https://www.sworldjournal.com/index.php/swj/article/view/swj30-04-075
DOI: 10.30888/2663-5712.2025-30-04-075

INTEGRATION OF LARGE LANGUAGE MODELS (LLM) INTO THE GAME CONTENT CREATION PROCESS

Yuliia Yermolaieva
ORCID: https://orcid.org/0009-0004-3523-1930
Game Development, Environment 3D Artist, Peloton Interactive HQ
441 Ninth Avenue, 6th Floor New York, NY 10001

Abstract. *The article explores the role of large language models in video game development. The main objective is to analyze their applications and assess future development prospects. The study employs general scientific methods of cognition, including critical analysis of academic literature, examination of expert materials, grouping, sorting, systematization, and synthesis. The research findings highlight the key advantages of using large language models, such as significantly accelerating development and enabling automated generation of descriptions for various elements, from game environments to characters, objects, and behavioral scenarios. Language models are also widely used for dialogue generation, which is one of their most common applications, as well as for scenario modeling and adapting character behavior to the game context. The main drawbacks of these models include hallucination phenomena, which remain an unresolved issue. This necessitates clearer instructions from developers to reduce errors and improve task accuracy. Optimizing approaches to working with language models will allow for more effective fine-tuning based on specific requests, ultimately enhancing their quality and efficiency. In order to improve work with large conditional environments, game developers need to learn how to break down tasks into smaller components, provide more detailed instructions, and clearly understand the structure of queries and operations. These should be specialists in game environment development who use language models to accelerate their work rather than replace it. As of today, language models in automated mode are still unable to create fully effective game environments. The practical significance of the study lies in the potential to reduce game content creation costs through the use of large language models. This not only optimizes developers' work but also directly benefits end users by lowering the cost of gaming products.*

Keywords: *large language models, content generation, video games, development automation, data validation.*

Introduction

The emergence of language models has significantly transformed approaches to software development, scripting, and process modeling. Surveys of entrepreneurs indicate that most have already integrated large language model technologies into their production processes. These models are most commonly used for research and development purposes.

The origins of such models stem from fundamental research in artificial intelligence and machine learning. However, they are now actively employed for process automation, data analysis, and the generation of original texts. At this point, it is clear that large language models can handle complex tasks that previously required substantial labor resources.

The gaming industry also actively utilizes large language models at various stages of game development. Industry reports highlight that implementing artificial intelligence, particularly language models, can reduce labor costs by 40%, as the software development cycle is significantly shortened. This, in turn, lowers the overall cost of game production.

Modern games go beyond simple coding and graphics. They incorporate intricate storylines, dialogues, interactive elements, and require a deep understanding of context and programming languages. As they continue to evolve, large language models enable the creation of more realistic and immersive game worlds, significantly enhancing product quality and increasing the profitability of game developers.

An analysis of expert literature suggests that the potential for using language models to generate game content is promising and virtually limitless. These models are already accelerating game development, and in the near future, they will enable the creation of virtual environments that dynamically adapt based on user behavior.

Literature Review

The topic of integrating large language models is only beginning to gain popularity, which is why there are still relatively few scientific studies dedicated to this area. However, a considerable amount of specialized literature can be found online, including developer blogs and reviews of various software tools that help automate different aspects of game content development. The study by N. Alshahwan et al. at Meta [1] focuses on ensuring stability in game content development. Additionally, research by R. Gallotta, A. Liapis, and G. Yannakakis [2], as well as M. F. Maleki and R. Zhao [5], was used in the current study, as they explore general methodologies for creating AI-driven game environments.

A more general overview of how LLMs work, their capabilities, and their limitations in the context of video games is systematically discussed in reports by Just AI [3]. Furthermore, the potential applications of language models are analyzed in studies by P. Lanzi and D. Loiacono [4]. The collective findings of these studies confirm the transformative potential of integrating LLMs into modern game design and development. The application of LLMs in game code testing is examined in the work

of C. Păduraru, A. Staicu, and A. Ștefănescu [7], who propose methods for automated unit testing in game development. T. Reichert, M. Miftari, C. Herling, and N. Marsden [8] analyze the impact of LLMs on serious game development. A previous analytical review of the role of LLMs in game development was conducted by P. Sweetser [9].

