DSL FOR BUILDING FRACTALS

Liviu Iordan[1*],
Arina Pereteatcu[1] ,
Pavel Țapu[1] ,
Cristian Prodius[1]

[1]*Department of Software Engineering and Automation, gr FAF-223, Faculty of Computers, Informatics and Microelectronics Technical University of Moldova, Chișinău, Republic of Moldova*

*Corresponding author: Liviu Iordan, email: liviu.iordan@isa.utm.md
Tutor: Mariana Catruc, university lecturer

Abstract.
The Scientific Conference paper introduces a Domain Specific Language (DSL) dedicated to simplifying fractal geometry through automated coding. This language aims to make fractal generation easier for both beginners and experts by providing intuitive syntax and functionalities for defining shapes and parameters. The DSL's development involved domain analysis, design principles, and iterative cycles. Results show its effectiveness in creating intricate fractal patterns for various purposes such as art, education, and science. Potential applications include artistic expression, educational exploration, and integration into scientific simulations. This project contributes to computational geometry, offering a valuable tool for fractal enthusiasts and researchers.

Keywords: *Fractals, Domain Specific Language (DSL), Computational Geometry, Syntax and Semantics*

Introduction

Fractals, intricate geometric structures characterized by self-similarity, have fascinated humanity for centuries, from ancient African architecture to modern art movements. Their ubiquity in nature and art underscores their profound influence across disciplines. In this paper, the intersection of fractal geometry with computer science is explored, emphasizing its relevance in contemporary research and creative endeavors.

Fractals, with their inherent ability to exhibit self-similarity across varying scales, serve as a captivating lens through which the convergence of art, nature, and scientific inquiry is explored. As one navigates through the rich tapestry of fractal imagery and its historical significance, the symbiotic relationship between mathematical abstraction and human creativity is unveiled. From the ancient origins of fractal motifs in indigenous architectural marvels to the avant-garde experimentation of 20th-century Surrealist artists, the allure of fractals has transcended cultural boundaries, captivating the imagination of scholars and artisans alike.

In this interdisciplinary exploration, the researchers aim to shed light on the transformative potential of fractal geometry within the realm of computer science. Through an in-depth analysis of fractal analysis techniques and their applications in data science, the study elucidates how these geometric marvels serve as a powerful tool for unraveling the complexities of real-world datasets. Moreover, the paper delves into the realm of domain-specific languages tailored for fractal generation, envisioning a future where intuitive programming interfaces empower individuals across diverse backgrounds to engage in the creation and exploration of fractal art and scientific inquiry.

Fractals in Art and Nature:

The historical and artistic significance of fractals is analyzed, tracing their roots in traditional African architecture to their adoption by Surrealist artists like Max Ernst. Notably, it

discusses the breakthrough analysis of Jackson Pollock's works, highlighting how fractals captivate viewers and contribute to stress reduction.

Fractal Analysis and Data Science:

Fractal analysis intersects with data science, offering insights into complex datasets through techniques like feature extraction, data visualization, and time series analysis. It explores how fractal dimensions enhance machine learning models and aid in understanding intricate data structures.

Domain-Specific Language (DSL) for Fractal Geometry:

The proposal suggests developing a user-friendly DSL tailored for fractal generation, addressing the limitations of existing tools. The DSL aims to democratize fractal exploration by offering intuitive syntax, comprehensive grammar, and versatile features for customization.

Implementation Approach:

The provided DSL would be implemented as an internal DSL, seamlessly integrated with existing programming languages. Leveraging language design principles, the focus is on performance and scalability to handle complex fractal computations efficiently.

Impact and Benefits:

The development of a comprehensive Fractal DSL promises to revolutionize artistic expression, scientific exploration, and educational advancement. By lowering the entry barrier, the DSL empowers users across disciplines to engage meaningfully with fractal geometry, fostering innovation and interdisciplinary collaboration.

Grammar

Grammar in computer science refers to the body of rules that specify a language's syntax. It outlines the proper combinations of a language's various building blocks, including its keywords, operators, variables, and expressions [4]. Formal notations like Backus-Naur Form (BNF) or Extended Backus-Naur Form (EBNF), which offer a succinct approach to define a language's syntax, can be used to represent the grammar of a language. Compilers and interpreters employ grammar to parse and evaluate source code because grammar defines a computer language's structure.

