An LLM-Based Behavior Agent with Natural Language Personality Control

Enabling Trait-Driven NPC Decision-Making through Prompt Engineering

Jos Timanta Tarigan
Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia
jostarigan@usu.ac.id (corresponding author)

Brian Wijaya
Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia
brianwijaya@students.usu.ac.id

Avin Chaili Salim
Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia
avinchailisalim@students.usu.ac.id

Sri Melvani Hardi
Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia
vani.hardi@usu.ac.id

Received: 8 June 2025 | Revised: 7 July 2025 and 20 July 2025 | Accepted: 23 July 2025

 DOI: https://doi.org/10.48084/etasr.12631

ABSTRACT

This study explores the use of Large Language Models (LLMs) for implementing personality-driven behavior in Non-Player Characters (NPCs) within games. A companion NPC leverages the OCEAN personality model to guide decision-making through natural language prompts, eliminating the need for traditional scripting or behavior trees. A stateless LLM combined with an automated prompt generator dynamically constructs context-aware prompts based on NPC traits, game states, and environmental factors. Implemented in the roguelike Rudantara RPG game, the companion NPC responds to gameplay conditions with behaviors aligned to its defined personality. The test results show that the system enables flexible and coherent decision-making and lowers the technical barrier to creating personalized behavior by allowing the player to interact using natural language instead of a complex behavior tree and scripting. Furthermore, to evaluate the decision-making process, participants with prior experience in RPG games were invited to play the prototype. Their responses indicated that the system was capable of simulating behavior aligned with the assigned personality traits.

Keywords-NPC agent; Large Language Model (LLM); behavioral agents; game development

I. INTRODUCTION

Traditional behavioral agents, such as NPCs, often use rule-based systems, FSMs, or behavior trees, which offer structured but predictable decision-making. These approaches lack adaptability, are difficult to scale, and offer limited support for personality-driven behavior customization. As user expectations for dynamic, intelligent gameplay grow, these methods fall short in delivering personalized or adaptive behavior based on personality traits. Integrating AI-driven solutions such as Large Language Models (LLMs) provides a promising alternative, enabling more adaptive, context-aware behavior through natural language understanding.

Various studies investigate prompt generators to integrate LLMs in game development. One of the most common uses of LLMs in games is to enhance the ability of game agents, such as NPCs, to interact with the player. These models, known for their ability to understand and generate human-like text, are particularly well-suited for applications that require dynamic and adaptive systems [1]. Various studies have discussed different aspects of game agents that can be integrated with LLMs, replacing traditional methods in game AI programming [2]. An example of integration of LLMs into games is to generate conversation on the run [3-5], which enhances the interactivity of conversational NPCs. Furthermore, the ability of LLMs to process and generate natural language, combined with their massive trained data, is a significant advantage in

developing dynamic, engaging, and rich content with traditional methods. Numerous studies have shown that LLMs are capable of generating high-quality dynamic in-game narratives, missions/quests, and objectives [6-9]. LLMs have also been proven to perform well in generating educational content in educational games [5, 10-12].

In addition to generating text-based content, several studies utilized LLMs as behavior agents to make decisions in games [13-16]. However, integrating LLMs as behavior agents can be challenging due to the unpredictable nature of LLMs. LLMs are designed to generate output based on probabilities derived from their training data, which can result in unexpected behavior. This inconsistency may not align with the context, narrative, or gameplay mechanics. To provide more coherent and contextually relevant responses, LLM integration requires a prompt generator that bridges the game's current state and the LLM by crafting prompts that encourage contextually relevant responses. The objective of a dynamic prompt generator is to translate the context, environment, NPC, and player states and actions, alongside a predefined behavioral guideline, into a text-based prompt to control unpredictability and reduce unwanted responses, allowing for more immersive and coherent experiences in interactive environments.

