Player Modeling and Large Language Models: An Interaction-Based Approach

Carlos H. R. Souza[1], **Luciana O. Berretta**[1,2], **Sérgio T. Carvalho**[1,2]

[1]Institute of Informatics – Federal University of Goiás (UFG)
[2]Advanced Knowledge Center for Immersive Technologies (AKCIT/UFG)

carlos_henrique_rorato@discente.ufg.br, {luciana.berretta,sergiocarvalho}@ufg.br

Abstract. Introduction: *Adaptive games increasingly demand methods to understand and personalize player experiences.* ***Objective:*** *This study explores the integration of Large Language Models (LLMs) into Player Modeling, leveraging in-game interaction data to analyze player states, predict reference values, and identify dynamic player profiles.* ***Methodology:*** *We propose a Player Model that harnesses the reasoning capabilities of LLMs to generate these insights, enabling adaptive gameplay and personalized player experiences. To demonstrate feasibility, we implemented a proof of concept within the Unity Engine and tested it in a custom-designed game. The evaluation employed G-Eval, an LLM-assisted framework, focusing on the correctness of outputs.* ***Results:*** *The results showed a high average correctness score of 4.044 out of 5.000 across 50 input-output pairs, with 75% of respondes scoring 4 or higher. These findings suggest that LLMs can support Player Modeling, offering a foundation for dynamic difficulty adjustment and personalized game design.*

Keywords *Player Modeling, Large Language Models, Artificial Intelligence.*

1. Introduction

The demand for adaptive games has grown steadily in recent years, reflecting players' expectations for more personalized and engaging gaming experiences [Seyderhelm and Blackmore 2021]. This demand highlights the relevance of understanding players and their behavior, a goal of Player Modeling [Farooq and Kim 2024, Yannakakis et al. 2013]. Player Modeling involves creating computational models that describe players' states, predict their future behaviors, and identify their preferences [Tagliaro 2022, Fernandes et al. 2025, Mortazavi et al. 2024]. These models enable games to adapt dynamically to players, improving challenge and enjoyment through mechanisms like Dynamic Difficulty Adjustment (DDA) [Tagliaro 2022, Farooq and Kim 2024].

Researchers have implemented Player Models using various techniques, ranging from simple descriptive models to sophisticated artificial intelligence systems [Farooq and Kim 2024, Fernandes et al. 2025, Mortazavi et al. 2024]. While advanced AI approaches have shown potential for precise player adaptation, they often require extensive data collection and processing [Farooq and Kim 2024, Tagliaro 2022, Mortazavi et al. 2024]. This requirement poses challenges in gaming contexts, where little is known about the player at the outset (cold-start problem) [Seyderhelm and Blackmore 2021]. This limitation underscores the need for approaches capable of extracting meaningful insights from limited datasets.

Large Language Models (LLMs), such as GPT, have demonstrated remarkable capabilities in processing and analyzing data across various domains [Hu et al. 2024, Gallotta et al. 2024, Radford et al. 2019]. Their ability to interpret complex patterns and generate meaningful inferences makes them promising tools for Player Modeling [Hu et al. 2024, Gallotta et al. 2024, Souza et al. 2024a]. By leveraging in-game interaction data, LLMs can analyze the player's current state and predict future attribute values or performance levels, aiding in real-time game adaptation. These models can use techniques such as prompt engineering to process player data effectively, make predictions, and tailor gameplay experiences [Liu et al. 2023a, Hu et al. 2024, Wei et al. 2022].

Despite the potential of LLMs in Player Modeling, their application in this field remains underexplored [Gallotta et al. 2024, Wang et al. 2024]. This gap in the literature motivates further investigation into how LLMs can enhance the understanding and personalization of player experiences. To our knowledge, no works discuss the application of LLMs on Player Modeling and its implications.

This study addresses this gap by exploring the application of Large Language Models in Player Modeling. The proposed model leverages prompt engineering techniques and in-game data to infer observed attributes' reference values and identify player profiles. These profiles are dynamic, evolving as the player interacts with the game, providing a basis for personalized adaptations [Cömert and Samur 2023, Herbert et al. 2014a]. As a proof of concept, we implemented this approach in the Unity Engine and conducted an evaluation experiment using the G-Eval framework [Liu et al. 2023b], which has been adopted in studies for LLM-assisted evaluations and aims to validate the proposed approach's feasibility and potential.

The remainder of this paper is structured as follows. Section 2 presents background information on Player Modeling, Large Language Models, and the G-Eval framework. Section 3 reviews related work. Section 4 introduces the proposed Player Model. Section 5 details the proof-of-concept implementation. Section 6 reports and discusses the results. Finally, Section 7 concludes the paper with reflections and directions for future research.

