Map Generation for Roguelike Games from Textual Descriptions Using Large Language Models

Gustavo Gurgel Medeiros[1], **Cristiano Bacelar de Oliveira**[1]

[1]Universidade Federal do Ceará (UFC) - Campus Quixadá

gusgurgel@alu.ufc.br, cristianobac@ufc.br

Abstract. Introduction: *Procedural Content Generation (PCG) enhances replayability in roguelike games but remains difficult to control via algorithmic parameters. Large Language Models (LLMs) introduce a "text-to-map" paradigm but still struggle with spatial reasoning.* ***Objective:*** *We propose an automated system that generates playable and structurally coherent roguelike maps from textual descriptions.* ***Methodology:*** *The architecture combines an LLM-driven Assets Generator for semantic content with a deterministic PCG algorithm for spatial layout generation and uses Retrieval-Augmented Generation (RAG) to map textual intent into visual assets, based on a dataset of 489 pixel-art images.* ***Results:*** *Evaluation indicates high semantic fidelity, with coherence scores of up to 93.6%, yielding visually cohesive and playable maps.*
Keywords *Procedural Content Generation; Large Language Models; Roguelike Games; Map Generation.*

1. Introduction

The roguelike genre in digital games has gained prominence through acclaimed titles such as Hollow Knight [Team Cherry 2017], UnderMine [Thorium Entertainment 2020], and Dead Cells [Motion Twin 2018]. Roguelikes are commonly characterized by procedurally generated dungeons, turn-based combat, and permanent death [Laurindo et al. 2024] in order to offer challenging and immersive experiences. A central component for such games is Procedural Content Generation (PCG), which automatically produces maps and scenarios to increase replayability [Sepänmaa 2024]. However, a longstanding challenge in PCG is controllability, since developers must often manipulate complex and abstract parameters to guide generation, making it difficult to produce content aligned with specific design intentions [Lazaridis e Fragulis 2024].

Recent advances in Large Language Models (LLMs) offer a promising alternative for addressing this controllability issue. Because LLMs can process natural language effectively, they provide a new interface for content creation: instead of tuning algorithmic parameters, users can describe the desired content through text. This "text-to-level" or "text-to-map" paradigm translates human intent into concrete game structures. Previous work, such as MarioGPT [Sudhakaran et al. 2023], has demonstrated the potential of generating levels from simple textual prompts. These results suggest that LLMs can produce structured and functionally valid game content, helping mitigate one of the central limitations of traditional PCG pipelines.

This work investigates LLM-driven generation in the roguelike domain, where strict structural constraints, such as room connectivity, corridor coherence, and logical placement of enemies and items, must be satisfied. The main objective is to develop a

system in which an LLM interprets textual descriptions and produces a structured map representation that can be rendered by a game engine. By combining prompt-engineering techniques with deterministic spatial generation, the proposed approach seeks to ensure the functional validity of generated maps while offering a more intuitive and expressive content-creation process than traditional algorithmic approaches.

From this perspective, the integration of LLMs can be interpreted as a natural evolution of data-driven PCG approaches, particularly for generating semantically rich structures and narrative elements. Retrieval-Augmented Generation (RAG) has emerged as a robust paradigm for improving the reliability of LLMs by grounding their outputs in external knowledge sources. Originally introduced by [Lewis et al. 2020] for knowledge-intensive natural-language tasks, RAG combines a model's parametric memory with non-parametric memory, typically implemented as a dense vector index.

This research makes both technical and practical contributions to the field of AI-assisted PCG and provides a bridge between abstract textual generation and concrete game artifacts. First, we propose a "text-to-map" architecture that separates semantic generation from structural topology, delegating spatial layout to deterministic algorithms while the LLM handles narrative elements. Second, this paper presents a RAG-based pipeline with Structured Outputs, supported by a manually curated dataset of 489 pairs of textual descriptions and pixel-art images, thereby enabling the alignment between natural language and visual game assets. Finally, it provides open-source tooling, including a Python-based Asset Generator, a Godot-based Map Generator, and the prompts used in the generator tests, all made publicly available to support future research[1].

2. Procedural Content Generation

Procedural Content Generation (PCG) encompasses a broad range of algorithmic approaches that have been widely used in level and narrative design. Its primary goal is to employ computational processes, either autonomously or with minimal designer intervention, to produce large quantities of varied content, often under limited development resources, or to create experiences that would be impossible or impractical to achieve through traditional manual design methods.