This study developed an automated prompt generator that leverages stateless LLMs to drive behavior agents based on personality traits through natural language input. Although supported by underlying code to execute actions, this system enables psychological traits to serve as the primary driver of NPC decision-making. To achieve this, the system relies on an LLM to interpret the psychological traits and translate them into context-appropriate actions. Rudantara, an action Role-Playing Game (RPG), was developed as the environment of the proposed agent. Specifically, the agents act as companions for the player. The modular system architecture combines memory, state, and environment modules to supply relevant context, allowing the LLM to generate decisions that align with the companion's defined personality. Through this setup, the aim is to demonstrate how natural language and psychological models can be combined to produce more expressive and customizable NPC behaviors.

II. PROPOSED METHOD

The proposed solution employs a stateless AI architecture, utilizing widely available free tools, such as the Gemini Pro API, to simulate the LLM-based NPC. The system relies heavily on the ability to design effective full context prompts to ensure continuity and relevancy in the companion's interactions. Thus, it is important to design an automated prompt generator that can trigger the correct response from the LLM using natural language. To ensure that the agent can generate structured and contextually appropriate prompts, an architecture was designed, composed of multiple interconnected modules, each having a distinct role. These modules work in tandem to process inputs and environments, generate prompts, and refine responses based on contextual cues, ensuring efficient and coherent interaction. As illustrated in Figure 1, the companion agent's architecture consists of five key modules, each contributing the necessary data for the prompt generator to create an accurate prompt.

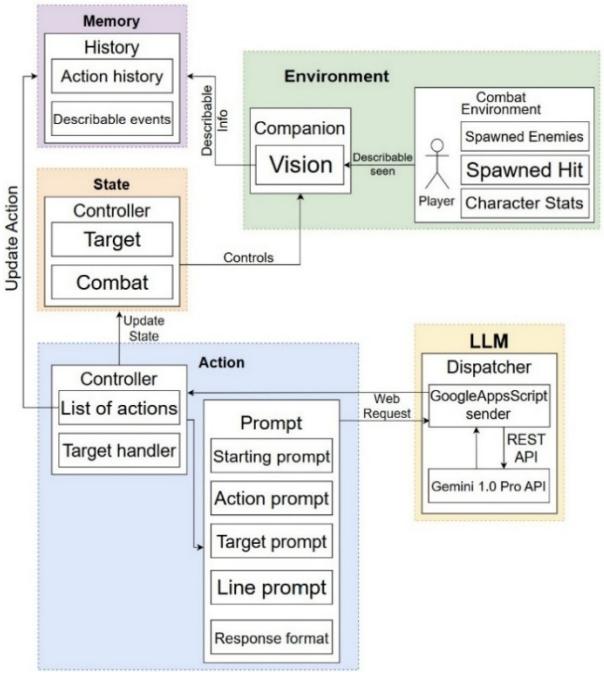

Fig. 1. System architecture of the LLM-based behavior agent.

A. *Large Language Model (LLM)*

The LLM module is the external part of the system, responsible for processing the prompt and returning a response. The proposed model uses Gemini Pro as the LLM solution. This service offers access to the LLM through the Google AI Gemini API using an HTTP Request. The free version comes with several constraints, such as maximum requests and tokens per minute, which were adequate for this study.

B. *Action Module*

The action module functions as an automated prompt generator, responsible for designing prompts using data from the memory and state modules, and communicating with the LLM via UnityWebRequest through a Google Apps Script that executes the Gemini API. In addition to generating prompts, it parses the JSON-formatted responses and forwards relevant information to the memory module (for long-term summaries) and the state module (for companion actions).

C. *Environment Module*

The environment module is responsible for gathering information about the player's surroundings. It collects various statistics such as enemies detected with their IDs, players, and statistics. The detection system uses vision with viewing distance, viewing height, and field of view set to 5 units, 6 units, and 150°, respectively.

D. *State Module*

This module regulates the companion using a finite state machine with two main states: combat (active engagement based on LLM input) and wavering (default following behavior). It also manages enemy detection through the Vision submodule, storing enemy data to be used by the action module during prompt generation.

