# DSL for Building Fractals

Team 16

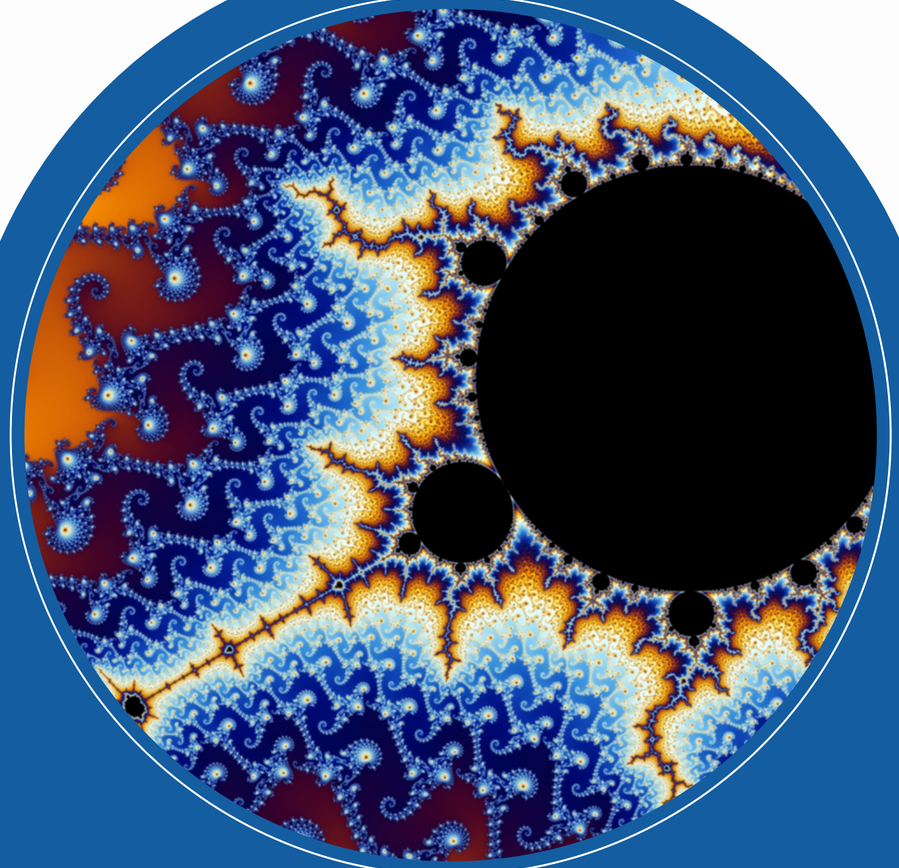
Iordan Liviu FAF-223
Țapu Pavel FAF-223
Pereteatcu Arina FAF-223
Prodius Cristian FAF-223

# Overview



# Problem Description

- Ubiquity
- Barrier to Entry
- Integration Across Disciplines
- Visual Appeal
- Mathematical Complexity