Among the well-known techniques are noise-based methods, such as Perlin noise [Perlin 1985] and Simplex noise [Perlin 2001], which are applied to generate organic spatial distributions, including terrains and textures. In parallel, formal-grammar approaches, including L-Systems [Lindenmayer 1968], Shape Grammars [Stiny e Gips 1972], and Graph Grammars [Ehrig et al. 1973], provide rule-based mechanisms for generating recursive geometric structures and complex topological layouts, such as roguelike dungeon maps.

PCG techniques are often complemented by agent-based simulations, such as the Random Walk algorithm [Pearson 1905][Koesnaedi e Istiono 2022], in which autonomous entities iteratively shape environments to produce intricate natural formations. Moreover, Search-Based PCG [Togelius et al. 2011] uses optimization techniques, like genetic algorithms, to explore large generative spaces and identify content that satisfies predefined design constraints. Earlier Machine Learning methods, including

[1] https://github.com/GusGurgel/sbgames-roguelike-map-generator

Artificial Neural Networks, Decision Trees, and Markov Chains, have likewise been used to extract patterns from existing data and generate sequential content.

3. Related Work

The use of LLMs in PCG has rapidly expanded in recent years. Several studies have investigated how these models can support the creation of coherent game worlds from high-level textual input. For example, MarioGPT [Sudhakaran et al. 2023] demonstrates the feasibility of text-to-level generation by using a fine-tuned model to create Super Mario Bros stages from simple descriptive prompts. This result shows that LLMs can produce functional game content. However, platformer levels are typically linear, whereas roguelike maps require non-linear graph structures with interconnected rooms and corridors, which introduces distinct layout challenges. Another relevant example is presented in [Auxtero 2023], where the authors propose the GEDC system, which uses AI-driven PCG to build interactive adventure game environments. Although GEDC shares the goal of making content creation more accessible, it places greater emphasis on manual customization during generation. In contrast, our approach prioritizes automated map generation under explicit roguelike design constraints.

In [Nasir et al. 2024], the authors introduce Word2World, a system that generates playable 2D worlds from short narratives. Their pipeline transforms narrative descriptions into concrete levels, but it does not account for the structural constraints inherent to roguelike maps like ours. Gallotta et al. [Gallotta et al. 2024] present LLMaker, a framework for dungeon crawlers that uses an LLM to improve consistency and reduce hallucinations. Despite this thematic proximity, LLMaker is primarily designed as an iterative, chat-based co-design tool. Our work, by contrast, targets a fully automated first-pass generation process driven by a single textual description.

These studies highlight the potential of text-to-level generation. However, they also reveal a research gap in the use of automated pipelines for producing structured data formats tailored to the architectural requirements of roguelike map generation.

4. Proposal Overview

As shown by [Yan et al. 2023], state-of-the-art models still struggle with tasks that require precise coordinate reasoning, such as plotting points in 2D/3D spaces and executing pathfinding algorithms. This leads to limitations of LLMs in geometric and spatial processing. These claims are further supported by [Gallotta et al. 2024], who argue that the lack of robust spatial reasoning hinders fully autonomous generation of complex, playable, and structurally coherent levels.

To overcome these constraints, this paper proposes a hybrid architecture that decouples creative intentions from structural building. In the proposed approach the generation process (please see Figure 1) is divided into two distinct but complementary modules: an LLM-driven Asset Generator, focused on semantics, and a PCG-driven Map Generator, focused on spatial topology. The LLM generates semantic assets (enemies, items, and tiles) using a texture Vector Store for RAG, while the PCG algorithm uses these assets to construct the spatial structure of the dungeon.

By integrating RAG into our text-to-map pipeline, the system performs semantic-similarity search over a curated database containing the visual textures. This strategy

reduces hallucinations and helps ensure that narrative concepts and entities generated by the LLM are mapped accurately and coherently to specific 2D pixel-art visual representations.