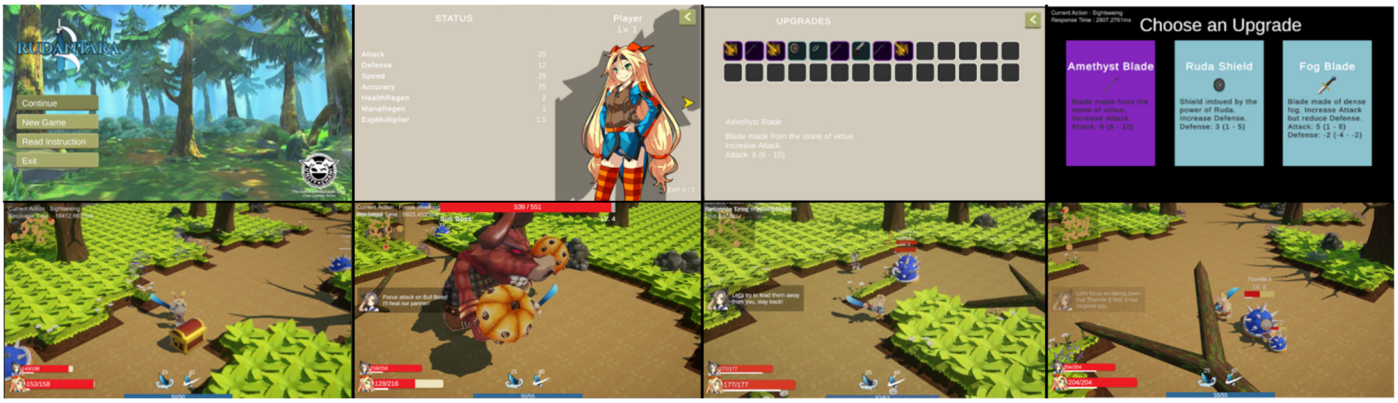

Fig. 2. Screenshots from Rudantara. Top: menu, character stats, items, and upgrades. Bottom: various gameplay events.

E. Memory Module

The memory module is responsible for returning the most relevant events that occurred previously. The NPC's memory is split into two parts: temporary and long-term. The short-term module logs each companion-player event and its frequency in temporary memory, and the LLM retrieves the three most recent events from this memory for prompt generation.

The long-term memory stores LLM-generated summaries of past events to guide consistent future actions in similar situations. This memory is part of the LLM response from a previous request in the form of a sentence. Additionally, each memory is attached to an attribute named recency to represent the age of the memory. The recency value is decaying, decreasing each time a new memory is added to the long-term memory module. The decaying formula of the memory is:

$$recency = recency * 0.9 \quad (1)$$

When the recency value is below the threshold of 0.25, the entry is wiped. This mechanism is necessary to limit memory usage since each entry is a set of sentences.

III. RUDANTARA

Rudantara is a 3D third-person hack-and-slash game prototype developed to test the proposed model. The game can be categorized as a rougelike RPG that contains rougelike components such as a procedurally generated level, a randomized loot system, and perma-death. In the gameplay, the player controls an anime-style character equipped with a sword as a melee weapon. The player's character can perform basic attacks and special attacks that deal more damage but require mana and cooldown. In its current version, the game features two types of enemies: Throntle and Bull Boss. The Throntle (short for thorned turtle) has two variants: the smaller and weaker one, and the bigger and stronger one. When the player reaches the other end of the level, he faces the Bull Boss. Defeating the Bull Boss ends the game, after which the player can start a new session with an increased difficulty level. Figure 2 presents various screenshots from Rudantara.

To help the player through the level, the game implements a friendly NPC (usually called companion) to aid the player. The companion will follow the player's character throughout the game. Whenever a specific event is happening, the companion will perform an action based on its traits. The companion has 2 states: combat and wandering. When not in combat, the companion has one action called "sightseeing", where he simply follows the player's character. During combat, the companion has seven types of predefined actions during the game. The actions are as follows.

- Focus on attacking one target.
- Attack the nearest seen enemy.
- Protect your partner by taunting enemies, tanking the damage.
- Protect your partner by luring enemies far from your partner.
- Seek protection from your partner.
- Flee from the battle to a safe place.
- Regroup with your partner.