Table 1

Grammar notations

Notation	Description
<foo>	foo is nonterminal
foo	**foo** is terminal
|	separates alternative
→	derives into

$G = (V_N, V_T, P, S)$
V_N - finite set of non-terminal symbols.
V_T - finite set of terminal symbols.
P - finite production rules.
S - start symbol.

S = { <program> }

V_N = { <program>, <statement>, <size_statement>, <color_statement>, <angle_statement>, <iterations_statement>, <shape_statement>, <move_statement>, <scale_statement> <rotate_statement>, <mirror_statement>, <axis>, <draw_statement>, <save_statement>, <filename> }

V_T = { **repeat**, **times**, **start**, **with**, **shape**, **circle**, **square**, **triangle**, **polygon,** **color**, **background**, **scale**, **rotate**, **save**, **as**, **PNG**, **JPG**, **[A-Z]**, **[a-z]**, **[0-9]**, =, **.**, **,**, **[**, **]** }

P = { <program> → <statement> | <statement> <program>
<statement> → <size_statement> | <color_statement> | <angle_statement> | <iterations_statement> | <shape_statement> | <move_statement> | <scale_statement> | <rotate_statement> | <mirror_statement> | <draw_statement> | <save_statement>
<size_statement> → **size** <value>
<color_statement> → **color** <value>
<angle_statement> → **angle** <value>
<iterations_statement> → **iterations** <value>
<shape_statement> → **shape** <shape>
<move_statement> → **move** <value> <value>
<scale_statement> → **scale** <value>
<rotate_statement> → **rotate** <value>
<mirror_statement> → **mirror** <axis>
<axis> → **x** | **y**
<draw_statement> → **draw**
<save_statement> → **save** <filename>
<filename> → <string>
<shape> → **circle** | **square** | **triangle** | **polygon**
<value> → <digit> | <digit> <value> | <string>
<digit> → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
<string> → <char> | <char> <string>
<char> → [**A-Z**] | [**a-z**] | [**0-9**] | = | **.** | **,** | **[** | **]** | ' ' | '_' }

Program showcase

The program generates a fractal image with a red color scheme and a square shape, using an angle of 45 degrees and three iterations of the fractal algorithm. The fractal image is moved to the coordinates (100, 100) on the screen, scaled to half its original size, and rotated 90 degrees. The image is then mirrored along the y-axis before being drawn to the screen. Finally, the resulting image is saved as a PNG file named "fractal.png".

size 800
color red
angle 45
iterations 3
shape square
move 100, 100
scale 0.5
rotate 90
mirror y
draw
save fractal.png