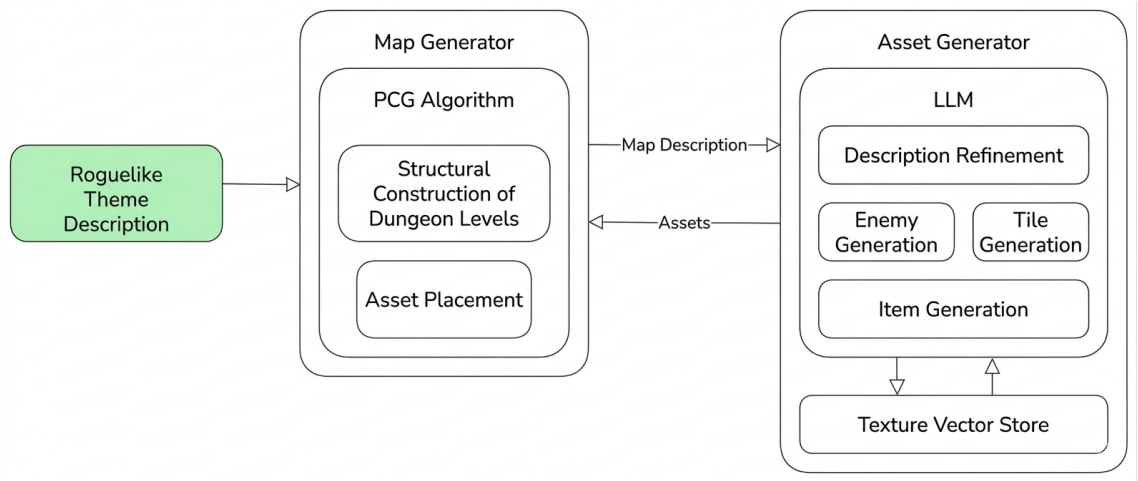


Figure 1. Proposal overview with Asset and Map generators

4.1. LLM-Driven Semantic Generation and Integration

The pipeline shown in Figure 2 presents the complete process which starts with a short text prompt provided by the user, defining the desired theme for the roguelike game. To increase semantic specificity, the system first refines this prompt by expanding it into a richer narrative context that defines the player background and the overarching dungeon lore. After this refinement step, the language model generates the entities that populate the map, including enemies and equippable items.

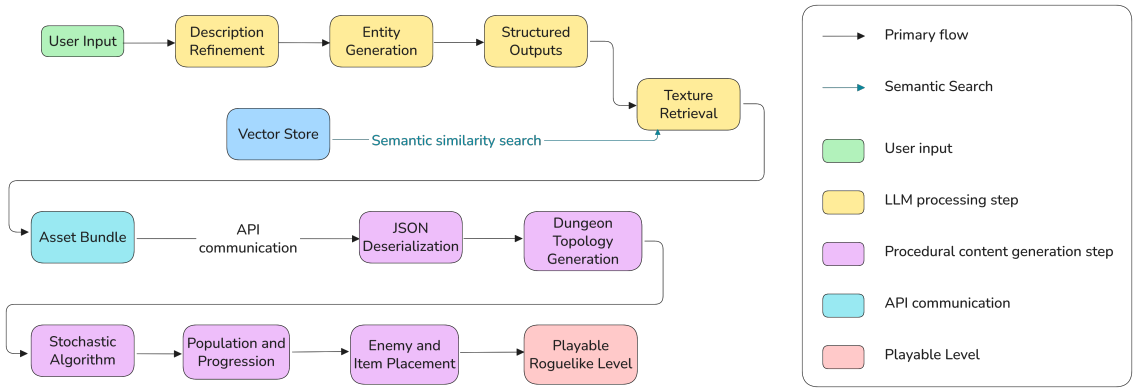


Figure 2. Complete flowchart of the hybrid map generation architecture

Because non-deterministic text output can hinder software integration, the system enforces structured outputs by requiring all generated entities to conform to strict JSON schemas, as suggested in [Shorten et al. 2024]. To decouple creative generation from numerical balancing, attributes such as rarity, threat, and weight are encoded on discrete scales from 0 to 10 and later converted by the game engine into final combat statistics.

In order to map semantic entities to concrete visual representations, the pipeline incorporates a RAG-based Texture Retrieval step (all visual game elements are considered

textures in this step). This component relies on a vector store of embeddings that associate textual descriptions with 2D pixel-art images A semantic similarity search using squared L2 distance then retrieves the most appropriate entity, based on its description, for each generated asset, following an approach validated by [Nasir et al. 2024]. The output of this phase is an Asset Bundle, represented as a comprehensive JSON object exposed through asynchronous REST endpoints built with FastAPI. Figure 3 shows an example. This modular design supports asynchronous communication and allows the game engine to retrieve and deserialize the data for dynamic node instantiation.