Purpose of the article The objective of this article is to highlight the key directions for utilizing large language models in game development.

Research results

A language model is a program designed for natural language processing (NLP). It predicts the likelihood of word placement in a sentence or phrase and generates responses based on this, simulating meaningful speech.

The technology began to develop actively in 2017. The new Transformer architecture, developed by Google, became the foundation for future Large Language Models and fundamentally changed previous principles of machine language processing. With this technology, input data could be processed in parallel rather than sequentially, significantly increasing the speed of both model operation and training [8].

Large Language Models are characterized by their vast number of parameters, measured in billions. The number of parameters determines the neural network's ability to process data with maximum accuracy and speed—where speed is just as important as the reliability and coherence of the generated information. These programs operate based on machine learning algorithms, allowing them to process massive volumes of text data within seconds. Deep learning enables the system to understand the complexities of human language, even when queries contain technical terms, colloquialisms, or errors.

Today, there are many language models, categorized as either static or neural. Static models rely on traditional statistical methods and probability theory to determine the next words in a sequence.

Neural language models are considered more advanced, surpassing static models in efficiency by utilizing multiple types of neural networks to generate natural language. The most well-known among them include:

- GPT-4 by OpenAI. The latest version of the popular neural network, distinguished by enhanced "human-like" responses, reliability, and creativity. Its key innovation over the previous version is multimodality – GPT-4 can process queries not only in text format but also in audio and video.
- LaMDA by Google. A conversational neural model designed for user interaction and communication.
- BERT by Google. Primarily used for search queries, language translation, and question-answering tasks.
- BLOOM by BigScience. The largest multilingual neural network, trained on 176 billion parameters. It can generate text in 46 languages and 13 programming languages [3].

Table 1 systematizes the key capabilities of large language models in game development.

Table 1 – Capabilities of large language models (LLM) for game development

LLM Capabilities	Description
Content generation	Large language models can create textual and visual content, which is useful for developing scripts, dialogues, character descriptions, and world-building in games.
Code generation	LLM can act as "junior programmers," assisting in code review or independently writing code snippets that can be used for game mechanics development.
Answering queries and questions	Leveraging knowledge accumulated during training, LLM can quickly and accurately respond to player inquiries, providing real-time support or functioning as part of in-game NPCs.
Dialogue management for chatbots and NPCs	LLM can be used to control dialogues in chatbots or NPCs, ensuring realistic interactions with players through text or voice.
Information retrieval	Integrating LLM into in-game search systems can enhance the speed and efficiency of finding necessary information, such as solving quests or gathering resources.
Summarization	LLM can quickly generate summaries of texts or instructions, which can be used for creating hints or reference materials for players.
Machine translation	The ability to translate text between languages can be utilized for game localization, making the content accessible to a wider audience.
Code testing	LLM can check code and identify overlooked errors, reducing debugging time and improving development efficiency.
Sentiment analysis	LLM can analyze emotions in text, providing deeper insights into player reactions to game content or adapting NPC behavior based on player emotions.

Note: Systematized by the author based on [1, 3].

Large language models are primarily used in game development to generate character movements by applying simple instructions and methods, such as Low-Rank Adaptation. These techniques enable the creation of realistic animations, significantly reducing the workload for animators. As a result, character movements can be generated in real time, maintaining a natural appearance without requiring specialized graphic skills. Additionally, language models are utilized for animating non-playable characters. For example, they can simulate emotional states, modify the environment, or adjust the properties of certain objects. If a character experiences fatigue, the model can automatically alter their walking style, make their facial expression appear more exhausted, slightly slouch their posture, and slow their steps. The same applies to objects that change their properties under external influences [9].