Parsing tree

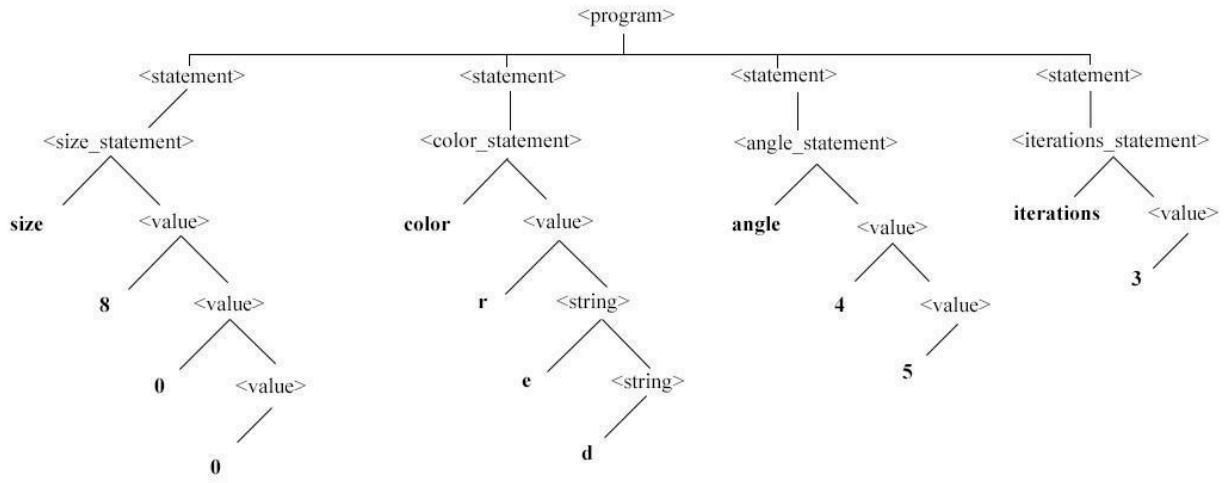

Figure 1. Parsing tree

In the context of domain-specific languages, the parse tree for FracLang can be quite extensive due to the presence of numerous statements. To alleviate the cognitive burden of reading such complex trees, a limited display of statements is proposed. The starting point for FracLang is the <program> production, which allows for the addition of an infinite number of statements. The <statement> non-terminal symbol contains several productions, including the <size_statement>, <color_statement>, <angle_statement>, <iterations_statement>, <scale_statement>, and <rotate_statement>. Each of these statements has similar construction rules and serves a specific purpose in the construction of fractals.

For instance, the <color_statement> determines the color of the fractal by accepting a string representing a color name or a tuple of three integers representing RGB values. The <iterations_statement> is used to control the number of repetitions of the fractal pattern. Too few iterations may mask the fractal's intricate beauty, whereas too many iterations may place a significant strain on the device hardware. Hence, it is important to strike a balance between the iterations and the complexity of the fractal design.

Conclusion

In conclusion, the exploration and creation of fractal patterns present both profound opportunities and significant challenges across various fields. Fractals, with their intricate geometric properties, offer insights into the underlying mechanisms of natural phenomena and provide avenues for artistic expression and scientific inquiry.

However, traditional programming languages and existing fractal generation tools impose barriers to entry, limiting access and usability for a diverse range of users. The technical complexities, restricted flexibility, and fragmented landscape of current tools hinder meaningful engagement with fractal geometry, preventing enthusiasts, artists, educators, and researchers from fully realizing the potential of fractal exploration.

The proposed solution, the development of a user-friendly Fractal Domain-Specific Language (DSL), aims to address these challenges by providing an intuitive platform for creating, exploring, and visualizing intricate fractal patterns. By abstracting away complex algorithms and mathematical concepts while offering high flexibility and accessibility, the Fractal DSL has the potential to democratize fractal exploration and revolutionize artistic expression, scientific exploration, and educational advancement within the realm of fractal geometry.

Through the implementation of the proposed solution, a future is envisioned where individuals across different skill levels can effortlessly engage with fractal geometry, unlocking new avenues of creativity, discovery, and understanding. As the project moves forward, it

remains committed to bridging the gap between fractal enthusiasts and the fascinating world of fractal geometry, fostering innovation and collaboration in this captivating field.

References:

1. Fractal Foundation: What are Fractals? [online], [accessed on 08.03.2024]: https://fractalfoundation.org/
2. HowStuffWorks: How Fractals Work? [online], [accessed on 07.03.2024]: https://science.howstuffworks.com/math-concepts/fractals.htm
3. HowStuffWorks: How DSL Works? [online], [accessed on 07.03.2024]: https://computer.howstuffworks.com/dsl.htm
4. CosmosMagazine: Do fractals exist in nature? [online], [accessed on 06.03.2024]: https://cosmosmagazine.com/science/mathematics/fractals-in-nature/
5. Eclipse: Forums [online], [accessed on 06.03.2024]: https://www.eclipse.org/forums/index.php/t/1107070/
6. Duncans: Nature of Code Fractals [online], [accessed on 08.03.2024]: https://wp.nyu.edu/tischschoolofthearts-duncanfigurski/2021/04/06/nature-of-code-fractals/
7. Ola Bini: Fractal Programming [online], [accessed on 09.03.2024]: https://olabini.se/blog/2008/06/fractal-programming/
8. MANDELBROT, B. *The fractal geometry of nature*. Times Books, 1982
9. MERNIK, M., HEERING, J., & SLOANE, A. M. When and how to develop domain-specific languages. In: *ACM Computing Surveys* (CSUR), 2005, 37(4), 316-344.
10. FOWLER, M. *Domain-specific languages*. Addison-Wesley Professional, 2010
11. AHO, A. V., LAM, M. S., SETHI, R., & ULLMAN, J. D. *Compilers: principles, techniques, and tools. Pearson Education.* Addison-Wesley, 2006