IV. PROMPT DESIGN

The action of the companion relies on the LLM response triggered by the prompt. The challenge here is to design a prompt that is capable of generating the correct response by using a stateless LLM. The prompt must be able to provide crucial information to produce a relevant action. This section describes the overall architecture of the prompt. Due to the length of the prompt and its response, the full text from tests with various scenarios, along with the gameplay video, can be seen in [17]. The design template of the prompt contains the following.

A. Context

The context consists of the game description that includes the environment, types and descriptions of enemies, the goal, and the role of the NPC. This content part is uniform for all generated prompts.

B. Personality

The personality of the NPC is defined by five properties of the OCEAN model:

1. Openness: curious, adventurous, prefers changes, and likes to take risks.
2. Conscientiousness: organized and reliable.

3. Extraversion: Energetic and able to initiate/engage action.
4. Agreeableness: Friendly and cooperative.
5. Neuroticism: Anxious, volatile, and pessimistic.

The OCEAN personality model, commonly called the Big Five personality traits, is a commonly used method to add a human-like personality to virtual agents [18, 19]. Several studies have demonstrated that the OCEAN personality model serves as an effective framework for guiding LLMs in generating NPC dialogues and behaviors [19, 20]. This approach allows users to modify the NPC's behavior using a more humane way, and LLMs to generate responses that align with an NPC's predefined personality, ensuring that their interactions, while emergent, remain coherent within the NPC's characteristics. The player can modify these values by accessing the companion settings in the main menu and adjusting each trait from the value of 0 to 1 (Figure 3). In the prompt, these personality traits are uniform throughout the game session, unless changed in-game by the player from the option in the main menu.

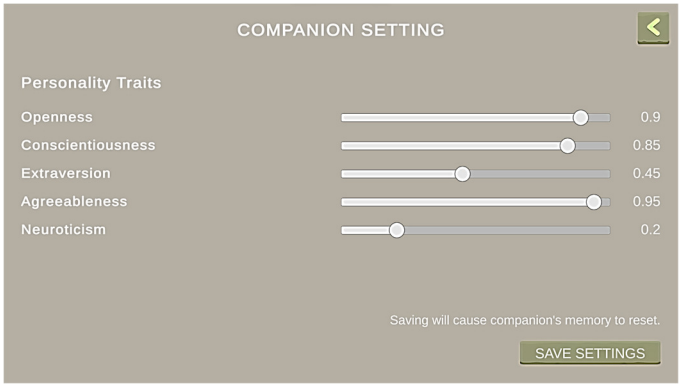

Fig. 3. The companion's settings, based on five personality traits.

C. Summary

The summary contains two parts: the companion's current state and its distance to the player, followed by the overall result from the previous action. The overall result is part of the LLM response that describes the environment, action, and intention of the companion. This information is necessary so that the LLM can generate an action relevant to the previous condition.

D. Recent Actions and Events

This part of the prompt contains information on the most recent action and its frequency. It acts as a short-term memory intended to give the LLM the current condition that affects the companion. This information is generated by the memory module.

E. Player and NPC's Current Status

This section of the prompt provides details about the player and their companion, outlining their current condition. It is crucial for describing their status, particularly the health information, which triggers healing and covering actions when needed.

F. Template of Expected Response from the LLM

The last part of the prompt describes how the LLM should respond and how it should form the response. It starts with informing the LLM that the response should be returned in JSON format using key and value. The next part of the template is to choose one of the seven actions to be performed by the companion. If the first action is selected (focus on attacking one target), the LLM should provide one specific target. Furthermore, since there is a possibility that the target is unreachable or no longer exists, the LLM must provide an alternative action that does not require a target, such as regrouping or seeking protection from the partner.

V. TESTING

The test was performed in two parts. The first one evaluates the validity and consistency of the prompt and the effect of personality traits on the response generated by the LMM on actions. The second part measures the performance of the proposed system based on the feedback of the respondents.

A. Validity Testing

In the first test, different personality trait settings were examined to generate unique and varied companion action outputs. To create personality combinations, common companion behaviors found in games were referenced. This research identifies three types of companion personalities commonly seen in commercial games: Protector, who prioritizes shielding their partner from danger; Leader, who actively engages in battle while supporting the player; and Rogue, who is highly unpredictable and often indifferent to the player's well-being. A Wildcard personality was also added, which was intended to act randomly without any pattern at all. Although this personality is rare in commercial games, it is interesting to observe how the traits affect the action.