2. Background

2.1. Player Modeling

Player Modeling refers to an abstract description of the current state of a player at a moment of the game [Farooq and Kim 2024, Yannakakis et al. 2013]. Tagliaro [Tagliaro 2022] focuses on Player Models as data collections utilized for metric adjustments on digital games. Player modeling addresses dynamic phenomena occurring during and as a result of gameplay interactions [Yannakakis et al. 2013, Fernandes et al. 2025]. Its applications include adapting game difficulty, generating personalized content, and enhancing the overall player experience by focusing on engagement, satisfaction, and enjoyment [Farooq and Kim 2024, Fernandes et al. 2025, Mortazavi et al. 2024, Tagliaro 2022, Souza et al. 2025, Souza et al. 2024b].

Player profiling is a concept related to player modelling. It involves categorizing and classifying players to better align with each player's expectations [Cömert and Samur 2023]. Additionally, player profile can evolve or even change over time, as well [Cömert and Samur 2023, Herbert et al. 2014a]. This recognition of the temporal aspect of player profiles highlights the importance of adapting game design strategies to accommodate players' evolving preferences and motivations throughout their gaming journey.

The data collected from the player can be *in-game data* (player actions and decisions), *objective data* (physiological signals and bodily movements), and *user data* (self-assessments and questionnaires) [Farooq and Kim 2024, Fernandes et al. 2025, Mortazavi et al. 2024, Tagliaro 2022]. Finally, data processing employs a variety of techniques [Farooq and Kim 2024, Fernandes et al. 2025, Mortazavi et al. 2024, Tagliaro 2022], ranging from heuristics and arithmetic operations to artificial intelligence (AI) and machine learning (ML) methods.

Despite the promise of current approaches, player modeling research still faces several challenges, particularly the lack of player data, especially at the beginning of a game, where little is known about the player (cold-start problem) [Farooq and Kim 2024, Fernandes et al. 2025, Mortazavi et al. 2024, Tagliaro 2022]. Most existing approaches require a large volume of multimodal gameplay and player data to build accurate and reliable models [Fernandes et al. 2025, Farooq and Kim 2024]. Other challenges noted in the literature include the generalization of player models, data interpretation, and the consideration of individual player characteristics (such as personality, motivations, and skills) for more precise modeling [Farooq and Kim 2024, Fernandes et al. 2025, Mortazavi et al. 2024, Tagliaro 2022].

2.2. Large Language Models

Large Language Models (LLMs) represent the current state of the art in artificial intelligence for natural language processing, capable of generating human-like text in response to input prompts [Gallotta et al. 2024]. These models are built upon transformer architectures, deep learning frameworks specifically designed to manage sequential data effectively [Radford et al. 2019]. A key feature of transformers is using multi-layer self-attention mechanisms, which allow the model to capture complex dependencies across elements in a sequence [Radford et al. 2019]. Prominent examples, such as OpenAI's GPT models[1], undergo extensive pre-training on vast and diverse text corpora, enabling them to internalize a wide range of linguistic patterns and semantic structures [Gallotta et al. 2024]. As a result, LLMs develop a nuanced understanding of syntax, semantics, and general world knowledge, enabling them to generate coherent and contextually relevant text [Hu et al. 2024, Gallotta et al. 2024, Radford et al. 2019].

One critical aspect of leveraging LLMs is prompt engineering. Prompt engineering involves crafting input prompts tailored to the specific task or context in which the model will be used [Liu et al. 2023a]. The design of these prompts significantly influences the output generated by the LLM, guiding its responses towards desired outcomes [Wei et al. 2022, Hu et al. 2024].

Recently, promising studies have emerged investigating possible roles that LLMs can play concerning digital games. Gallota *et al.* [Gallotta et al. 2024] point to more research on applying LLMs to adjust the game for a more engaging experience, showing that the path is open for research in this direction.

2.3. G-Eval

Evaluating content generated by natural language generation (NLG) systems is a complex challenge, mainly as Large Language Models can produce high-quality and diverse texts [Kalyan 2024, Zhao et al. 2024, Liu et al. 2023b]. However, most metrics (such as BLEU, ROUGE, and METEOR) rely on reference outputs, which are expensive and challenging to produce [Liu et al. 2023b].