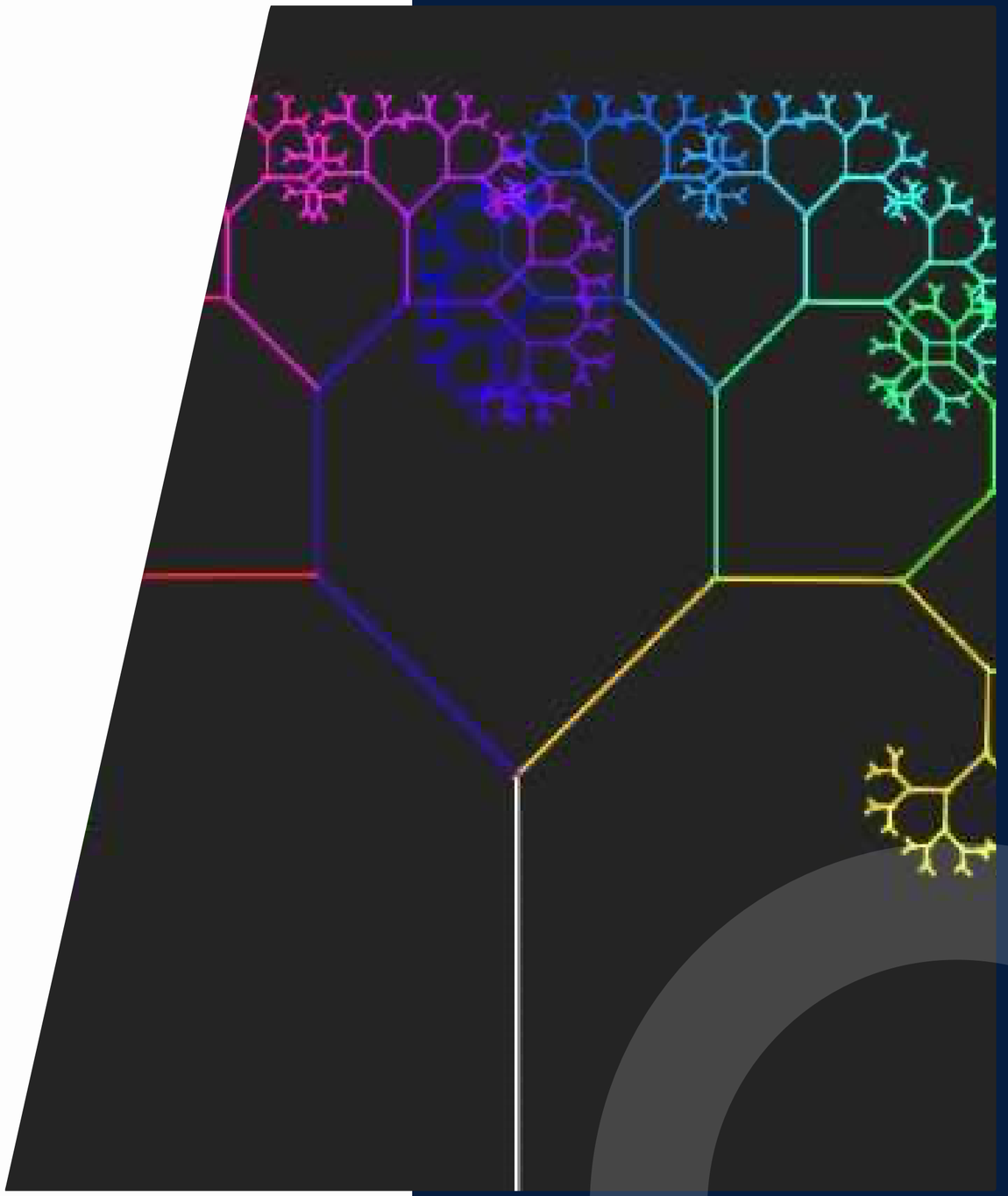
# Problem Statement

**Need for a Specialized DSL**

**Challenges in Fractal Creation**

**Democratizing Fractal Generation**

**User-Friendly Interface**



# SOLUTION PROPOSAL

To address the challenges and limitations outlined in the problem description and analysis, we propose the development of a comprehensive Fractal Domain-Specific Language (DSL)

## Key Features of the Fractal DSL

Intuitive Functionality
High Flexibility
Lowered Technical Barrier

## Implementation Approach

-Design Considerations
- Technical Considerations

## Impact and Benefit

- Artistic Expression
- Scientific Exploration
- Public Understanding

**In conclusion, a Fractal DSL could revolutionize art, science, and education in fractal geometry, empowering users of all levels to explore its mesmerizing beauty.**

# Solution Statement

Our solution is to create a user-friendly DSL for fractal generation, offering an intuitive platform for users to create and visualize intricate patterns without advanced programming knowledge.

## Key Components

- Intuitive Syntax
- Built-in Functions and Libraries
- Flexibility and Customization
- User-Friendly Interface
- Documentation and Suppor

## Implementation Approach

- Requirements Gathering
- Design and Development
- Testing and Iteration
- Deployment and Distribution

## Expected Impact

The Fractal DSL revolutionizes fractal exploration, making mathematical beauty accessible. Our solution empowers users of all levels, fostering expression, exploration, and education in fractal geometry.

# Key features of our DSL

### Intuitive sintax

Designed with keywords and operators that align with the concepts of fractal generation.
Reduces the learning curve for users familiar with fractal mathematics or visual concepts.

### Data structure

Supports numeric types (integers, floats), Booleans, strings, lists, and dictionaries.
Allows for custom data types (classes) to represent complex fractal structures.

### Scope management

Global variables
Local variables
Enclosing Scope & Shadowing

### Control Flow

Employs if statements for conditional execution.
Implements for loops for iterative processes.

# Grammar Essentials of our DSL

### Lexical Considerations

Case-sensitive keywords and identifiers.
Uses comments (#) for code clarity.
Relies on whitespace for readability.

### Operators

Arithmetic (+, -, *, /, %, **)
Comparison (==, !=, <, >, <=, >=)
Logical (and, or, not)

### Data types

Integer (example: width = 800)
Boolean (example: use_colors = True)
String (example: fractal_name = "Mandelbrot Set")

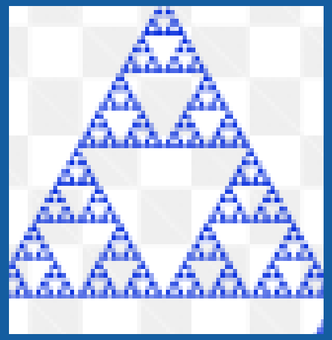
### Error handling

Syntax and Runtime Error Detection
Exception Handling: Enables graceful recovery from errors during fractal generation,
allowing the program to continue or adjust.

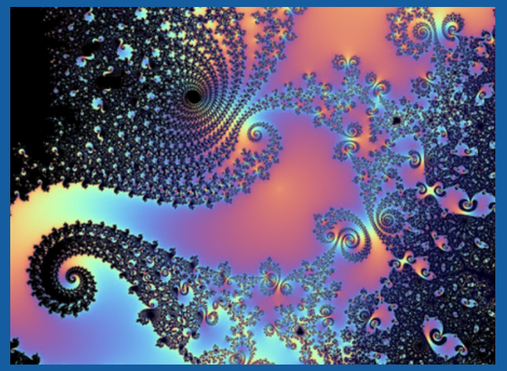
# FUTURE GOALS


- powerful parser generator for reading, processing, executing, or translating structured text or binary files

# FUTURE GOALS



**Sierpinski Triangle**
the recursive subdivision of a triangle into smaller equilateral triangles



**Julia Set**
boundary patterns arising from iterating complex numbers under a quadratic polynomial

**Dragon**
iterated system of line segments

# Thank you