Since decision-making is done by the LLM, it is important to understand how the LLM maps the OCEAN to these personalities. To determine appropriate trait combinations, the LLM was prompted to generate suitable OCEAN values for each role. The following description and value combination are concluded based on the response from the LLM.

1. Protector (P): Prioritizing the safety of the player, he requires high discipline. Openness: Low, Conscientiousness: High, Extraversion: Moderate-to-High, Agreeableness: High, Neuroticism: Low.
2. Leader (L): Leading the player in engaging the battle, he is highly disciplined and reliable during battle. Openness: Moderate-to-High, Conscientiousness: High, Extraversion: High, Agreeableness: Moderate-to-High, Neuroticism: Low.
3. Rogue (R): Prioritizing personal gain rather than discipline, acts independently, is open to creativity, but not a reliable companion to work as a team. Openness: High, Conscientiousness: Moderate, Extraversion: Low-to-Moderate, Agreeableness: Low, Neuroticism: High.
4. Wildcard (W): He is chaotic and tends to make random, impulsive decisions. Its action is unpredictable and has no

pattern at all. Openness: Moderate to High, Conscientiousness: Low to Moderate, Extraversion: Any, Agreeableness: Low, Neuroticism: High.

In this test, the game was played using each personality type, and the actions and decisions generated by the LLM were collected in response. To focus on battle-related behavior, non-combat responses were filtered out (action 7). Table I shows the frequency of each action assigned to the companion type and the personality setting.

TABLE I. PERSONALITY AND ACTION MAPPING

Type	Personality Settings					Action (%)					
	O	C	E	A	N	1	2	3	4	5	6
P	0.5	0.9	0.5	1	0.4	19	31	42	0	0	8
L	0.9	0.5	1	0.7	0.2	42	2	19	35	2	0
R	0.9	0.3	0.8	0.2	0.9	58	37	0	3	0	2
W	0.8	0.3	0.5	0.1	0.9	25	20	12	21	12	10

Figure 4 shows companion dialogs with the types Protector, Leader, and Rogue. Similar to the previous text, the dialogues show the characteristics of the companion. The Protector advises the player to stay back into safety, the Leader leads and suggests an action to the player, and the Rogue tries to take care of the enemy by itself. Additionally, the Wildcard dialogue is a mix between these three personalities without any specific pattern.

Fig. 4. Two example dialogues of the companion with Protector (a), Leader (b), and Rogue (c) personalities.

B. Usability Testing

The second test aimed to evaluate the impact of the proposed system on the experience, engagement, and perceived intelligence of the NPC. User testing involved eight participants with good knowledge and experience in gaming. Based on short interviews, all invited participants had at least 5 hours of gaming sessions per week. Before the test, the participant was informed about the objective of the investigation and the expected results. The player was also informed regarding the OCEAN-based NPC's traits, since this method is uncommon in commercial games. During the test, respondents were guided to engage with various in-game scenarios to assess the agent's ability to deliver appropriate responses. After completing the game session, participants were asked to complete a questionnaire consisting of six Likert-scale questions, ranging from 1 (strongly disagree) to 5 (strongly agree), as follows:

1. The game offers a good experience in terms of engagement and interaction quality.
2. The game mechanic is challenging, yet fair, encouraging players to replay the game.
3. The companion is responsive and able to react to changes in real time.
4. The companion dialogues are compatible with situations that occur during the game.
5. The companion's actions are coherent with the current player's actions.
6. The proposed model could be a feature in commercial games in the near future.