Recent studies have proposed using LLMs as reference-free evaluators for NLG [Liu et al. 2023b]. Within this context, G-Eval emerges as a framework that utilizes LLMs

[1]https://openai.com/

with a chain of thought (CoT) approach to assess the quality of generated texts using a form-filling paradigm [Liu et al. 2023b]. G-Eval prompts the LLM with a task description and evaluation criteria, guiding it to generate a CoT that outlines the reasoning process step by step. The system then uses both the prompt and the CoT to evaluate the NLG outputs. It formats the evaluator's response into a structured form and can incorporate token-level classification probabilities to refine the final evaluation metric.

Experimental results demonstrate that G-EVAL-4 outperforms current NLG evaluators, showing stronger alignment with human judgments on benchmarks such as SummEval, QAGS, and Topical-Chat. It also provides more accurate continuous scoring by weighting discrete scores using token probabilities, enhancing reliability in summarization and dialogue evaluation. Nevertheless, LLM-based evaluators may exhibit bias toward outputs generated by the same underlying model [Liu et al. 2023b]. However, G-Eval has demonstrated strong potential for reference-free content evaluation, encouraging its adoption. Consequently, researchers have applied it in various studies involving similar evaluation tasks.

3. Related Work

Recently, promising studies have explored the potential roles of LLMs in digital games. Through a Systematic Literature Review, Gallota *et al.* [Gallotta et al. 2024] highlight the demand for further studies on leveraging LLMs to enhance game adjustments for a more immersive experience. Player modeling may be among these approaches aimed at making games more engaging.

Tagliaro [Tagliaro 2022] explores player modeling as an essential component for dynamic difficulty adjustment in games. These models consist of numerical attributes that describe play styles, such as accuracy or encounter duration, and are continuously adjusted to reflect changes in player behavior. The work of Charles *et al.* [Charles and Black 2004, Charles et al. 2005, Herbert et al. 2014b], in turn, conceptualizes player models and their use for content adaptation. It highlights the importance of creating dynamic player models that evolve through player-game interaction. Our work relates to these studies by similarly leveraging interaction data to model the player and adapting the game based on player preferences. However, our research investigates LLMs as components in analyzing the player data.

The study by Lima *et al.* [de Lima et al. 2020] combines player modeling and machine learning to create a computational model that identifies and predicts individual players' fears in a virtual reality horror game. By capturing gameplay data and player behavior, the model infers emotional responses to stimuli such as darkness, apparitions, and unfamiliar sounds. Using an Artificial Neural Network trained on labeled gameplay data, it predicts which horror elements are likely to evoke fear. Our approach aligns with this work, as both aim to personalize the gaming experience by capturing relevant player data to understand behavior. However, considering a continuous interaction with an LLM, our method reduces the need for pre-trained data.

4. The Proposed Player Model

The proposed Player Model leverages a Large Language Model to analyze data gathered directly from the player's interaction with the game, enabling a contextualized and adaptive interpretation of the information.

Initially, player attributes (in-game data) are captured in real-time through player interaction. Combined with a game description, this data is processed within the Player

Model and structured into a prompt fed into the LLM, which utilizes its extensive knowledge base to infer relevant information about the player. The interaction between the Player Model and the LLM reflects an adaptive process, ensuring continuous analysis tailored to the context of each player. In this way, the Player Model integrates player and game data to produce outputs supporting dynamic difficulty adjustment and performance analysis.

Player attributes serve as the foundational data points observed during gameplay. These attributes are measurable elements that represent specific aspects of the player's interaction with the game, such as reaction time, accuracy, resource management, or decision-making patterns. Each attribute is defined by:

- An *id*, which uniquely identifies the attribute.
- A *description*, providing a clear explanation of what the attribute measures.
- A *value*, representing the attribute's current state based on in-game events.
- A *threshold*, indicating the attribute's acceptable upper and lower bounds.

The game's objectives, mechanics, and intended player experience guide the selection of these attributes. For example, in a strategy game, designers might prioritize attributes like "time to make decisions" or "number of optimal moves per turn." In a first-person shooter, they may focus on accuracy and reaction time.

In addition to player-specific data, the model requires a comprehensive understanding of the game context. It achieves this by incorporating a game description, which can be manually curated or automatically generated using game assets, textual descriptions, or gameplay videos. Structured formats, such as the XML-Based Video Game Description Language (XVGDL) [Quiñones and Fernández-Leiva 2020], can provide an additional layer of precision, enabling the Player Model to interpret the game's mechanics, objectives, and dynamics more effectively.