Adapting character movements to the surrounding environment is particularly crucial in combat games, as language models can generate unique animations based on the selected weapon or fighting style. For instance, if a character wields a heavy weapon, their movements become slower, but their attacks gain more impact. Similarly, complex maneuvers such as jumps and dodges are modeled by considering environmental factors. High jumps and falls require different dynamics, necessitating distinct design approaches. Moreover, artificial intelligence enhances gameplay personalization by adapting characters to a player's style. If a player prefers an aggressive approach, the model can adjust movements, combat stances, and even the character’s appearance to match this playstyle [5].

Overall, the use of language models contributes to the naturalization of character behavior and appearance, ensuring their alignment with gameplay, settings, and environmental conditions.

Key aspects of adapting language models for character movement generation in video games should be examined. First, it is essential to focus on the accuracy of instruction design. Instructions consist of short tasks or task templates interpreted by artificial intelligence for execution. The more precisely these instructions are formulated, the higher the likelihood that the developer, organizer, or game designer will achieve the desired result.

Another important aspect is defining control conditions, particularly the initial and final positions of a character. Language models generate movement sequences in a specialized code format compatible with the game's programming language. This code facilitates the animation of characters, game worlds, objects, and other elements within the gaming environment.

An instruction consists of two key parts:

- Task description – specifying what needs to be done (e.g., creating a sword attack movement or running downhill).
- Control conditions – defining starting pose, keyframes, movement speed, etc.
- Example instruction: [Jump movement] [Starting pose: bent legs, arms raised] [Final pose: fully extended body in mid-air]. The LLM response would be encoded character movements that can be transformed into animation.
- Model training. LLM utilizes prior knowledge of human movements to generate realistic animations. If the model lacks prior training, it undergoes additional learning on a large dataset of human motion recordings. The LoRA method enables rapid training with minimal parameter modifications.
- Improving movement quality. When control conditions are provided, movements become more precise and realistic. Research indicates that motion generation with additional control parameters (e.g., text + pose) yields better results than using text descriptions alone.
- Unified training approach. Training the model on multiple movement types enhances its ability to interpret various scenarios and accurately reproduce new movement combinations. For example, if the model is trained on both walking and running, it can smoothly transition between these states when needed [10].

Despite the key advantages of using language models, it is important to highlight the main drawbacks and challenges faced by every designer and programmer who incorporates them into game development.

One of the challenges in using large language models is the phenomenon known as "hallucinations," which frequently occur in text-based models such as OpenAI's ChatGPT or Google's BART [4]. The core issue lies in the fact that generated code

often turns out to be fabricated, yet the model presents it as accurate and functional. In game development, for instance, there are cases where generated content does not align with the initial requests or simulates working processes, but upon publication and testing, it becomes evident that the results are non-functional [6]. Even the most advanced language models are capable of producing incorrect outputs, a fact acknowledged by AI system developers themselves. These challenges become particularly complex in fields requiring multi-step logical reasoning, as even a minor logical error can render an entire request unfeasible. To reduce the occurrence of hallucinations in language models, proper training of game environment developers is essential [2].

Effectively addressing hallucinations requires precise instruction formulation. Tasks should be concise and clearly defined. Large-scale tasks should be broken down into smaller sub-tasks that artificial intelligence can process sequentially in a coherent logical flow. For example, if a developer is tasked with creating a specific world, they should start with minor details, such as describing the appearance of trees, buildings, surface types, characters, weather conditions, or geographical features. By developing individual elements and subsequently integrating them into a unified whole, it becomes possible to achieve better coherence between components, thereby enhancing the efficiency of machine code generation.

Detailed task formulation is crucial, as it enables artificial intelligence to respond to a developer's requests with greater accuracy rather than generating its own versions of environments, objects, or characters. Another key aspect is the continuous learning process of the AI model. Through regular interaction, both the developer and the generative model adapt to the typology, sequence, and specifics of assigned tasks. This approach not only facilitates the training of the machine model but also helps developers refine their ability to formulate precise instructions, ultimately leading to significantly improved outcomes in the future [7].