TABLE II. QUESTIONNAIRE RESPONSES

Q	Assessment					Total	AVG	STDEV
	1	2	3	4	5			
Q1	0	1	1	5	1	30	3.75	1.95
Q2	0	2	0	4	2	30	3.75	1.67
Q3	0	1	0	4	3	33	4.12	1.82
Q4	0	0	3	2	3	32	4	1.52
Q5	0	0	2	4	2	32	4	1.67
Q6	0	0	2	1	5	35	4.37	2.07

Table II shows the results of the questionnaire. The participants responded positively to all aspects of the proposed system. The highest average score (4.37) was recorded for Q6, indicating a strong agreement that the model has the potential to be featured in commercial games. High scores were also observed in Q3 (M = 4.12) and Q4-5 (M = 4.00), suggesting that the companion NPC was perceived as responsive, with dialogue and actions well-aligned to gameplay context. Meanwhile, general aspects such as overall experience (Q1) and game mechanics (Q2) also received favorable responses (both M = 3.75), indicating that the game was engaging and fairly challenging. Standard deviation values ranged from 1.52 to 2.07, reflecting moderate variability among participants but no indication of outlier responses. Overall, the results support the conclusion that the system delivers a coherent and context-aware NPC experience and was positively received by the participants in terms of functionality, interaction quality, and future applicability.

C. Constraints and Limitations

Although the test results demonstrated that the proposed method can simulate decision-making for companion NPCs in RPG games, several limitations emerged. Delayed NPC responses was a key issue, mainly due to inconsistent LLM response times rather than network instability. Although minor delays could be masked by expanding the NPC's detection range to give it more reaction time, occasional significant lags negatively impacted responsiveness. To address this, fallback behaviors, such as following the player or attacking nearby enemies, were implemented. Another limitation involved the use of the free-tier Gemini Pro, which imposed strict time-based quotas and placed requests in a non-priority queue. These restrictions caused delays and limited the number of simultaneous users. Although mitigated by allowing only one user at a time during the tests, this constraint could be resolved by switching to a higher-tier LLM service or deploying a dedicated open-source LLM server.

VI. CONCLUSIONS AND FUTURE WORKS

This study explored the use of LLMs to enable personality-driven behavior in NPCs through natural language input. By integrating the OCEAN personality model with an automated prompt generator, this study demonstrated how psychological traits can guide stateless LLMs in making context-aware decisions without relying on traditional scripted logic. Implemented in the Rudantara RPG prototype, the system allows NPC behavior to be flexibly defined using trait values and dynamically translated into in-game actions. The modular architecture, consisting of memory, state, and environment modules, provides the necessary context to the LLM, ensuring consistency and relevance in decision-making. The tests showed that the system produced believable and varied NPC actions aligned with their assigned personalities, offering a more intuitive and customizable approach to character behavior.

This method lowers the technical barrier for behavior design, allowing developers or players to shape NPC responses through high-level trait input rather than code. Although current limitations include the reliance on external LLM services and a fixed set of action templates, future work will focus on expanding behavioral diversity, reducing latency through local model deployment, and exploring additional psychological models for even greater flexibility in personality-driven game AI. Furthermore, while this research focuses on the RPG genre, the proposed system can be adapted to various other game genres with minimal modifications. Its modular design allows for easy integration into different gameplay mechanics and narrative structures, enhancing its applicability across diverse game types.

REFERENCES

[1] S. Ahriz, H. Gharbaoui, N. Benmoussa, A. Chahid, and K. Mansouri, "Enhancing Information Technology Governance in Universities: A Smart Chatbot System based on Information Technology Infrastructure Library," *Engineering, Technology & Applied Science Research*, vol. 14, no. 6, pp. 17876–17882, Dec. 2024, https://doi.org/10.48084/etasr.8878.

[2] S. Hu *et al.*, "A Survey on Large Language Model-Based Game Agents." arXiv, 2024, https://doi.org/10.48550/ARXIV.2404.02039.

[3] S. Abdelnabi, A. Gomaa, S. Sivaprasad, L. Schönherr, and M. Fritz, "LLM-Deliberation: Evaluating LLMs with Interactive Multi-Agent Negotiation Games," 2024, https://doi.org/10.60882/CISPA.25233028.V1.

[4] L. M. Csepregi, *The Effect of Context-aware LLM-based NPC Conversations on Player Engagement in Role-playing Video Games*. 2021.