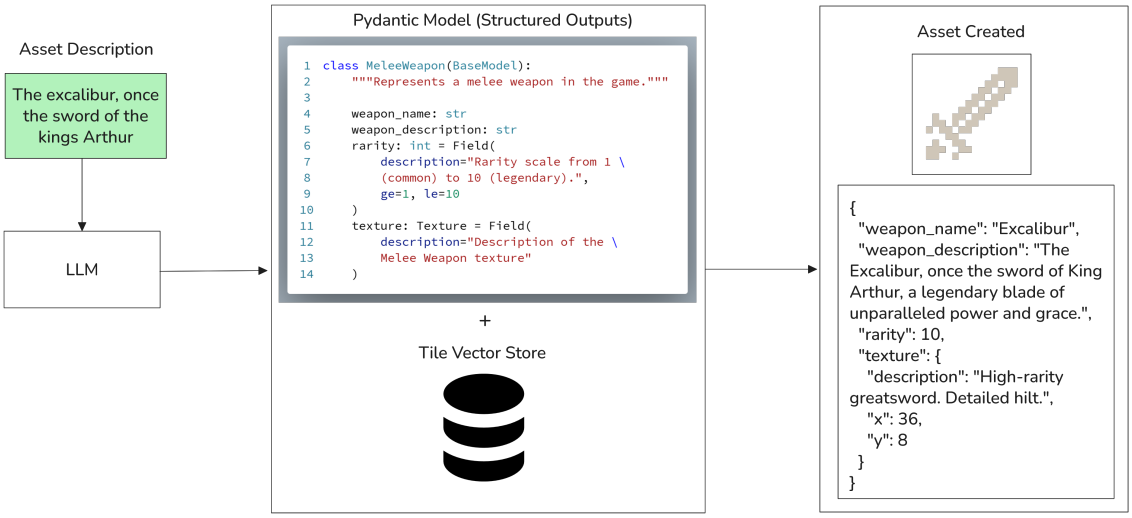


Figure 3. Example of assets retrieval based on textual description

For the Vector Store database, the system relied on the "1-Bit Pack" provided by Kenney under a Creative Commons CC0 license [Kenney 2026]. A rigorous manual curation process produced a dataset of 489 textual descriptions associated with base64-encoded image files, intentionally segmented into three macro-categories to reduce hallucinations during retrieval. Specifically, the dataset comprises 88 instances of *entities*, including enemies, NPCs, and the player avatar; 183 instances of *items*, such as potions, weapons, armor, keys, and treasures; and 218 instances of *environments*, covering structural and natural elements like walls, floors, obstacles, and vegetation.

4.2. Gameplay Loop with PCG

The next phase addresses the spatial limitations of LLMs by delegating dungeon construction to a stochastic algorithm based on random rectangle allocation with overlap checking. This approach produces structured, well-connected rooms that support tactical navigation in roguelike games. After the topology is generated, the algorithm populates the map by distributing deserialized entities according to a depth-based probability curve. Therefore, as the player progresses deeper into the procedural dungeon, the likelihood of encountering higher-threat enemies and obtaining higher-rarity items increases exponentially.

To validate structural integrity and thematic cohesion, we implemented a complete gameplay loop inspired by the classic game Rogue [Epyx Inc. 1984]. The player traverses procedurally generated depths to recover a rare final artifact generated by the LLM and

must then return safely to the surface. In accordance with core genre principles, the system enforces permanent death: failure resets structural progress, although thematic assets derived from the initial prompt remain consistent across attempts.

Finally, the Map Generator and gameplay logic were implemented in Godot Engine (version 4.6.1). As noted by [Salmela 2022], Godot offers strong integration between its native scripting language (GDScript) and its node system, making it suitable for parsing external JSON data into functional, modular architectures.

5. Experiments and Results

We defined two experimental scenarios that differ in scale, model configuration, and evaluation setup. These scenarios were designed to evaluate the robustness and scalability of the proposed system under varying parametrizations. Scenario 1 was evaluated using 5 prompts, while Scenario 2 used 15 prompts. All prompts were intentionally simple, with direct descriptions of typical roguelike game themes, ranging from medieval fantasy to alien bio-horror settings, as shown in Table 1.

Table 1. Prompts used in each experimental scenario.

#	Scenario 1	Scenario 2
P1	*Cursed Dwarven Forge* (Medieval/Fantasy)	*Neon Gothic Underwater* (Gothic/Underwater)
P2	*Derelict Starship* (Sci-Fi/Space)	*Overgrown Space Station* (Sci-Fi/Bio-Horror)
P3	*Smuggler's Grotto* (Pirate/Nautical)	*Clockwork Purgatory* (Steampunk/Supernatural)
P4	*Neon Skyline Penthouse* (Cyberpunk/Urban)	*Titan Bone Desert* (Dark Fantasy/Desert)
P5	*The Living Hive* (Alien/Bio-Horror)	*Frozen Citadel* (Dark Fantasy/Medieval)
P6	—	*Eldritch Flesh Labyrinth* (Cosmic Horror)
P7	—	*Toxic Cyberpunk Slum* (Cyberpunk/Dystopia)
P8	—	*Astral Floating Library* (High Fantasy)
P9	—	*Victorian Ghost Train* (Gothic/Terror)
P10	—	*Corrupted Gold Mines* (Fantasy/Underground)
P11	—	*Radioactive Amusement Park* (Post-Apoc.)
P12	—	*Crystalline Alien Mothership* (Sci-Fi/Alien)
P13	—	*Blood Moon Yokai Village* (Japanese Folklore)
P14	—	*Surreal Dreamscape* (Surrealism/Oneiric)
P15	—	*Dieselpunk Necromancy Trenches* (Dieselpunk)