It can be noted that the current practice of creating game content using large language models demonstrates both positive and less successful results. Among the less successful cases, there are instances where AI-generated characters, objects, or

environments fail to meet realistic expectations and significantly differ from those created without artificial intelligence. This issue is particularly evident in AI-generated character dialogues, which often lack realism, do not reflect contemporary speech styles, fail to incorporate relevant slang, or are entirely inappropriate for the given context.

More successful cases include game projects that undergo thorough verification of game code and environments. In such projects, developers pay close attention to every detail, ensuring that any unrealistic or incorrect outputs are promptly corrected and refined. Based on this, it can be concluded that the future prospects of language models are tied to their continuous improvement, particularly in reducing hallucinations and achieving better alignment with developer requests. It is expected that models will become more adept at understanding natural language without requiring simplified task formulations or the avoidance of complex terminology.

Looking ahead, language models are likely to be adapted to the specific needs of various industries. Specialized applications tailored for scriptwriting, dialogue generation, or the creation of specific game interfaces are expected to become more widespread. Since the initial setup of such applications will be conducted at a high professional level, they will enable lower-skilled specialists to interact with them effectively. This will accelerate game content development while reducing production costs.

Conclusions

Based on the conducted research, it can be noted that large language models already play a key role in video game development. They not only generate meaningful and in-depth visual content but also enable real-time analysis, adjustment, and verification, significantly simplifying the game creation process and making it more efficient. As a result, player interaction with the game environment improves, as characters become more adaptive to their surroundings.

The impact of language models on programming automation is substantial, as they reduce the overall workload on developers and enhance productivity. However, despite their great potential, language models still face unresolved issues, such as

hallucinations and information distortions, leading to the generation of inaccurate content. To address this problem, developers need to learn how to formulate clear tasks, providing detailed requirements for game code creation.

References

1. Alshahwan, N., et al. (2024). Automated unit test improvement using large language models at Meta. 32nd ACM Symposium on the Foundations of Software Engineering (FSE 24). DOI: https://doi.org/10.48550/arXiv.2402.09171.

2. Gallotta, R., Liapis, A., & Yannakakis, G. (2024). Consistent game content creation via function calling for large language models. 2024 IEEE Conference on Games (CoG), Milan, Italy, 1-4. DOI: https://doi.org/10.1109/CoG60054.2024.10645599.

3. Just AI. (2023). Large language models: What they are and how they work. URL: https://just-ai.com/blog/bolshie-yazykovye-modeli-chto-eto-takoe-i-kak-oni-rabotayut.

4. Lanzi, P. L., & Loiacono, D. (2023). ChatGPT and other large language models as evolutionary engines for online interactive collaborative game design. Proceedings of the Genetic and Evolutionary Computation Conference, 2023. DOI: https://doi.org/10.1145/3583131.3590351.

5. Maleki, M. F., & Zhao, R. (2024). Procedural content generation in games: A survey with insights on emerging LLM integration. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2024. DOI: https://doi.org/10.1609/aiide.v20i1.31877.

6. OpenAI. (2024). ChatGPT: Large Language Model. Accessed: 2024-07-03.

7. Păduraru, C., Staicu, A., & Ștefănescu, A. (2024). LLM-based methods for the creation of unit tests in game development. Procedia Computer Science, 246, 2459-2468. DOI: https://doi.org/10.1016/j.procs.2024.09.473.

8. Reichert, T., Miftari, M., Herling, C., & Marsden, N. (2024). Empowering female founders with AI and play: Integration of a large language model into a serious game with player-generated content. International Conference on Human-Computer

Interaction, 2024. DOI: https://doi.org/10.1007/978-3-031-60695-3_5.

9. Sweetser, P. (2024). Large language models and video games: A preliminary scoping review. Proceedings of the 6th ACM Conference on AI in Games. DOI: https://doi.org/10.1145/3640794.3665582.

10. Zhang, Y., Huang, D., Liu, B., Tang, S., Lu, Y., Chen, L., Bai, L., Chu, Q., Yu, N., & Ouyang, W. (2023). MotionGPT: Finetuned LLMs are general-purpose motion generators. Computer Vision and Pattern Recognition. DOI: https://doi.org/10.48550/arXiv.2306.10900.