

Demokritos: Interactive, Community-Centred, Self-Improving Shader Generation using Large Language Models

Santiago Ferrando¹, Christian Clark², Tomas Laurenzo³

¹ Universidad de la República, santiago.ferrando@fing.edu.uy

² Universidad de la República, cclark@fing.edu.uy

³ University of Colorado, Boulder, tomas@laurenzo.net

Abstract

The automatic generation of source code from natural language input is advancing rapidly, driven by large-scale language models (LLMs). While LLMs have shown effectiveness in generating code for various languages and contexts, producing shaders—specialised GPU programs essential for computer graphics rendering—from user-provided descriptions remains a complex challenge. This paper presents a free, open-source, web-based platform that leverages state-of-the-art LLM-based techniques to democratise shader generation. Furthermore, by incorporating user feedback on the graphical quality of outputs, the system not only improves its performance but also contributes to an expanding, labelled, open corpus.

Keywords

Computer Graphics, Large Language Models, LLMs, Shaders, Generative AI, Open Source, Open Data

Introduction

The automatic generation of source code from natural language input is advancing rapidly, driven by Large Language Models (LLMs). Although LLMs have shown proficiency in generating code [6], producing shaders from textual prompts remains complex due to the need for both visual fidelity and technical correctness. Demokritos is a free, open-source platform leveraging LLMs and Retrieval-Augmented Generation (RAG) to address this challenge, while also continuously expanding its corpus of shader examples. By integrating user-generated prompts and outputs into its dataset, Demokritos improves over time, enhancing its ability to generate accurate and diverse shaders.

Demokritos

Demokritos [4] [3] generates shader code from natural language prompts by integrating Retrieval-Augmented Generation (RAG) with a preprocessing pipeline tailored for GLSL. Upon receiving a prompt, the system converts it into a vector representation and retrieves similar shader examples from a vector database using FAISS [5]. These examples are provided as context to the Large Language Model (LLM) to guide the generation of more accurate shader code. The output then passes through a preprocessing module that corrects common syntax errors, ensures compatibility with GLSL

standards, and adds necessary declarations for execution on the GPU. This pipeline enables *Demokritos* to address the unique challenges of shader programming while improving both code quality and compilation success rates.

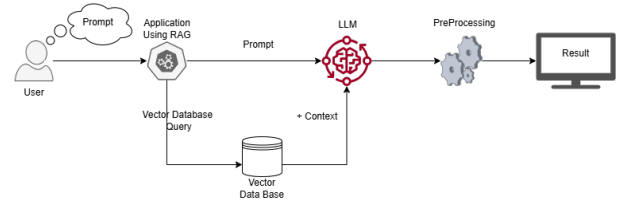


Figure 1: *Demokritos*'s Pipeline.

Discussion

Unlike many of the success cases of LLM-based code generation, the automatization of shader coding requires solving specific, relatively unique difficulties.

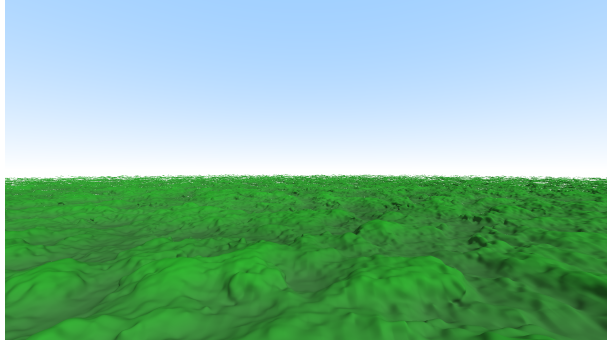
1. Aligning the prompt provided by the user—the description of the image the shader should produce—with the result of the shader's execution.
2. An endless variability in user's goals.
3. A restrictive programming language that lacks many features present in most modern languages.
4. A problem space that only admits a subjective assessment.
5. Additional constraints imposed by the need of produce parallelisable code.

Our high-level model of *Demokritos*'s performance comprises two distinct variables: a) its ability to produce compilable shader code in the GLSL shading language and b) the visual quality of the image that the shader produces.