The LLM is a key component of the Player Model, responsible for analyzing the captured data and providing the expected insights. The Player Model interacts with the LLM by constructing a prompt that includes the description of the game, recent player attributes logs, and contextual examples. This prompt was designed to enable the LLM to perform two primary tasks (although we understand that it is possible to direct the LLM to other outputs):

- **Generate Reference Values:** These are target values for each Player Attribute, representing ideal states that guide the evaluation of the player's performance. For instance, if an attribute measures "time to complete a level," the reference value might indicate the time expected for that player at that given difficulty level at that point in the game's progression.
- **Identify Player Profiles:** The LLM classifies the player into a predefined profile based on the player's current attribute values. These profiles may follow established frameworks, such as Bartle's taxonomy (Explorers, Socializers, Achievers, and Killers) [Bartle 1996] or be customized to the specific game.

The process aims to help tailor the insights the LLM provides to the game's and its players' specific needs. For example, a player's classification as an "Achiever" versus an "Explorer" can arouse adjustments, such as modifying resource availability or providing hints.

The prompt sent to the LLM is structured to provide all necessary context for accurate analysis. Table 1 outlines the typical format of this prompt. The prompt format, adapted from the SPAR framework[2] (Situation, Purpose, Action, and Result), is tailored to guide the LLM effectively. The "Action" field contains a concise description of adaptation guidelines based on the Chain-of-Thought Prompting (CoT)[3] technique, aiming to direct the LLM's reasoning explicitly [Hu et al. 2024, Wei et al. 2022]. The "Examples" section utilizes few-shot prompting to offer flexibility and efficiency in scenarios with limited data processing capabilities [Hu et al. 2024]. This approach balances the need for quick responses with the model's capacity to learn from a few examples. Finally, the Request field, an integral part of the prompt, contains the game variables' values used for input, which LLM uses to create the output, including the reference values and the current player profile.

Table 1. Structure of the prompt sent to the LLM.

```
# System: You are a Player Model mechanism for digital games. You receive current player attribute values and thresholds, along with the current player profile.
# Purpose: Generate the reference values for every attribute, indicating its ideal value at this point. Additionally, choose the profile that best matches the current player.
# Action: 1. Generate a JSON tuple with the reference (float) values of each player attribute, based on its current values, examples, the game context, and the objectives. Ensure all values are within their thresholds. 2. Compare the player's attribute values with the references and choose the player profile that best suits the current player, based on the comparison, examples and input data.
# Game Description: {game description in text or structured language}
# Data Log (most recent comes last): {Player Model's attribute reading history}
# Examples: {Pairs of input-output that can serve as examples for the LLM}
# Request: {Input data}
```

The ability of an LLM to generate meaningful outputs, such as a player profile and reference values for player attributes, is based on its capacity to process and reason about structured input data using patterns and relationships learned during training. It derives these outputs by establishing logical connections between player attributes, the game description, and the intended outputs. We advocate that the combination of player attributes and the game description provides a contextual foundation that guides the LLM's reasoning process, enabling it to produce coherent and goal-oriented outcomes.

The game description serves as the lens through which the LLM interprets the player attributes (understanding the types of player behaviors and tendencies relevant to the game context). Conversely, the player attributes are a real-time snapshot of the player's state. These attributes provide essential information about the player's progress and current situation within the game context. By examining these attributes in light of the game description, the LLM may infer how the player performs and how the game environment is likely shaping their behavior.

5. Proof of Concept

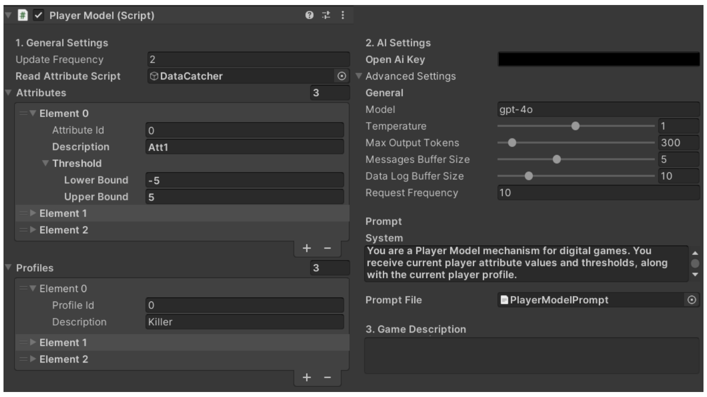
5.1. Developing the Player Model

The proposed Player Model proof of concept was implemented in the Unity Engine[4] using C#, leveraging the ChatGPT 4-o[5] Large Language Model to analyze player data dynamically. To enable communication between the Player Model and the LLM, we employed the open-source OpenAI library for Unity[6], which facilitates HTTP requests to the OpenAI API. The model comprises several core classes, each fulfilling a specific role in managing player data and interacting with the LLM.