In Scenario 1, the generation parameters were standardized to a larger scale: 6 depth levels, 20 enemy variations, and 30 weapon types. Three LLMs were evaluated as generators: **meta-llama/llama-4-maverick-17b** (a large-scale model with 17B active parameters), **openai/gpt-oss-120b** (a dense high-capacity reference model), and **openai/gpt-oss-20b** (a compact model for local hardware validation). For automated evaluation, **Google Gemini 3 Pro** was used as a blind judge, meaning it evaluated the outputs without knowing which underlying model generated the map assets. Semantic reconstruction was measured using the **text-embedding-004** model. Following prior work such as [Nasir et al. 2024], the temperature was set to 0.4.

In Scenario 2, the generation parameters were standardized to a smaller scale: 3 depth levels (initial, intermediate, and final), 10 enemy variations, and 10 weapon types. This ensures a fair comparison and accommodates the token limitations of the evaluator

LLM. Two LLMs were maintained as generators (**openai/gpt-oss-120b** and **openai/gpt-oss-20b**), while the third was replaced by **meta-llama/llama-4-scout-17b-16e-instruct**, which was also used as the blind judge. Semantic reconstruction and temperature settings were the same as in Scenario 1. Table 2 summarizes the two scenarios.

Table 2. Experimental configurations for scenarios 1 and 2.

Parameter	Scenario 1	Scenario 2
Number of Prompts	5	15
Dungeon Levels	6	3
Enemy Variations	20	10
Weapon Types	30	10
Generator LLMs	llama-4-maverick-17b-128e gpt-oss-120b gpt-oss-20b	llama-4-scout-17b-16e gpt-oss-120b gpt-oss-20b
Generator Temperature	0.4	0.4
Embedding Model	text-embedding-004	text-embedding-004
Evaluator LLM	gemini-3-pro-preview	llama-4-scout-17b-16e-instruct

5.1. Quantitative Analysis

Evaluating procedurally generated content along subjective dimensions such as narrative quality and aesthetics is inherently challenging, and traditional quantitative metrics often fail to capture these aspects. Therefore, we adopt an LLM-based evaluation strategy in which LLMs act as judges of generated content quality, following recent work such as [Nasir et al. 2024] and [Huang 2025].

The pipeline was evaluated using two primary metrics normalized to the range [0, 1], where higher values indicate better performance . The first metric is Coherence, inspired by [Nasir et al. 2024], which quantifies alignment between textual intent and the final game artifacts. An evaluator LLM receives both the refined textual description and the generated Asset Bundle and assigns a score based on visual similarity and thematic consistency across entities. The second metric is Semantic Reconstruction, which operates as a cyclic-consistency test similar to the approach in [Huang 2025] in order to evaluate fidelity in comparison to the prompt. Thus, it measures how well the generated assets represent the original theme. In this setting, generated asset data are provided to an LLM without access to the original prompt, and the model must infer the game's theme and setting solely from the assets. The reconstructed text is then compared to the original refined description using cosine similarity over vector embeddings. Higher similarity scores indicate that the generated visual and structural assets preserve and convey the user's initial thematic intent.

Table 3 shows the results for Coherence score in Scenario 1. In this regard, **gpt-oss-120b** model achieved the best overall average of 0.94. Notably, the compact **gpt-oss-20b** model reached a highly competitive average of 0.878. The proximity of the smaller model's results to the larger ones supports the hypothesis that compact models are viable for local procedural generation while maintaining an acceptable level of coherence.

In terms of Semantic Reconstruction (please see Table 4) in Scenario 1, all models demonstrated high fidelity, with averages near 0.89. The **gpt-oss-120b** model obtained the highest individual score on P2 of 0.921, indicating an excellent capacity to capture the nuances of the prompted Sci-Fi theme.

