search the docks for a misplaced or stolen sack of flour.

REWARD: 50 coins

Observations:

Each produced quest conformed to the TITLE-DESC-OBJ-REWARD format.

Participants noted that the quests were personalized because the story adjusted according to the input wording.

The system prevented illogical goals because of the role-dependent constraints on the prompt.

C. Storyteller Narrative Results

The storyteller NPC created lore-stories based on earthed knowledge extracted from a local document. The LLM always converted lore statements into entertaining vocal stories without adding information that was not in the source text.

Sample Generated Story:

Player Query: "Tell me about the founding war."

Storyteller (LLM Output):

"Ah, the founding war… a tale etched into the very soil beneath our feet. When the river swelled and the northern raiders swept across the valley, it was Ardal the Stalwart who rallied the farmers behind the western wall. For three bitter winters they fought—not for glory, but for the hope of a peaceful town where children might grow without fear. And though the battles left scars on our land, they forged the unity that still binds our people today."

Observations:

The narrative aligned with the selected lore fragment.

The participants perceived the narrator as "alive" and "informed," leading to high levels of immersion.

The variation in vocabulary made the repeated inquiries seem fresh without interfering with the rhythm of the story.

D. System Performance and User Perception

For all the NPCs, the integration of the LLM yielded smooth, contextual conversations that outperformed conventional static models in terms of versatility and expressiveness.

Performance Observations:

There was nearly instantaneous response time when using the mock client, whereas actual API response time was between 0.3 and 2.0seconds. Using structured formats minimized parsing errors and facilitated a reliable implementation of the game. Caching aided in minimizing latency in the process of prompting.

Summary of User Feedback:

The participants pointed out: Greater immersion resulting from more authentic-sounding NPC dialogues. Greater engagement due to quests customized based on user inputs.

Depth created through intelligent storytelling by the AI.

Some participants have proposed implementing NPC memories and environmental factors, but this is listed as future work and is not required for this prototype.

VI. DISCUSSION

The prototype indicates that there is an enormous potential for utilizing large language models within the scope of videogame design in order to increase flexibility, expressiveness, and intelligence of NPC dialogue. By being modular, each NPC role – merchant, quest giver, and storyteller – could focus on showcasing one aspect of LLM-based dialogue generation that can be tremendously useful for a game. Structured prompts have played an essential role in generating output in a specific manner, allowing associating it with particular mechanics that required text input.

As mentioned above, one of the main strengths of the current system is its hybrid nature. In this regard, it should be emphasized that the combination of deterministic logic of the gameplay and probabilistic language generation makes sure that critical mechanics are still controlled by the designer, while the latter allows implementing expressive dialogue and even adaptation to the state of the world. As such, users indicated that they felt immersed into the world of the game and engaged by NPC dialogue, especially that of the storyteller, who told earthbound tales contributing to a sense of coherence in the narrative.

On the other hand, certain limitations were revealed throughout testing the prototype. First, using external LLMs means that there will always be some variability in response quality and possibility of inconsistencies in case templates are not constrained appropriately. In this respect, even though the above-mentioned problems were mitigated by using the templates, there remain certain deviations from the output format, meaning that additional validation mechanisms are needed. Second, the current prototype uses simple keyword matching when implementing the retrieval function, which works, but does not allow establishing semantic connections like those established by embedding-based retrieval algorithms. Third, real-time API usage introduces latency and costs.

Despite some limitations discussed above, the findings indicate that LLM-assisted NPC systems are quite a viable concept that can potentially be used to facilitate interactive storytelling and improve videogame experience through dynamic dialogues.

VII. CONCLUSION

Component	Function	AI Mechanism	Gameplay Impact
Merchant NPC	Price negotiation	Structured LLM output (PRICE: multiplier)	Dynamic item pricing and adaptive trade interaction
Quest-Giver NPC	Procedural quests	Multi-field LLM generation	Updates player tasks and shapes progression
Storyteller NPC	Lore narration	LLM generation grounded by retrieved lore chunk	Enhances narrative depth and world coherence