The central class, `PlayerModel`, orchestrates the interaction between the game environment and the LLM. It maintains a list of instances of `PlayerAttribute`, representing

[2]https://beeazt.com/knowledge-base/prompt-frameworks/the-spar-framework/
[3]https://www.promptingguide.ai/pt/techniques/cot
[4]https://unity.com
[5]https://chatgpt.com/
[6]https://github.com/RageAgainstThePixel/com.openai.unity

dynamic, in-game data points such as "scores" or "remaining lives." The class also manages key configurations for LLM communication, including the OpenAI API key, model parameters (such as maximum tokens), and input/output prompts. A set of timing variables (e.g., `updateFrequency` and `requestFrequency`) regulates when the system updates attribute readings and sends requests to the LLM. The class implements methods like `GenerateValues()`, which asynchronously creates prompts and sends them to the LLM for processing. Outputs from the LLM, including player profiles and reference values, are processed by the `ProcessResponse()` method. The system also includes logging capabilities via the `PlayerLog` class, offering a structured record of player attributes and profiles over time.



Player Model (Script)

1. General Settings
Update Frequency: 2
Read Attribute Script: DataCatcher
Attributes: 3
Element 0
Attribute Id: 0
Description: Att1
Threshold
Lower Bound: -5
Upper Bound: 5
Element 1
Element 2

Profiles: 3
Element 0
Profile Id: 0
Description: Killer
Element 1
Element 2

2. AI Settings
Open Ai Key: [redacted]
Advanced Settings
General
Model: gpt-4o
Temperature: 1
Max Output Tokens: 300
Messages Buffer Size: 5
Data Log Buffer Size: 10
Request Frequency: 10

Prompt
System: You are a Player Model mechanism for digital games. You receive current player attribute values and thresholds, along with the current player profile.
Prompt File: PlayerModelPrompt

3. Game Description

Figure 1. Player Model's interface at Unity Inspector.

The `PlayerProfile` class encapsulates the concept of a player profile, a key output of the Player Model, which includes an identifier, description, and status flag. The `PlayerAttribute` class represents player-specific metrics, each defined by a value, a target reference, and an acceptable threshold. These attributes model real-time measurements relative to ideal states and form the core in-game data used to construct prompts for the LLM. To enable historical tracking, the `PlayerLog` class records past `PlayerAttribute` and `PlayerProfile` instances, providing context on player evolution that can inform future predictions and adaptations ("Data Log" section of the prompt on Table 1). The `ChatGPTClient` class abstracts interaction with the OpenAI API, managing model settings, token limits, and a buffered message list that simulates conversational memory; it includes methods such as `GPTComplete()` for asynchronous prompt submission and `ExtractJson()` for enforcing structured outputs. Together, these classes implement the core functionality of the Player Model in a modular way, supporting the integration of LLMs into the Unity game environment and enabling flexible experimentation.

The Player Model configuration is facilitated through an interface in Unity Engine's Inspector, as illustrated in Figure 1. The "General Settings" section allows developers to define the frequency, in seconds, at which the Player Model updates player data and interacts with other components. It also includes a field to link a script or component for gathering player attribute data. The "Attributes" section provides a structured way to manage the `PlayerAttribute` objects. Within the "AI Settings" section, tools are available to integrate OpenAI's language models.

5.2. The Testbed

We developed an authorial game to serve as an experiment tesbed, featuring a red capsule character that navigates the environment to collect yellow coins within a time limit. Once the player collects all available coins, the system instantiates new ones to ensure continuous gameplay. The game designer can adjust both the match duration and scoring parameters. In addition to collecting coins, the player must avoid a green spherical enemy; collision results in the loss of one of five lives, and the game ends when all lives are lost. This setup provides a testbed for evaluating the proposed Player Model's feasibility.

We implemented the Player Model based on the discussion presented in the previous section. Configured with a reading frequency of 2 seconds, the model monitored three core attributes: score, remaining lives, and time left. We considered the four Bartle profiles as possible player profiles and empirically set ChatGPT (GPT-4o model) with a temperature of 0.5. We also provided a brief description of the game to contextualize the LLM's analysis[7] We kept the remaining model settings at their default values (Figure 1).

5.3. Evaluation Methodology

We initially conducted game sessions using a simulated player that followed a straightforward behavior by moving randomly within the game environment, thereby generating random scenarios. During these sessions, the Player Model logged both input data (prompts) and output responses. Several subsequent simulations were conducted until the number of prompt-output pairs obtained reached 50.