The results for Coherence and Semantic Reconstruction achieved in Scenario 2 are shown in tables 5 and 6, respectively. As in Scenario 1, the **gpt-oss-120b** model achieved the highest overall average of 0.893 for Coherence in Scenario 2. The **gpt-oss-20b** model followed closely (0.878), while **llama-4-scout-17b** achieved an average of 0.849. The Semantic Reconstruction metric shows that all models in Scenario 2 achieved strong fidelity, with averages ranging from 0.763 to 0.793. Interestingly, the **meta-llama/llama-4-scout-17b-16e-instruct** model achieved the highest overall average of 0.793. Additionally, **gpt-oss-20b** obtained the highest individual score in this scenario on P12 (0.855), demonstrating a strong capacity to capture complex thematic nuances, such as a crystalline alien mothership.

These results indicate that model scale has a limited impact on semantic fidelity when using structured generation pipelines.

Table 3. Results for Scenario 1 - Coherence Score

Model	P1	P2	P3	P4	P5	Avg
llama-17b	0.95	0.65	0.92	0.95	0.98	0.89
gpt-120b	0.82	0.95	0.98	0.95	0.98	**0.94**
gpt-20b	0.95	0.68	0.95	0.85	0.96	0.88

Table 4. Results for Scenario 1 - Semantic Reconstruction

Model	P1	P2	P3	P4	P5	Avg
llama-17b	0.880	0.883	0.890	0.882	0.905	0.888
gpt-120b	0.877	0.921	0.908	0.895	0.867	**0.894**
gpt-20b	0.882	0.913	0.865	0.888	0.895	0.888

Table 5. Results for Scenario 2 - Coherence Score

Model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	Avg
llama-4-scout-17b	0.82	0.88	0.88	0.82	0.90	0.85	0.82	0.85	0.88	0.85	0.82	0.85	0.85	0.82	0.85	0.849
gpt-120b	0.88	0.88	0.92	0.92	0.90	0.85	0.92	0.88	0.88	0.92	0.92	0.85	0.85	0.90	0.92	**0.893**
gpt-20b	0.88	0.90	0.90	0.85	0.92	0.90	0.88	0.85	0.88	0.85	0.85	0.85	0.90	0.88	0.88	0.878

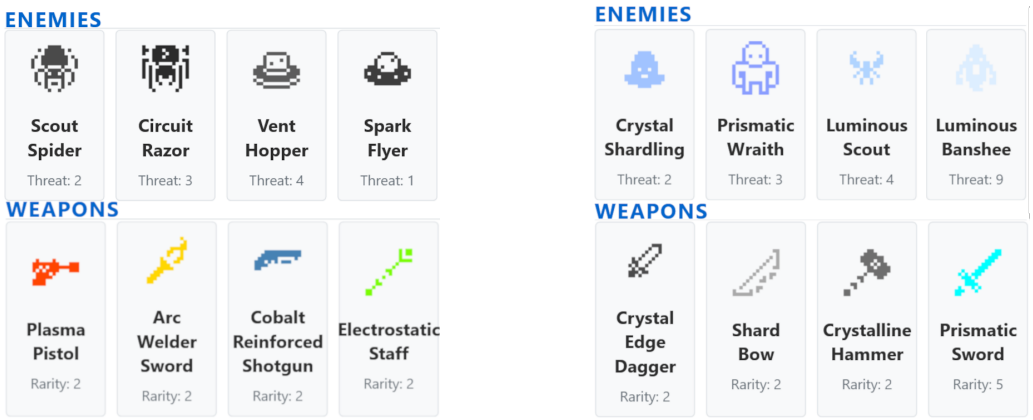
Table 6. Results for Scenario 2 - Semantic Reconstruction

Model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	Avg
llama-4-scout-17b	0.845	0.800	0.835	0.758	0.776	0.746	0.811	0.838	0.833	0.816	0.771	0.772	0.768	0.752	0.777	**0.793**
gpt-120b	0.764	0.780	0.720	0.788	0.732	0.772	0.763	0.757	0.822	0.742	0.740	0.797	0.682	0.770	0.824	0.763
gpt-20b	0.840	0.837	0.821	0.751	0.695	0.789	0.777	0.773	0.742	0.747	0.780	0.855	0.777	0.776	0.745	0.780

5.2. Qualitative Analysis

To better understand the subjective quality and visual cohesion of the generated content, we conducted a qualitative analysis across representative outputs from different scenarios and themes. Across the evaluated cases, the system consistently produced asset bundles that reflect the intended atmosphere described in the input. Environments characterized by industrial or technological settings, for instance, frequently incorporated elements such as metallic structures, exposed conduits, and signs of structural degradation, contributing to

a coherent visual identity. Similarly, fantasy-oriented prompts resulted in assets featuring organic materials, natural textures, and architectural motifs aligned with medieval or mythical contexts. Figure 4 presents a representative example of generated assets.



