Vaporwave Style Graphics Domain Specific Language

Helberth Alejandro, Garcia Melo
Dept. of Computer Engineering
Universidad Distrital Francisco Jose de Caldas
Email: hagarciam@udistrital.edu.co

Abstract—
***Index Terms*—Project guidelines, IEEE LaTeX, engineering education, templates**

I. Introduction

The demand for customized graphic content generation has surged in recent years, driven by the proliferation of digital art communities and the growing accessibility of creative tools.Within this creative renaissance, vaporwave has emerged as a prominent aesthetic movement, characterized by nostalgic references to the 1980s and 1990s, pastel and neon color palettes, glitch effects, retro digital typography, and surreal juxtapositions. Despite its popularity and the abundance of tools available for general-purpose graphic design, the creation of vaporwave art still often relies on manual composition and stylistic intuition, lacking automated or domain-specific support that encapsulates its unique visual grammar. To address this gap, this paper proposes the design and implementation of a domain-specific language (DSL) dedicated to the generation of vaporwave-style graphics. DSLs are programming languages tailored to a specific application domain, offering constructs and abstractions that align closely with domain concepts. Prior work on graphic and image generation using DSLs includes Processing [1], a flexible software sketchbook and a language for visual arts based on Java, and Shadertoy, which leverages GLSL for real-time shader development. More recently, domain-specific generative approaches have been explored in systems such as Context Free [2], which uses a grammar-based DSL to generate recursive artistic patterns, and Manim [3], a Python-based library for programmatically generating mathematical animations. These tools, while powerful, are either too general or focused on other visual languages and do not offer abstractions aligned with vaporwave's visual lexicon. In the field of computational creativity, techniques such as grammar-based generation, rule-based systems, and procedural modeling have been extensively used to encode stylistic patterns into code [4]. These methods can be harnessed within a DSL to encapsulate vaporwave aesthetics in a structured, reusable, and extensible format. Additionally, recent developments in differentiable graphics and neural rendering [5] suggest future integration opportunities, although this paper focuses on symbolic and procedural approaches to maintain interpretability and user control.

Funded by *Your Funding Source* under grant No. 12345.

II. Methods and Materials

The idea to desing this DSL is to offer intuitive, high-level abstractions that reflect the key compositional elements of vaporwave art, including pastel and neon color palettes, retro-futuristic typography, glitch effects, Japanese and Greco-Roman iconography, and grid-based layouts reminiscent of early digital interfaces. To achieve this, the language is implemented as an interpreted, embedded DSL in Python, enabling easy integration with existing graphic generation libraries such as Pillow and Cairo, while maintaining a lightweight syntax focused on artistic expressiveness.

The core objective is to produce a proof-of-concept implementation that enables users to define simple graphic compositions using a small, readable set of custom commands. The DSL will be embedded in Python, allowing us to leverage existing libraries such as Pillow for image manipulation and rendering.

The project will include only the most essential features needed to create typical vaporwave scenes. These features include setting a background color or gradient, placing pre-designed retro-style assets (such as a bust, a sun, or a palm tree), and adding stylized text using nostalgic fonts. The language syntax is intentionally kept simple and declarative. A typical script might include commands like background gradient pink blue, add asset "sun" at center, or text "Aesthetic" at bottom in MS-Gothic. This limited set of commands is sufficient to produce recognizable vaporwave compositions while keeping the language small and the interpreter logic straightforward.

One key design decision is to constrain the visual elements to a fixed canvas size and limit asset placement to a basic grid system. This allows us to avoid dealing with complex coordinate systems while still giving users some creative control. The MVP will support a few hardcoded templates for layout and colors, which can be extended in future versions.

Although limited in scope, this MVP will demonstrate the feasibility of using a DSL to create stylized art in a specific aesthetic genre. It lays the foundation for future expansion while delivering a fully working and visually coherent prototype. The chosen tools and architecture prioritize simplicity, modularity, and clarity, making the solution suitable for both artistic and educational applications.

III. Results

IV. Conclusions

References

[1] Mernik, M., Heering, J., Sloane, A. M. (2005). When and how to develop domain-specific languages. ACM Computing Surveys (CSUR), 37(4), 316–344. (Base teórica sobre DSLs.)

[2] Reas, C., Fry, B. (2007). Processing: A programming handbook for visual designers and artists. MIT Press. (DSL para arte visual interactivo.)

[3] Witkin, A., Kass, M. (2001). Context Free Art: A system for rule-based art. contextfreeart.org (DSL basado en gramáticas para generar imágenes recursivas.)

[4] Grant Sanderson (3Blue1Brown). Manim: Mathematical Animation Engine. https://github.com/3b1b/manim (Librería para animación matemática en Python, con DSL interno.)

[5] Sitzmann, V., Zollhöfer, M., Wetzstein, G. (2020). Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. NeurIPS. (Gráficos diferenciables.)