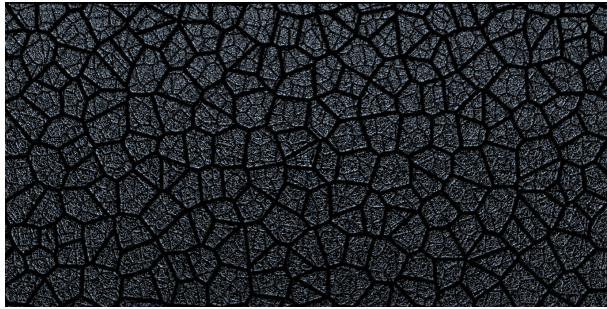
Our evaluation of its current state provides a relatively comprehensive view of the system's strengths and limitations. It also highlights the effectiveness of *Demokritos*'s pipeline design, together with the importance of implementing both RAG and an efficient preprocessing stage.

While hard to measure, democratising the access to shader creation remains one the project main goals and so it provides with a general direction for *Demokritos*'s ongoing and future development.

Visual Results



(a) Prompt: "A forest in a clear sky landscape."



(b) Prompt: "Ice melting."

Figure 2: Examples of shader outputs generated by *Demokritos* from text prompts.

Quantitative Evaluation and Limitations

Demokritos was evaluated based on compilation success rate and user feedback on visual quality. In our own testing, where the same set of prompts was used across different models, the system achieved a 75% compilation success rate, outperforming models like GPT-4o (54.5%), GPT-3.5 (26.1%), Claude.ai (29.7%), and GPT-3.5 fine-tuned [2] (3.3%). These results highlight the impact of RAG in improving shader generation. Users rated most shaders above 7/10 for visual appeal, though complex prompts increased compilation errors, revealing a trade-off between complexity and reliability.

Despite its strengths, *Demokritos* faces limitations. The quality of the RAG corpus is critical, and expanding it with more complex and diverse examples could improve performance. Additionally, complex prompts, while generating richer visuals, also raise error rates. Finally, fine-tuning on a limited dataset yielded minimal improvements, stressing the need for a larger, more representative corpus for model specialization.

Ethical Considerations in Shader Generation Using LLMs

The use of LLMs for shader generation raises ethical concerns regarding authorship, data privacy, and intellectual

property. Shaders, as both functional code and artistic outputs, blur the line between technical and creative work, complicating issues of ownership. Additionally, LLMs are trained on large datasets, which may include publicly available code, raising questions about originality and potential legal implications [7]. Transparency, including metadata on data sources, and clear licensing guidelines are essential to address these issues. Ethical frameworks must balance innovation with respect for intellectual property and privacy, ensuring responsible AI use in creative domains [1].

Conclusions and Future Work

Demokritos demonstrates the feasibility of using LLMs to generate shaders from natural language prompts by integrating RAG and preprocessing techniques, improving alignment with user expectations and compilation success rates. However, limitations persist: fine-tuning on a small dataset yielded limited improvements, highlighting the need for larger and more diverse corpora. Future enhancements could include multimodal inputs (e.g., images or sketches), interactive prompt guidance, and performance evaluation tools. Additionally, exploring open-source LLMs and extending the system to other graphical programming domains, such as real-time rendering, could broaden its impact. These developments will help evolve *Demokritos* into a versatile tool for both technical and creative applications.

References

- [1] Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; Arx, S. v.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; Brynjolfsson, E.; Buch, S.; Card, D.; Castellon, R.; Chatterji, N.; Chen, A.; Creel, K.; Davis, J. Q.; Demszky, D.; Donahue, C.; Doumbouya, M.; Durmus, E.; Ermon, S.; Etchemendy, J.; Ethayarajh, K.; Fei-Fei, L.; Finn, C.; Gale, T.; Gillespie, L.; Goel, K.; Goodman, N.; Grossman, S.; Guha, N.; Hashimoto, T.; Henderson, P.; and Hewitt, e. a. 2021. On the Opportunities and Risks of Foundation Models.
- [2] Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. 1–75.
- [3] Demokritos github page. <https://github.com/eseph/demokritos>.
- [4] Demokritos web site. <https://demokritos.xyz>.
- [5] Hervé Jegou, Matthijs Douze, J. J. 2017. Faiss: A library for efficient similarity search.
- [6] Peng, S.; Kalliamvakou, E.; Cihon, P.; and Demirer, M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.
- [7] Priyanshu, A.; Maurya, Y.; and Hong, Z. 2024. AI Governance and Accountability: An Analysis of Anthropic’s Claude. [eprint: 2407.01557](https://arxiv.org/abs/2407.01557).