(a) Assets with Sci-Fi theme **(b) Assets with Medieval/Fantasy theme**

Figure 4. Examples of visual cohesion of the generated assets using RAG

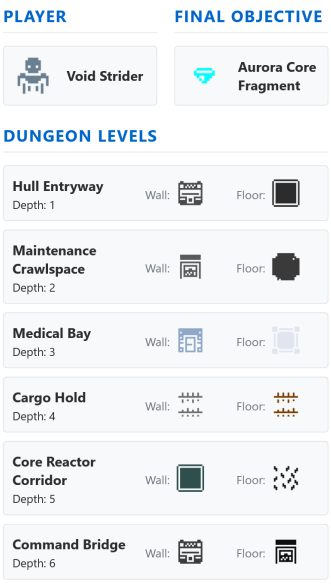
It is possible to notice that thematic cohesion extends consistently to enemies and weapons. Across different scenarios, adversaries are generated in alignment with the underlying setting, ranging from mechanical constructs in technological environments to organic or mythical creatures in fantasy contexts. In industrial and science-fiction settings, for example, enemies often take the form of drones, automatons, or hybrid machines, with visual characteristics such as metallic surfaces and color palettes dominated by gray, blue, and red tones, reinforcing their artificial nature.

Weapons similarly reflect the thematic constraints of each scenario. In technology-driven contexts, the system generates items that combine advanced functionality with improvised or industrial aesthetics, such as energy-based firearms or modified tools repurposed as weapons. In contrast, fantasy scenarios tend to produce weapons aligned with traditional archetypes, including blades, staffs, and enchanted artifacts. This alignment demonstrates that the system not only preserves high-level semantic intent but also propagates it consistently across multiple asset categories.

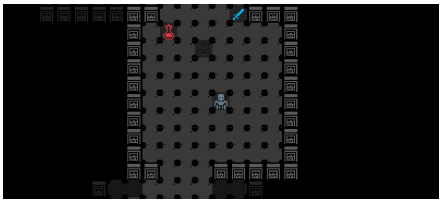
The player character is typically aligned with the narrative context, often represented using appropriate visual proxies retrieved by the RAG pipeline. Described player entities are mapped to corresponding textures, preserving semantic consistency between description and representation.

In terms of spatial organization, the generated layouts exhibit a logical progression of areas, transitioning from entry zones to intermediate regions and culminating in a final objective. Figure 5 shows an example of generated level maps. This progression is reinforced by the consistent use of thematic textures, ensuring that both structural topology and visual elements contribute to a unified gameplay experience.

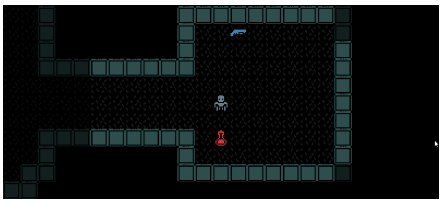
Overall, the coherence observed across environments, characters, enemies, and equipment suggests that the proposed pipeline effectively integrates semantic generation with retrieval mechanisms. The resulting asset bundles exhibit a unified visual and narrative identity, indicating that the system is capable of translating abstract textual



(a) Map/Dungeon Assets



(b) Map for Level 2



(c) Map for Level 3

Figure 5. Example of level maps and assets

descriptions into cohesive and contextually consistent game elements.

6. Conclusion

This paper presented a novel procedural content generation (PCG) approach for roguelike games leveraging Large Language Models. To address the controllability challenge in traditional PCG, we proposed a fully automated hybrid "text-to-map" architecture decoupling semantic generation from spatial structure. In this framework, the LLM handles high-level creative elements (narrative context, enemies, items) while deterministic algorithms ensure coherent and playable dungeon layouts.

Integrating Retrieval-Augmented Generation (RAG) and structured outputs effectively translated abstract textual descriptions into consistent visual assets, demonstrating strong thematic coherence and semantic fidelity. While larger models performed best, compact models remained competitive, highlighting the feasibility of local deployments without dependence on cloud resources. The system also demonstrated practical applicability through game engine integration via structured data interfaces, enabling an end-to-end generation pipeline for graph-based environments.

Despite promising results, limitations remain. Visuals are constrained by a fixed dataset, and the evaluation risks bias, as qualitative analysis was restricted to the authors and quantitative metrics relied on an LLM judge. Future iterations will address this via cross-validation with human evaluators (players and designers). Furthermore, future work will explore generative models for dynamic assets, 3D environments, and player-adaptive generation. Overall, this work highlights natural language as a powerful, intuitive interface for procedural game design.