After the simulated matches, the logged input-output pairs were subjected to a formal evaluation using G-Eval [Liu et al. 2023b]. The evaluation criteria focused on the *quality of response* [Kalyan 2024, Zhao et al. 2024, Liu et al. 2023b], specifically measuring the correctness of the Player Model's outputs. Table 2 shows the prompt used for the evaluation.

For each input-output pair, we ran G-Eval using three language models: Gemini 2.0 Flash[8], Meta LLaMA 3[9] and GPT-4o. To introduce variability in the LLM-generated outputs, we tested each of the 50 prompt-output pairs five times with each model, totaling 750 evaluations, 250 per model, as highlighted in prior research [Liu et al. 2023b].

Table 2. Structure of the prompt for G-Eval evaluation.

```
You will be given the output from a Player Model mechanism for digital games, that receive player data as input and give as output reference
values for player attributes and the profile that is being assumed by the player. Your task is to rate the Player Model output on one metric.
Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as
needed.
Evaluation Criteria:
Correctness (1-5) -- is the property of a system, process, or solution that ensures it behaves as intended and produces the desired outcomes
under the given conditions. It indicates compliance with predefined specifications, rules, or objectives, serving as a measure of accuracy
and reliability.
Evaluation Steps:
1. Ensure the output aligns with the expected structure and specifications.
2. Check if reference values and player profiles meet predefined criteria.
3. Ensure the player profile aligns logically with input data and attribute trends.
4. Assign a score for correctness on a scale of 1 to 5, where 1 is the lowest and 5 is the highest based on the Evaluation Criteria.
Data: {prompt-output that will be evaluated}
Evaluation Form (scores ONLY): -- Correctness:
```

The scores were then weighted by their associated probabilities to estimate a reliable score distribution for each input-output pair [Liu et al. 2023b]. Next, the score given to the Player Model by the judge model was calculated by averaging the scores given by this LLM for each prompt-output pair. The final score of the model was composed of the arithmetic mean of the overall scores provided by the three judges.

[7] *"It is a chasing game that features a character which moves through the environment, attempting to collect coins (10 points each) within a time limit of 3 minutes. When the player collects all the available coins, new ones are instantiated to keep the gameplay continuous. The goal is to get 500 points. In addition, the player must evade an enemy. If the player collides with the enemy, they lose one of their five lives. The game ends prematurely if all lives are lost."*

[8]https://gemini.google.com/

[9]https://www.llama.com/

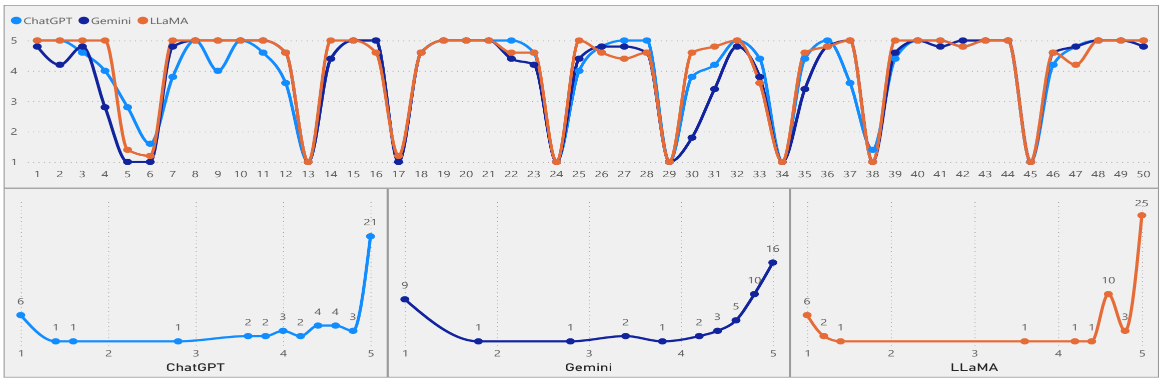


Figure 2. Grades: overall scores (top) and distribution per model (bottom).

6. Results and Discussion

The first key outcomes of this research were the successful implementation of the Player Model within the Unity Engine environment and its subsequent application in a functional game. This implementation points to the technical feasibility of the proposed Player Model, showing that it can be developed and effectively integrated into an existing game. These results highlight the Player Model's practicality, suggesting that it is theoretically viable and can also be applicable in a real gaming context.

The evaluation experiment results utilizing LLM-assisted judgment appear in Figure 2. The upper chart provides a high-level overview of the grades assigned by each model for all 50 evaluated prompt-output pairs. Meanwhile, the lower charts break down the distribution of grades, showing how frequently each score appeared. This visualization offers insights into the overall performance of the Player Model across different LLM judges and reveals the tendencies of each LLM when grading outputs. While most scores were positive, there were variations in the grading behavior between models.

