Introduction

Fractals were first introduced by mathematician Benoit Mandelbrot in the 1970s. They are self-similar patterns that repeat themselves at different scales, meaning that the same shape is repeated again and again, at ever-decreasing sizes. This property is known as "self-similarity," and it is what gives fractals their distinct and recognizable appearance.

Fractals can be found in nature, from the branching patterns of trees and ferns to the intricate designs found on snowflakes and seashells. They can also be created using mathematical formulas and algorithms, using computer programs to visualize and manipulate the resulting patterns.

Fractals are more than just interesting patterns, however. They have a wide range of practical applications, such as in image compression and rendering, landscape generation for video games and simulations, and modeling of complex systems in physics and biology. They have also been used in art and design, where they can be used to create intricate and beautiful patterns, and in education, where they are used to teach mathematical concepts such as recursion and iteration.

The study of fractals has led to many new discoveries and insights into the nature of complex and chaotic systems, and has had a significant impact on many different fields. Fractal geometry has also influenced the way we understand the structure of natural systems, and has led to new insights into areas such as turbulence, fluid dynamics, and crystal growth.

Fractals have had a significant impact on many different fields, and continue to inspire new discoveries and applications in areas ranging from physics and biology to art and education.

Tools

The existing tools and languages that are used in the domain of fractals include MATLAB, Python, and JavaScript libraries like d3.js and p5.js. Understanding the limitations and shortcomings of these existing tools and languages will help us identify the problem that our DSL will solve.

Problem

Based on our research, we have identified problems such as:

- **Steep Learning Curve**: Many existing tools and languages for building fractals have a steep learning curve, making it difficult for students and educators to get started. A new DSL designed specifically for this purpose could be more accessible and user-friendly.
- **Limited Customization**: Some existing tools and languages have limited options for customizing fractals, which can be frustrating for students and educators who want to experiment with different designs and patterns. A new DSL could offer more flexibility in terms of customization.
- **Lack of Interactivity**: Some existing tools and languages are not interactive, which means that students and educators cannot see the results of their changes in real-time. A new DSL could provide immediate feedback and make the process of building fractals more engaging and interactive.

- **Incompatibility**: Some existing tools and languages are not compatible with certain operating systems or hardware, which can limit their accessibility. A new DSL that is designed to work on a variety of platforms and devices could make it more widely accessible to students and educators.

Target Audience

The target audience for our new DSL for building fractals is primarily educators and students. Educators in this context may include teachers at the K-12 level as well as instructors in higher education who teach courses related to mathematics, computer science, art, and other related subjects. Students in this context may range from elementary school students to college students who are interested in learning more about fractals and experimenting with different designs and patterns.

These users are likely to have varying levels of experience and expertise with fractals, ranging from complete beginners to more advanced users. As a result, the DSL should be designed to accommodate users at all levels and should offer both basic and advanced features to allow for greater creativity and experimentation.

Additionally, since the target audience is primarily in the educational realm, the DSL should be designed to be user-friendly, accessible, and engaging. It should be easy for educators to integrate into their lesson plans and for students to use on their own, both in the classroom and at home.