FRACLANG - A DOMAIN SPECIFIC LANGUAGE FOR BUILDING FRACTALS

Ciprian BOTNARI[1*], Grigore GUZUN[1], Emanuil CUCOȘ[1], Artiom MIROVSKI[1]

[1]*Department of Software Engineering and Automatics, Group FAF-213, Faculty of Computers, Informatics and Microelectronics, Technical University of Moldova, Chișinău, Republic of Moldova*

*Corresponding author: Ciprian Botnari, ciprian.botnari@isa.utm.md

Scientific coordinator: Mariana CATRUC, University Lecturer

Abstract. *Fractals are mathematical structures that are fascinating due to their self-similar patterns, which repeat at different scales. However, building fractals requires a considerable amount of mathematical knowledge and expertise. To simplify the process, domain-specific languages (DSLs) have been developed that allow users to create fractals using simple syntax and commands. This article discusses a domain-specific language that is flexible and convenient for building fractals, FracLang.*

Keywords*: domain specific language, fractal, fractal language, grammar, parse tree*

Introduction

Mathematicians and computer scientists have been fascinated by fractals for many years. Several natural phenomena, including coastlines, clouds, snowflakes, and even the human body, contain fractals. They have also been utilized to produce stunning animations and digital art. Fractal geometry is an area of mathematics that examines objects that exhibit self-similarity at various scales [1].

Mathematicians and computer scientists have created algorithms that iteratively apply a set of rules to a geometric object in order to produce fractals. To create a new version of the item, these rules specify how the object should be rotated, scaled, and modified. Recursively repeating the technique results in a string of items that eventually congregate into a fractal shape. Domain-specific languages (DSLs) have been created by programmers to make it easier for users to explain fractal algorithms in a clear and understandable manner.

Language Design

A DSL is a programming language designed for a specific domain and is created to simplify programming tasks in a specific field, and has a lower learning curve than a general-purpose programming language [2]. DSLs can be either external or internal. External DSLs have their own syntax and are often compiled to a target language. Internal DSLs are embedded within a host language and use the syntax and structure of the host language [3]. FracLang is a language that enables users to create fractals with minimal coding as it has a set of predefined statements that are easy to use and learn, and addresses the challenges associated with building fractals.

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

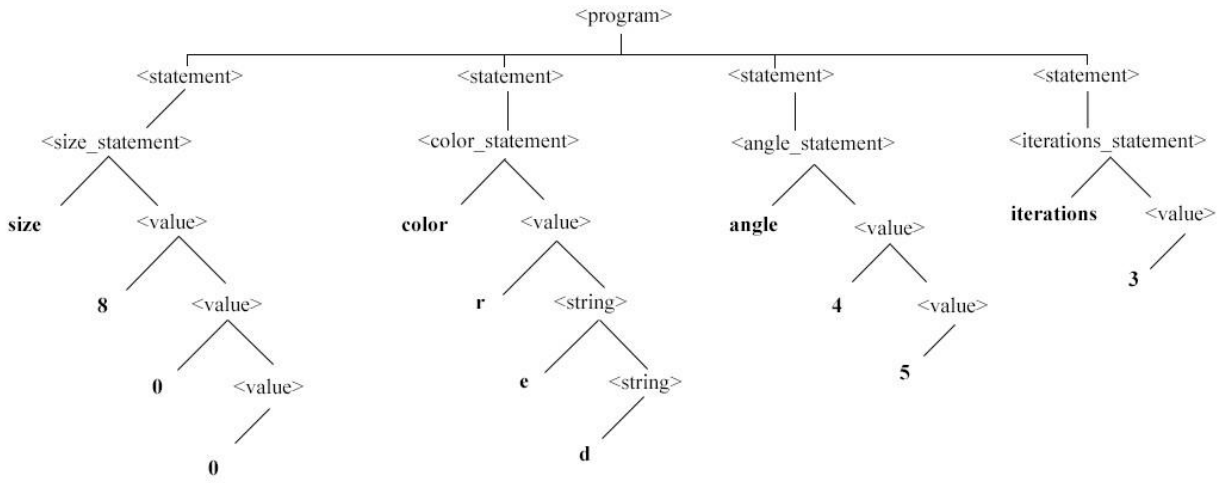

Figure 1. Parsing tree

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

FracLang is an innovative domain-specific language that is specifically designed to make it easier for both experts and novices to generate stunning fractals. With its intuitive grammar and user-friendly interface, The language is accessible to users with varying levels of experience, making it an ideal tool for anyone interested in exploring the fascinating world of fractals.

What makes FracLang stand out from other programming languages is its ability to simplify the complex task of generating fractals. FracLang achieves this by providing a set of easy-to-use

commands that allow users to create complex fractal patterns with ease. This makes it possible for anyone, regardless of their experience level, to create intricate and beautiful fractal designs without having to learn complex programming languages.

It is just one of many domain-specific languages developed to simplify complex tasks in various fields. Its success serves as a testament to the power of DSLs in making complex tasks more accessible to a wider audience. As technology advances, it is likely that we will see more DSLs developed for other fields, making it easier for people to accomplish complex tasks and achieve their goals.

In conclusion, FracLang is a powerful and innovative tool that simplifies the process of generating fractals for both experts and novices. Its intuitive grammar and user-friendly interface make it accessible to users of varying levels of experience. As the demand for DSLs continues to grow, it is exciting to see what new innovations and discoveries will be made possible in the future.

References

1. MANDELBROT, B. *The fractal geometry of nature*. Times Books, 1982
2. 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.
3. FOWLER, M. *Domain-specific languages*. Addison-Wesley Professional, 2010
4. AHO, A. V., LAM, M. S., SETHI, R., & ULLMAN, J. D. *Compilers: principles, techniques, and tools. Pearson Education.* Addison-Wesley, 2006