The evaluation revealed a generally positive assessment of the correctness of the Player Model's outputs by the LLMs. All three models – ChatGPT, Gemini, and LLaMA – exhibited high levels of agreement on the adequacy of the Player Model's inferences, though some differences in grading tendencies were observed. Notably, the Gemini model tended to assign more conservative scores, generally lower than those designated by ChatGPT and LLaMA. Despite this conservatism, we advocate that this consistency may indicate the model's effectiveness in interpreting and analyzing player behavior.

Our analysis of the assigned grades pointed to the Player Model's potential for analyzing player data and making inferences. Of the 50 evaluated prompt-output pairs, a significant proportion received high grades. Specifically, ChatGPT assigned 21 scores of 5, Gemini assigned 16, and LLaMA assigned 25, collectively accounting for approximately 42% of all evaluations. Additionally, around 75% of all scores were four or higher, reflecting a strong consensus among the LLMs on the quality of the Player Model's outputs. These results suggest that the Player Model can generate outputs that align closely with the expectations of the LLMs. According to the G-Eval studies [Liu et al. 2023b], it may also reflect the general opinion of human judges.

The final scores of the evaluation provide evidence of the Player Model's performance: Gemini 2.0 Flash scored 3.936/5.000; Meta LLaMA 3 scored 4.148/5.000; and ChatGPT 4o scored 4.048/5.000. The overall average score of 4.044 indicates a high level of correctness in the Player Model's outputs. This result is promising and demonstrates that the proposed approach has the potential to leverage LLMs for the development of Player Models. It highlights that abstract and indirect data analysis through LLMs can yield meaningful insights into player behavior. As proof of concept, this evaluation underscores the feasibility of this approach and supports its potential for further refinement and application.

The analysis also revealed some challenges, mainly related to hallucinations in the LLM-generated outputs. Of the 50 evaluated outputs, 9 exhibited hallucinations, representing 18% of the total. Hallucinations accounted for 90% of the prompts that received low scores (2 or 1). The hallucination rate is within acceptable ranges for general use cases – literature often reports thresholds between 30% and 50% [Gunjal et al. 2024, Lonergan et al. 2023, Cheng et al. 2023]. The hallucinations observed in this study were primarily related to the output format (invalid JSON strings). However, they did not present significant issues, as we always validated the outputs before integrating them into the game.

In conclusion, the proof of concept results provide evidence of the proposed approach's feasibility. The findings suggest that LLMs can effectively support the development of Player Models for dynamic difficulty adjustment. The high average score points to this approach's potential. These promising results encourage continued research in this area, with future efforts to refine LLM configurations and expand datasets.

7. Final Remarks

This study proposed and explored the feasibility of a Player Model powered by Large Language Models (LLMs). The Player Model analyzes player data, generates inferences about the player's profile, and predicts reference values for gameplay adaptation. A proof-of-concept experiment pointed out that the Player Model can be implemented in the Unity Engine and integrated into a game. The results revealed positive feedback regarding the correctness of the outputs generated by the Player Model, with an average grade of 4.044 out of 5.000. Most responses received favorable evaluation from the judge models.

Nonetheless, certain limitations require attention. The G-Eval methodology, while effective, relies on the judgment of LLMs, which may introduce inherent biases [Liu et al. 2023b]. Evaluating complex prompt-output pairs that involve multiple variables and non-trivial logic is challenging, even for human evaluators [Kalyan 2024, Zhao et al. 2024, Liu et al. 2023b]. The Player Model's performance also depends on configuration settings, meaning different parameters can produce divergent outcomes. This emphasizes the need to investigate how configurations affect the process and results.

Future work should focus on further exploring the relationship between Player Modeling and LLMs. Specifically, experiments should include testing LLMs with optimized configurations to reduce hallucinations, analyze larger datasets, and expand the scope of evaluations to assess scalability and reliability. This research has the potential to uncover new ways for leveraging LLMs in Player Modeling and demonstrate their applicability and adaptability across different contexts.

Acknowledgments

This work has been fully/partially funded by the project "Research and Development of Algorithms for Construction of Digital Human Technological Components" supported by Advanced Knowledge Center in Immersive Technologies (AKCIT), with financial resources from the PPI IoT/Manufatura 4.0 / PPI HardwareBR of the MCTI grant number 057/2023, signed with EMBRAPII.

References

Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD Research*, 19(1).