References

Auxtero, A. L. S. (2023). Game environment design creator using artificial intelligence procedural generation. Master thesis (computer science and engineering), NOVA

University, Lisbon, Portugal. Acesso em: 27 jan. 2026.

Ehrig, H., Pfender, M., e Schneider, H. J. (1973). Graph-grammars: An algebraic approach. In *Proceedings of 14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 167–180, Iowa City, Iowa. Institute of Electrical and Electronics Engineers. Acesso em: 27 jan. 2026.

Epyx Inc. (1984). Rogue. Digital game.

Gallotta, R., Liapis, A., e Yannakakis, G. (2024). Consistent game content creation via function calling for large language models. In *Proceedings of 2024 IEEE Conference on Games (CoG)*, pages 1–4, Milan, Italy. Acesso em: 27 jan. 2026.

Huang, S. (2025). Word2minecraft: Generating 3d game levels through large language models. arXiv preprint arXiv:2503.16536. Acesso em: 27 jan. 2026.

Kenney (2026). 1-Bit Pack. `https://kenney.nl/assets/1-bit-pack`. Accessed: 2026-01-23.

Koesnaedi, A. e Istiono, W. (2022). Implementation drunkard's walk algorithm to generate random level in roguelike games. *International Journal of Multidisciplinary Research and Publications*, 5(2):97–103. Acesso em: 27 jan. 2026.

Laurindo, E. K., Oti, K. H., Brocatto, M. d. A., e Belchior, V. (2024). Relatório técnico de desenvolvimento do jogo digital thorn ascent. Trabalho de conclusão de curso (curso superior de tecnologia em jogos digitais), Faculdade de Tecnologia de Americana "Ministro Ralph Biasi", São Paulo. Acesso em: 27 jan. 2026.

Lazaridis, L. e Fragulis, G. F. (2024). Creating a newer and improved procedural content generation (pcg) algorithm with minimal human intervention for computer gaming development. *Computers*, 13(11):304. Acesso em: 27 jan. 2026.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., e Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299. Acesso em: 27 jan. 2026.

Motion Twin (2018). Dead cells. Digital game.

Nasir, M. U., James, S., e Togelius, J. (2024). Word2world: Generating stories and worlds through large language models. arXiv preprint arXiv:2405.06686. Acesso em: 27 jan. 2026.

Pearson, K. (1905). The problem of the random walk. *Nature*, 72(1865):294. Acesso em: 28 jan. 2026.

Perlin, K. (1985). An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pages 287–296, New York, NY, USA. Association for Computing Machinery. Acesso em: 27 jan. 2026.

Perlin, K. (2001). Noise hardware. In Olano, M., editor, *Real-Time Shading SIGGRAPH Course Notes*. ACM SIGGRAPH. Course Notes. Acesso em: 27 jan. 2026.

Salmela, T. (2022). Game development using the open-source godot game engine. Bachelor's thesis (media and arts), Tampere University of Applied Sciences, Tampere, Finlândia. Acesso em: 27 jan. 2026.

Sepänmaa, T. (2024). Procedural level generation in 2d roguelite games. Bachelor's thesis (business information systems), Tampere University of Applied Sciences, Tampere, Finlândia. Acesso em: 27 jan. 2026.

Shorten, C., Pierse, C., Smith, T. B., Cardenas, E., Sharma, A., Trengrove, J., e van Luijt, B. (2024). Structuredrag: Json response formatting with large language models. arXiv preprint arXiv:2408.11061. Acesso em: 27 jan. 2026.

Stiny, G. e Gips, J. (1972). Shape grammars and the generative specification of painting and sculpture. In Freiman, C. V., editor, *Proceedings of Information Processing 71*, pages 1460–1465, Amsterdam. North-Holland. Acesso em: 27 jan. 2026.

Sudhakaran, S., González-Duque, M., Freiberger, M., Glanois, C., Najarro, E., e Risi, S. (2023). Mariogpt: open-ended text2level generation through large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 54213–54227, Red Hook, NY, USA. Curran Associates Inc. Acesso em: 27 jan. 2026.

Team Cherry (2017). Hollow knight. Digital game.

Thorium Entertainment (2020). Undermine. Digital game.

Togelius, J., Yannakakis, G. N., Stanley, K. O., e Browne, C. (2011). Search-based procedural content generation: a taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186. Acesso em: 28 jan. 2026.

Yan, H., Hu, X., Wan, X., Huang, C., Zou, K., e Xu, S. (2023). Inherent limitations of llms regarding spatial information. arXiv preprint arXiv:2312.03042. Acesso em: 27 jan. 2026.