[5] A. Isaza-Giraldo, P. Bala, P. F. Campos, and L. Pereira, "Prompt-Gaming: A Pilot Study on LLM-Evaluating Agent in a Meaningful Energy Game," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, Honolulu, HI, USA, May 2024, pp. 1–12, https://doi.org/10.1145/3613905.3650774.

[6] A. Marincioni *et al.*, "The Effect of LLM-Based NPC Emotional States on Player Emotions: An Analysis of Interactive Game Play," in *2024 IEEE Conference on Games (CoG)*, Milan, Italy, Aug. 2024, pp. 1–6, https://doi.org/10.1109/CoG60054.2024.10645631.

[7] X. Peng *et al.*, "Player-Driven Emergence in LLM-Driven Game Narrative," in *2024 IEEE Conference on Games (CoG)*, Milan, Italy, Aug. 2024, pp. 1–8, https://doi.org/10.1109/CoG60054.2024.10645607.

[8] S. Rao *et al.*, "Collaborative Quest Completion with LLM-driven Non-Player Characters in Minecraft," 2024, https://doi.org/10.48550/ARXIV.2407.03460.

[9] S. Rao *et al.*, "Collaborative Quest Completion with LLM-driven Non-Player Characters in Minecraft." arXiv, 2024, https://doi.org/10.48550/arXiv.2407.03460.

[10] S. Värtinen, P. Hämäläinen, and C. Guckelsberger, "Generating Role-Playing Game Quests With GPT Language Models," *IEEE Transactions on Games*, vol. 16, no. 1, pp. 127–139, Mar. 2024, https://doi.org/10.1109/TG.2022.3228480.

[11] A. Goslen, Y. J. Kim, J. Rowe, and J. Lester, "LLM-Based Student Plan Generation for Adaptive Scaffolding in Game-Based Learning Environments," *International Journal of Artificial Intelligence in Education*, vol. 35, no. 2, pp. 533–558, Jun. 2025, https://doi.org/10.1007/s40593-024-00421-1.

[12] K. Plupattanakit *et al.*, "LLMs in Eduverse: LLM-Integrated English Educational Game in Metaverse," in *2024 IEEE 13th Global Conference on Consumer Electronics (GCCE)*, Kitakyushu, Japan, Oct. 2024, pp. 257–258, https://doi.org/10.1109/GCCE62371.2024.10760474.

[13] X. Feng *et al.*, "ChessGPT: Bridging Policy Learning and Language Modeling." arXiv, 2023, https://doi.org/10.48550/ARXIV.2306.09200.

[14] S. Hu, T. Huang, and L. Liu, "PokeLLMon: A Human-Parity Agent for Pokemon Battles with Large Language Models." arXiv, 2024, https://doi.org/10.48550/ARXIV.2402.01118.

[15] W. Ma *et al.*, "Large Language Models Play StarCraft II: Benchmarks and A Chain of Summarization Approach." arXiv, 2023, https://doi.org/10.48550/ARXIV.2312.11865.

[16] O. Topsakal and J. B. Harper, "Benchmarking Large Language Model (LLM) Performance for Game Playing via Tic-Tac-Toe," *Electronics*, vol. 13, no. 8, Apr. 2024, Art. no. 1532, https://doi.org/10.3390/electronics13081532.

[17] J. T. Tarigan, "Rudantara Supplementary Files," *Google Drive*. https://drive.google.com/drive/folders/1d57gxZDeydPgum47UbxDcy2jAJpvpP6R.

[18] G. Liapis, A. Vordou, and I. Vlahavas, "Machine Learning Methods for Emulating Personality Traits in a Gamified Environment," in *Proceedings of the 13th Hellenic Conference on Artificial Intelligence*, Piraeus, Greece, Sep. 2024, pp. 1–8, https://doi.org/10.1145/3688671.3688757.

[19] S. Noh and H. C. H. Chang, "LLMs with Personalities in Multi-issue Negotiation Games." arXiv, 2024, https://doi.org/10.48550/ARXIV.2405.05248.

[20] L. J. Klinkert, S. Buongiorno, and C. Clark, "Driving Generative Agents With Their Personality." arXiv, 2024, https://doi.org/10.48550/ARXIV.2402.14879.