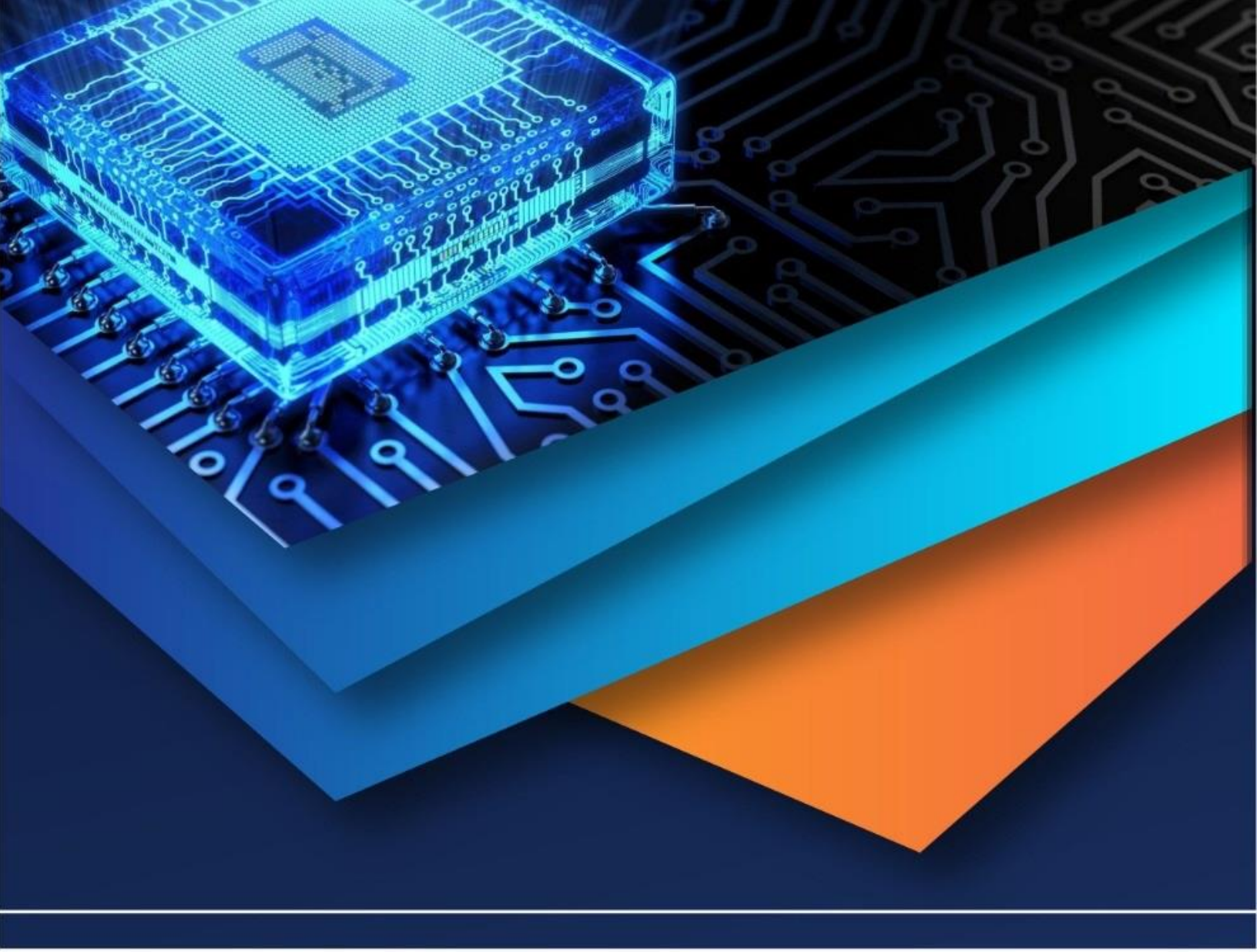
Table. 1. Overview of NPC Roles and AI-Driven Capabilities

This project provides a model system for integration of large language models into an interactive videogame environment in order to increase the NPC dialogue generation, procedural task generation, and storytelling based on lore. The current solution involves the use of structured prompts, contextual metadata, and modular Unity architecture that allows for dynamic interactions impossible for a traditional script-based approach for dialogue generation. The three implemented NPC roles in this project, such as merchant, quest giver, and storyteller, each involve a different application of LLM-aided generation.

It has been shown through analysis of the prototype that LLM-aided dialogue generation can contribute positively to the believability of NPCs and quality of narrative when constrained and rooted into the context. Despite the existence of some problems with consistency, computational complexity, and necessity to incorporate additional APIs, the results imply that the combination of deterministic game logic and probabilistic dialogue generation in hybrid models is a viable direction to further explore. The further research may involve development of more sophisticated retrieval techniques, memory for NPCs, and integration of AI systems into games.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
[2] T. Brown et al., “Language Models are Few-Shot Learners,” in Advances in Neural Information Processing Systems (NeurIPS), 2020.
[3] OpenAI, “GPT-4 Technical Report,” 2023. [Online]. Available: https://openai.com/research/gpt-4
[4] OpenAI, “Chat Completions API Documentation,” 2024. [Online]. Available: https://platform.openai.com/docs
[5] Meta AI, “LLaMA 3: Open and Efficient Foundation Language Models,” 2024. [Online].
[6] R. Koster, A Theory of Fun for Game Design, 2nd ed. Sebastopol, CA, USA: O’Reilly Media, 2013.
[7] J. McCoy, M. Treanor, B. Samuel, C. A. Mateas, and I. Horswill, “Prom Week: Social Physics as Gameplay,” in AIIDE, 2011.
[8] Bethesda Softworks, “Radiant AI System Overview,” 2006. Official Developer Documentation (Oblivion/Skyrim).
[9] C. Hecker, “Behavior Trees for Next-Gen Game AI,” Game Developers Conference (GDC), 2008.
[10] M. O’Hara, “The Nemesis System: Emergent Narrative in Shadow of Mordor,” GDC, 2015
[11] H. Ryan and M. Mateas, “Interactive Story Generation for Virtual Environments,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 3, pp. 173–186, 2009.
[12] N. Montfort, Procedural Generation in Games, MIT Press, 2020.
[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: A Method for Automatic Evaluation of Machine Translation,” in ACL, 2002.
[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in NAACL-HLT, 2019.
[15] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in Advances in Neural Information Processing Systems (NeurIPS), 2020.
[16] Unity Technologies, Unity Manual and Scripting API, 2024. [Online]. Available: https://docs.unity3d.com
[17] J. Togelius and J. Schmidhuber, “An Experiment in Automatic Game Design,” in IEEE Conference on Computational Intelligence and Games, 2008.
[18] S. Parekh, T. Smith, and W. Swartout, “Illusion of Life: A Formal Model for Narrative Variation in Interactive Storytelling,” in AIIDE, 2018.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY