From Pixels to Code: A Process for Generating Scalable Vector Game Art Assets Using Large Language Models

Italo Thiago Felix Dos Santos[1], **Luis Cuevas Rodriguez**[1]
Cristina Souza de Araujo[1] **Jucimar Maia da Silva Junior**[1]

[1] Escola Superior de Tecnologia - Universidade do Estado do Amazonas
Av. Darcy Vargas 1200 - 69050-020 - Manaus - AM - Brazil

[2]Ludus - Laboratório de Tecnologia, Inovação e Economia Criativa (EST/UEA)

{itfds.eng20, lrodriguez,csdaraujo,jjunior}@uea.edu.br

Abstract. *The increasing demand for high-quality visual assets in indie and AAA games poses significant challenges for traditional raster-based generation methods, particularly due to limitations in scalability and resolution independence.* ***Objective:*** *This paper investigates the use of Large Language Models (LLMs) to support the generation of Scalable Vector Graphics (SVG) as game art assets, aiming to enhance production workflows.* ***Methodology:*** *We propose a process for code-based asset generation using locally deployed LLMs, including Qwen, Gemma, and GLM, constrained by rendering and structural rules to ensure valid SVG output. The generated assets are evaluated through quantitative metrics and statistical analysis, combined with a qualitative user study assessing aesthetic coherence, prompt fidelity, and usability for integration into game projects. The performance of local models is benchmarked against Gemini 3 as a strong proprietary baseline.* ***Results:*** *The results demonstrate that locally deployed models can generate structurally valid and scalable assets, with statistical evidence confirming consistent performance differences across models. However, a gap remains in aesthetic quality compared to the proprietary baseline.* ***Conclusion:*** *The findings indicate that LLM-based SVG generation is a viable and scalable approach for supporting iterative game art production workflows, highlighting its potential and limitations in aligning semantic intent with visual design quality.*

Keywords *Large Language Models, Scalable Vector Graphics (SVG), Game Assets, Local Inference.*

1. Introduction

The contemporary video game industry is characterized by a growing demand for high-fidelity visual content, placing significant pressure on developers, particularly within independent studios and emerging markets such as Brazil [Omena 2025]. As game complexity increases, the production of visual assets becomes a critical bottleneck, especially when relying on traditional raster-based approaches such as pixel art. These methods, while widely adopted, are inherently limited by resolution dependency, often resulting in visual degradation when scaled across modern display environments [Rogers 2014].

In this context, Generative Artificial Intelligence (GAI) has emerged as a transformative technological paradigm, reshaping how digital content is created and

integrated into production workflows. Generative AI encompasses multiple model architectures supporting diverse content generation tasks.

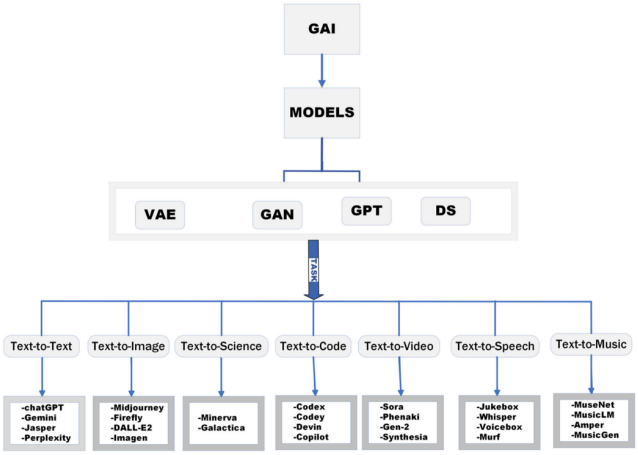

Figure 1. Overview of GAI models and their task categorization. Source: Adapted from [Bengesi et al. 2024].

Despite these advancements, current asset generation pipelines in games are still largely dominated by diffusion-based approaches, which produce high-quality raster images but do not adequately address the requirements of scalability and integration into real-time rendering systems [Stability AI 2023, Midjourney Team 2026]. In parallel, research efforts, including contributions from Brazilian authors, have explored procedural and machine learning-based approaches for content generation [Minini e Assunção 2020, Werneck e Clua 2020], reinforcing the relevance of computational techniques in game production. However, there remains a gap in approaches that combine semantic control, structural precision, and direct applicability to game asset pipelines.

To address this limitation, this work proposes a shift from pixel-based generation to code-driven asset creation. Specifically, we explore the use of Large Language Models (LLMs) for the direct generation of Scalable Vector Graphics (SVG) as game art assets. Unlike raster images, SVG representations encode visual elements as structured geometric primitives, enabling resolution-independent rendering and facilitating integration into game development pipelines [Dahlström et al. 2011]. By leveraging LLMs for structured code generation, it becomes possible to transform textual descriptions into functional and scalable visual assets, supporting more flexible and automated art production workflows. This approach contributes to bridging the gap between generative AI capabilities and practical asset production in game development environments.

This paper introduces a process for generating SVG-based game assets using locally deployed LLMs, including Qwen [Qwen Team 2024], Gemma [Google DeepMind 2026b], and GLM [Zhipu AI Team 2026]. The proposed approach emphasizes not only the generation of visual content but also its structural correctness and usability within game development contexts. To assess its effectiveness, the generated assets are evaluated through both technical validation metrics and qualitative user perception, with comparisons performed against Gemini 3 as a strong proprietary

baseline [Google DeepMind 2026a].

This work makes the following contributions to the field of game art production:

1. We propose a novel process for scalable and code-driven generation of SVG game assets using Large Language Models.
2. We introduce a validation framework for assessing structural correctness in generated SVGs, including metrics such as syntactic validity, viewBox adherence, and spatial consistency.
3. We provide a comparative evaluation between locally deployed LLMs and a proprietary baseline, analyzing trade-offs between performance, reliability, and generation time.
4. We conduct a qualitative user study to assess aesthetic usability and integration potential of generated assets in game development workflows.
5. We identify and analyze failure modes in LLM-based code generation, including reasoning instability and structural inconsistencies, contributing insights into the limitations of current approaches.

2. Related Work

Recent advancements in Generative Artificial Intelligence (GAI) have fundamentally transformed the landscape of game asset creation. Al-Surayhi et al. [Al-Surayhi et al. 2025] conducted a comprehensive narrative review highlighting how GAI tools, ranging from Generative Adversarial Networks (GANs) to diffusion models, are shifting from auxiliary utilities to core design paradigms in the development pipeline. In terms of visual asset production, recent studies indicate that image-generation systems such as DALL·E and Stable Diffusion have significantly reduced time-to-market for concept art [Singh et al. 2026].

In the domain of visual asset generation, the current state of the art is dominated by diffusion models. Architectures detailed in technical reports by Stability AI and the Midjourney Team have demonstrated unprecedented capabilities in translating natural language into high fidelity raster images [Stability AI 2023, Midjourney Team 2026]. However, the stochastic nature of diffusion processes makes it notoriously difficult to control precise geometric boundaries, resulting in assets that require extensive manual post processing before integration into a game engine. Conversely, the evolution of LLMs has introduced robust code generation capabilities.

While traditional Procedural Content Generation via Machine Learning (PCGML) has historically prioritized functional elements like levels and maps over cosmetic assets [Shaker et al. 2016, Summerville et al. 2018], recent shifts in neural architectures demonstrate proficiency in modeling existing content styles for graphical synthesis. In the Brazilian context, this evolution is evidenced by works such as [Minini e Assunção 2020] and [Werneck e Clua 2020], which successfully combined constructive procedural methods with machine learning to automate complex game environments. Despite these advancements, systematic reviews indicate a significant lack of conceptual frameworks to guide developers through specific asset generation techniques effectively, leaving gaps in mapping methods to user needs [Fukaya et al. 2024]. This necessitates new approaches capable of handling semantic directives with structural precision, motivating the exploration of code-based generation.

3. Methodology and System Architecture

The proposed system architecture is designed to facilitate the rapid prototyping and generation of vector assets using consumer-grade hardware. The core objective is to translate creative prompts into valid, renderable SVG code while strictly adhering to W3C specifications to ensure compatibility with standard game engines. This approach leverages the shift in game development where generative artificial intelligence functions as a core design paradigm to automate the production of high-quality assets.

3.1. Pipeline Implementation and Experimental Setup

The generation pipeline is orchestrated via a Node.js application that interfaces with Large Language Models through a local inference, by adopting the OpenAI API specification [OpenAI 2025] for compatibility and interchangeability between various local and proprietary models. A demonstration of this pipeline is available in [Dos Santos 2026][1] (**Figure 2**).

The experimental procedure is structured to evaluate the technical viability of direct code synthesis across distinct visual categories and spatial constraints. To this end, a comprehensive dataset was developed consisting of three specific target resolutions: 64x64 pixels for discrete objects, 64x128 pixels for character-centric assets, and 128x64 pixels for landscape and environments with parallax depth potential. For each of these three resolution categories, ten unique and descriptive prompts were meticulously authored, totaling thirty distinct generative tasks designed to stress-test the models' ability to maintain spatial cohabitation and semantic alignment. This methodology is consistent with recent reviews that highlight how such technologies empower independent developers by reducing production costs and democratizing access to professional-grade content.The generation process itself is divided into three consecutive stages. First, the user prompt is enriched through a context injection module. An example of the outputs produced by this automated pipeline can be observed in Figure 3, which illustrates the execution of the prompt "Hooded necromancer with a glowing staff" extracted from the 64x128 sample, across the different models evaluated in this study.

This module appends technical constraints to the prompt, forcing the LLM to output pure XML data and aiming to prevent conversational filler. The augmented processed by the inference engine using calibrated sampling parameters: temperature of 0.3, a Top-P value of 0.8, and a Top-K value of 40, to balance creative variance with syntactic precision. Finally, the raw output undergoes a rigorous automated validation and post-processing step. In this concluding stage, the system parses the generated string to verify XML well-formedness, confirms the presence of mandatory root elements and closing tags, and evaluates the correct definition of the spatial coordinate system relative to the specified viewBox. Assets that exhibit logical failures, such as out-of-bounds coordinates or insufficient geometric complexity, as monochrome figures, are flagged through a metric-driven adherence check to ensure the reliability of the resulting development pipeline. It is important to note that this study is exploratory in nature, focusing on evaluating the feasibility of code-based SVG generation using LLMs under controlled conditions. While the dataset consists of 30 prompts, it was intentionally designed to cover distinct asset categories and spatial constraints relevant

[1] https://github.com/PixelDust64/SVG-Pixel-Art-Generator

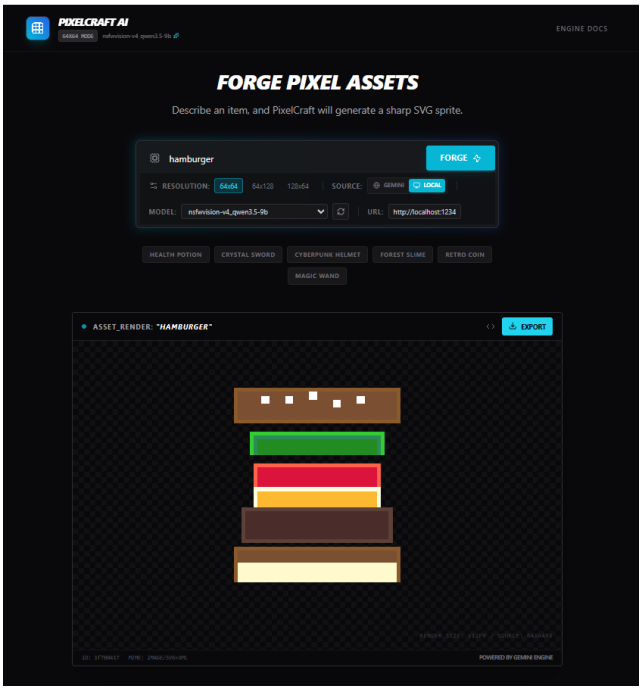

Figure 2. PixelCraft AI graphical user interface: demonstrating the generation of a 'hamburger' asset using the local variation of Qwen 3.5-9B model.

to game development. Future work will expand this dataset to include a broader range of visual styles, complexity levels, and production scenarios.

3.2. Experimental Setup

All experiments were conducted on a workstation equipped with an AMD Ryzen 7 5700X processor, 64 GB of DDR4 RAM, running Windows Server 2025, and a single NVIDIA RTX 5090 GPU.

The evaluation consisted of a comparative analysis of three locally deployed models: Qwen 3.5-9B [Qwen Team 2024], Gemma 4 26B-4B [Google DeepMind 2026b], and GLM-4.7 Flash 30B-A3B [Zhipu AI Team 2026]. All models were deployed using 4-bit GGUF quantization (Q4) for efficient local inference, following prior findings on the impact of GPU architecture and VRAM capacity on long-context tasks [Dos Santos et al. 2026]. Each model was initialized with its maximum available context window. The results obtained from these local models were benchmarked against Google's Gemini 3, used as a proprietary baseline for comparison.

To support a rigorous comparison between models, statistical analyses were conducted on both technical and qualitative evaluation metrics. For binary outcomes, such as syntactic validity and viewBox adherence, Cochran's Q test was applied to assess statistically significant differences across models. For continuous variables, such as generation time, the Friedman test was used due to its suitability for non-parametric repeated measures data. In the qualitative evaluation, a chi-square goodness-of-fit test was employed to determine whether user preferences differed significantly from a uniform distribution. All statistical tests were conducted with a significance level of $\alpha = 0.05$.

4. Evaluation and Results

The evaluation was structured into two primary dimensions: a quantitative analysis of the technical validity of the generated code, and a qualitative assessment of the aesthetic usability of the assets in a game development context.

4.1. Quantitative Technical Analysis

The quantitative evaluation focuses on the mechanical validity and structural consistency of the generated SVG assets under constrained generation conditions. An output is classified as a Success only if the produced code renders correctly in a standard browser environment without syntax errors and preserves structural integrity. Compliance with the viewBox attribute is explicitly measured, as it is a fundamental requirement for resolution independence and dynamic scaling in real time rendering systems. The evaluation metrics are summarized in Table 1 (Gemini 3 is included as a proprietary baseline for comparison).

Table 1. Quantitative Metrics for SVG Asset Generation

Metric	Qwen 3.5	Gemma 4	GLM 4.7	Gemini 3 (Base)
Success Rate (Valid Code)	80%	80%	40%	100%
Average Generation Time (s)	50.2	110.4	160.0	118.5
ViewBox Adherence	80%	75%	45%	100%

The results indicate that although Gemini 3 achieves superior syntactic validity, locally deployed models maintain competitive performance across multiple metrics. Qwen 3.5 demonstrates consistent spatial structuring capability, achieving 80% adherence to the viewBox specification, which supports its applicability in automated asset generation pipelines. The evaluation also reveals the presence of false positive cases in which an SVG is visually rendered within an HTML environment despite containing underlying syntactic inconsistencies in its source code. Additionally, inverse cases are observed where the SVG is flagged as invalid due to formal syntax violations while remaining visually interpretable by browser rendering engines, indicating a discrepancy between parser strictness and rendering tolerance.

Statistical analysis using Cochran's Q test revealed significant differences between models in terms of syntactic validity ($p < 0.05$), indicating that the observed variations in success rate are not due to random variation. Similarly, significant differences were found for viewBox adherence ($p < 0.05$), reinforcing the variability in structural correctness across models. For generation time, the Friedman test indicated statistically significant differences between models ($p < 0.05$), suggesting that performance efficiency varies consistently across the evaluated approaches.

GLM 4.7 exhibited high verbosity and frequent repetitive reasoning before producing SVG output. This behavior contributed to its lower success rate (41%) and higher average generation time (160.0 seconds, calculated only from successful samples). In some cases, the model remained in loops for up to 33 minutes before exhausting its 256k context window, reinforcing the negative impact of excessive reasoning on structured code generation.

4.2. Qualitative User Assessment

A blind preference study was conducted to evaluate the artistic merit and practical viability of the generated assets. A total of 20 participants were recruited, consisting of undergraduate students from Computer Engineering, Computer Science Education, and Design programs. Participants completed an online questionnaire in which the generated SVG assets were presented in randomized order, were asked to evaluate the rendered SVG outputs according to four criteria: visual coherence, absence of visual noise, prompt fidelity (semantic alignment), and immediate readiness for integration into a game project. All participants evaluated the same set of assets.

The results revealed a clear hierarchy in aesthetic quality. Gemini 3 was consistently preferred, securing 55% of the total votes, and was frequently praised for its superior color harmony and more complex compositional layering. Qwen 3.5 ranked second with 25% of the preferences, being favored for its clean and minimalist geometry, which aligns well with flat design principles. Gemma 4 and GLM 4.7 shared the third position in terms of preference, each receiving approximately 10% of the votes. However, they exhibited notable differences in performance. Gemma 4 demonstrated greater reliability by producing a higher number of syntactically valid SVG files, whereas GLM 4.7 frequently generated fragmented paths and unclosed shapes, negatively impacting both structural integrity and visual quality. The preference distribution is presented in Figure 4.

To further validate this preference pattern, a chi-square goodness-of-fit test was conducted to assess whether the observed distribution differed significantly from a uniform distribution. The results indicated a statistically significant difference ($p < 0.05$), confirming that the preference for specific models is not due to random variation.

Figure 3. Visual synthesis of the prompt "Hooded necromancer with a glowing staff" generated by (from left to right): Gemini 3, Qwen 3.5, Gemma 4, and GLM 4.7.

These findings support the qualitative observation that the proprietary baseline is consistently preferred in terms of aesthetic quality and usability. As illustrated in Figure

3, Gemini 3 produces outputs with more refined compositional structure, improved visual balance, and greater adherence to stylistic expectations associated with the prompt.

The observed aesthetic failures highlight a critical disconnection between semantic directives and design intent. As discussed by Al-Surayhi et al. [Al-Surayhi et al. 2025], generative models often produce outputs that satisfy probabilistic prompt alignment without adhering to the functional requirements of a game asset. In our study, even when local models correctly identified semantic labels such as "Cyberpunk," they failed to capture essential design principles, including silhouette readability, spatial balance, and parallax depth.

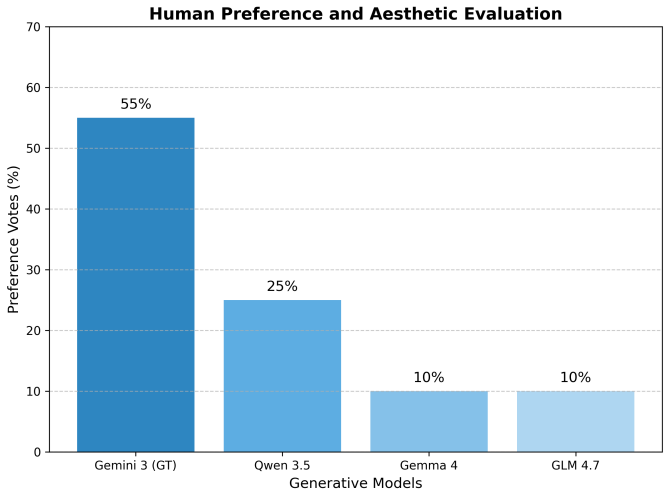

Figure 4. Distribution of Human Preference and Aesthetic Evaluation metrics across the evaluated models.

This indicates that current local LLMs prioritize the syntactic structure of SVG code over high-level design intent, failing to translate the "why" behind the prompt into a functional graphical element that satisfies both usability and aesthetic expectations. As results emphasize the need for specialized evaluation metrics in code-based generation. Unlike diffusion-based approaches that rely on pixel similarity, SVG generation enables the application of strict structural and functional constraints. As suggested by [Fukaya et al. 2024], evaluating such assets requires metrics that go beyond visual plausibility. Table 1 illustrates that while models such as GLM 4.7 may achieve high geometric complexity, their low ViewBox adherence (50%) directly compromises the asset's fill box metric: the ability of an SVG to correctly occupy its designated coordinate space. This functional limitation renders the asset unusable in real-time game environments, regardless of its apparent artistic detail.

4.3. Discussion

The experimental results highlight a critical trade-off between model complexity and functional reliability. A notable phenomenon observed during the testing of MoE (Mixture of Experts) architectures, such as GLM 4.7 and Gemma 4, was the occurrence of infinite reasoning loops. In these cases, the models would engage in repetitive self-correction or internal dialogue, eventually exhausting their 256k context window without producing a terminal SVG tag. This behavior aligns with the "Reasoning Collapse" theory

proposed by [Zhang et al. 2026] and [Shojaee et al. 2025], where increased logical depth beyond a critical threshold leads to a catastrophic failure in task completion. While reducing the context window appeared to partially mitigate this issue in preliminary tests, the correlation between context size and reasoning stability in vector synthesis requires further investigation. Furthermore, as discussed in [Dos Santos et al. 2026], performance is affected by the "race-to-idle" phenomenon in modern GPUs. Due to its smaller size, Qwen 3.5-9B does not fully utilize high-end hardware such as the RTX 5090, resulting in lower effective throughput than the theoretical maximum. However, its efficiency is better reflected in task completion time rather than token generation speed, since output length varies across SVG samples. Despite this, it maintains strong end-to-end performance, supporting its suitability for local deployment.

The local deployment of Qwen 3.5-9B presented significant practical advantages. Notably, it is the smallest model among those evaluated. Beyond the inherent benefits of data privacy and zero inference costs, Qwen's generation speed was remarkably consistent, averaging 50.2 seconds per asset. Its ability to produce reliable geometric structures resulted in a 25% preference rate among participants, consistent with prior work on evaluation metrics for automatically generated graphical assets [Fukaya et al. 2024, Al-Surayhi et al. 2025]. This efficiency is crucial for iterative game development. While proprietary APIs like Gemini 3 offer superior aesthetic complexity, they are subject to network latency and fluctuating server loads. A local pipeline using an optimized, quantized model allows for higher throughput, enabling developers to generate and refine dozens of variations in the same timeframe required for a single cloud-based API call. These findings position LLM-based SVG generation as a complementary tool rather than a replacement for human artists.

The inclusion of statistical analysis strengthens the validity of the findings, demonstrating that the observed differences between models are consistent and not attributable to random variation. From a game art production perspective, this reinforces the reliability of LLM-based pipelines for generating structurally valid assets. However, the statistically significant preference for the proprietary model highlights an important gap in aesthetic quality, indicating that current local models are not yet fully aligned with artistic design expectations. These results suggest that while LLM-based generation is a viable approach for supporting asset production workflows, further improvements are required to bridge the gap between structural correctness and visual design quality.

The performance gap observed between Gemini 3 and the local models may be explained by differences in model scale and training. Proprietary models are typically trained on substantially larger and more diverse datasets and undergo extensive post-training optimization, contributing to stronger semantic and structural representations. This interpretation is consistent with the quantitative results, where Gemini 3 achieved the highest rate of syntactically valid SVG generation and complete viewBox adherence. Additionally, manual inspection showed that Gemini 3 produced more hierarchically organized SVGs, preserving individual visual components as separate graphical elements, whereas local models frequently merged them into shared paths. This structural organization improves editability and suggests that Gemini 3 better captures the semantic relationships required for practical vector asset production.

Use of Generative AI

Large Language Models (LLMs) were used exclusively as part of the experimental evaluation described in this work. The demonstration interface utilized resources from the Google AI Studio application. Additionally, proprietary models such as Gemini 3 were employed solely for text revision and orthographic correction. All research design, analysis, and conclusions remain the responsibility of the authors.

5. Conclusion

This paper investigated the feasibility of using Large Language Models to generate Scalable Vector Graphics (SVG) as game art assets, addressing the limitations of raster-based approaches regarding scalability and resolution independence. The experimental results show that locally deployed LLMs are capable of producing structurally valid SVG code under controlled generation conditions, with Qwen 3.5 and Gemma 4 achieving the highest levels of syntactic correctness among the evaluated local models. However, the qualitative assessment indicates that structural validity alone is insufficient to ensure assets that meet aesthetic expectations for practical game development. The comparison with the proprietary baseline revealed a clear gap in visual quality, suggesting that current local models still struggle to consistently translate semantic intent into coherent visual design. Although the qualitative evaluation involved only twenty participants, it was conceived as an exploratory study aimed at identifying relative preference trends rather than providing population-level estimates. Consequently, the findings should be interpreted within the scope of this experimental setting.

Overall, the results suggest that LLM-based SVG generation represents a promising direction for supporting asset creation workflows, particularly during ideation and rapid prototyping. At the same time, the observed limitations indicate that further advances in semantic alignment, design-aware generation, and aesthetic quality are necessary before locally deployed LLMs can be considered reliable tools for broader adoption in professional game production.

6. Future Work

Future work will explore design-aware constraints and model distillation techniques to transfer the aesthetic capabilities of high-performing models like Gemini 3 into smaller architectures. Fine-tuning on curated game-ready SVG datasets, iterative refinement mechanisms, and deeper integration with game engines and professional art pipelines will also be investigated. To complement these technical improvements, future research will expand the qualitative evaluation by recruiting a more diverse group of participants, including professional game artists, technical artists, game designers, and developers from the game industry and members of the general public representing end users and game consumers. This broader evaluation will enable comparisons between academic participants and experienced professionals, providing a more comprehensive understanding of the practical applicability, aesthetic quality, and usability of LLM-generated SVG assets in real-world game development workflows. Future evaluations will also investigate additional criteria related to production readiness, workflow integration, and asset quality across different artistic styles and game genres.

References

Al-Surayhi, M. M., Al-Qahtani, M. R., Al-Marshadi, S. N., Al-Ghamdi, A. A. Q., e Al-Surayhi, M. M. (2025). Generative artificial intelligence in game design: A narrative review. *Journal of Ecohumanism*, 4(4):1702–1712.

Bengesi, S., El-Sayed, H., Sarker, M. K., Houkpati, Y., Irungu, J., e Oladunni, T. (2024). Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers. *IEEE Access*, 12:69812–69837.

Dahlström, E., Ferraiolo, J., e Jackson, D. (2011). Scalable vector graphics (svg) 1.1 specification. Technical report, World Wide Web Consortium (W3C).

Dos Santos, I. T. F. (2026). Pixelcraft ai: Svg pixel art generator. `https://github.com/PixelDust64/SVG-Pixel-Art-Generator`.

Dos Santos, I. T. F., Barboza, R. S., Almeida, F. P., Araujo, C. S., e Silva Junior, J. M. (2026). Impact of GPU architecture and VRAM on quantized LLM inference for code deobfuscation. In *International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. IEEE.

Fukaya, K., Daylamani-Zad, D., e Agius, H. (2024). Evaluation metrics for intelligent generation of graphical game assets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Google DeepMind (2026a). Gemini 3: Frontier multimodal capabilities. Technical report, Google LLC.

Google DeepMind (2026b). Gemma: Open models based on gemini research. Technical report, Google LLC.

Midjourney Team (2026). Midjourney v6: Generative art and text-to-image models. Accessed: 2026-04-09.

Minini, P. e Assunção, J. (2020). Combining constructive procedural dungeon generation methods with wavefunctioncollapse in top-down 2d games. In *Proceedings of SBGames*, pages 27–29.

Omena, M. (2025). Indústria de games no brasil vira o jogo no cenário global e com apoio das grandes marcas.

OpenAI (2025). Generative ai in game development and asset creation: Api documentation and guidelines. Technical report, OpenAI. Accessed: 2025-11-09.

Qwen Team (2024). Qwen 3.5: Technical report on large language models. Technical report, Alibaba Cloud Intelligence.

Rogers, S. (2014). *Level Up! The Guide to Great Video Game Design*. John Wiley & Sons.

Shaker, N., Togelius, J., e Nelson, M. J. (2016). *Procedural Content Generation in Games*. Springer International Publishing, Cham.

Shojaee, P., Mirzadeh, I., Alizadeh, K., Horton, M., Bengio, S., e Farajtabar, M. (2025). The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity.

Singh, A., Singh, H., e Mattoo, A. (2026). Advancements in generative ai based on large language models and diffusion/gan-based image generation. In *Doctoral Symposium on Computational Intelligence*, pages 193–206. Springer.

Stability AI (2023). Stable diffusion xl 1.0: Technical report. Technical report, Stability AI.

Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, A., Nealen, A., e Togelius, J. (2018). Procedural content generation via machine learning (pcgml). *arXiv preprint arXiv:1702.00539*.

Werneck, M. e Clua, E. W. (2020). Generating procedural dungeons using machine learning methods. In *Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 90–96. IEEE.

Zhang, X., Zhang, Y., Chen, Z., Yu, J., Yang, W., e Song, Z. (2026). Logical phase transitions: Understanding collapse in llm logical reasoning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Zhipu AI Team (2026). Glm-4.7. Accessed: 2026-01-09.