Charles, D. and Black, M. (2004). Dynamic player modelling: A framework for player-centred digital games.

Charles, D., McNeill, M., McAlister, M., Black, M., Moore, A., Stringer, K., Kücklich, J., and Kerr, A. (2005). Player-centred game design: Player modelling and adaptive digital games.

Cheng, Q., Sun, T., Zhang, W., Wang, S., Liu, X., Zhang, M., He, J., Huang, M., Yin, Z., Chen, K., and Qiu, X. (2023). Evaluating hallucinations in chinese large language models.

Cömert, Z. and Samur, Y. (2023). A comprehensive player types model: player head. *Interactive Learning Environments*, 31(5):2930–2946.

de Lima, E. S., Silva, B. M. C., and Galam, G. T. (2020). Towards the design of adaptive virtual reality horror games: A model of players' fears using machine learning and player modeling. In *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 171–177, Recife, Brazil. IEEE.

Farooq, S. S. and Kim, K.-J. (2024). *Game Player Modeling*, chapter 1, pages 1–5. Springer International Publishing, Cham.

Fernandes, P. M., Lopes, M., and Prada, R. (2025). Generating game levels by defining player experiences. In *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE '24, Washington, DC, USA. AAAI Press.

Gallotta, R., Todd, G., Zammit, M., Earle, S., Liapis, A., Togelius, J., and Yannakakis, G. N. (2024). Large language models and games: A survey and roadmap.

Gunjal, A., Yin, J., and Bas, E. (2024). Detecting and preventing hallucinations in large vision language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18135–18143.

Herbert, B., Charles, D., Moore, A., and Charles, T. (2014a). An investigation of gamification typologies for enhancing learner motivation. In *2014 International Conference on Interactive Technologies and Games*, pages 71–78, Nottingham, United Kingdom. IEEE.

Herbert, B., Charles, D., Moore, A., and Charles, T. (2014b). An investigation of gamification typologies for enhancing learner motivation.

Hu, S., Huang, T., Ilhan, F., Tekin, S., Liu, G., Kompella, R., and Liu, L. (2024). A survey on large language model-based game agents.

Kalyan, K. S. (2024). A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, 6:100048.

Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T., Wang, K., and Liu, Y. (2023a). Jailbreaking chatgpt via prompt engineering: An empirical study.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. (2023b). G-eval: Nlg evaluation using gpt-4 with better human alignment.

Lonergan, R. M., Curry, J., Dhas, K., and Simmons, B. I. (2023). Stratified evaluation of gpt's question answering in surgery reveals artificial intelligence (ai) knowledge gaps. *Cureus*, 15(11):e48788.

Mortazavi, F., Moradi, H., and Vahabie, A.-H. (2024). Dynamic difficulty adjustment approaches in video games: a systematic literature review. *Multimedia Tools and Applications*, 83(35):83227–83274.

Quiñones, J. R. and Fernández-Leiva, A. J. (2020). Xml-based video game description language. *IEEE Access*, 8:4679–4692.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Seyderhelm, A. J. A. and Blackmore, K. (2021). Systematic review of dynamic difficulty adaption for serious games: The importance of diverse approaches. *SSRN Electronic Journal*, 1(1):1–45.

Souza, C., Oliveira, S., Berretta, L., and Carvalho, S. (2024a). Large language models and dynamic difficulty adjustment: An integration perspective. In *Anais Estendidos do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 31–36, Porto Alegre, RS, Brasil. SBC.

Souza, C. H. R., de Oliveira, S. S., Berretta, L. O., and Carvalho, S. T. (2025). Extending a mape-k loop-based framework for dynamic difficulty adjustment in single-player games. *Entertainment Computing*, 52:100842.

Souza, C. H. R., De Oliveira, S. S., Berretta, L. O., and de Carvalho, S. T. (2024b). DDA-MAPEKit: A framework for dynamic difficulty adjustment based on MAPE-K loop. In *Proceedings of the 22nd Brazilian Symposium on Games and Digital Entertainment*, SBGames'23, page 1–10, New York, NY, USA. Association for Computing Machinery.

Tagliaro, L. R. G. (2022). An implementation of adaptive difficulty systems for challenging video games. Undergraduate Thesis.

Wang, T., Honari-Jahromi, M., Katsarou, S., Mikheeva, O., Panagiotakopoulos, T., Asadi, S., and Smirnov, O. (2024). player2vec: A language modeling approach to understand player behavior in games.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models.

Yannakakis, G. N., Spronck, P., Loiacono, D., and André, E. (2013). *Player Modeling*, chapter 4, pages 45–59. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2024). A survey of large language models.