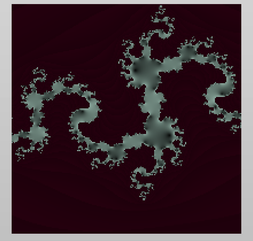
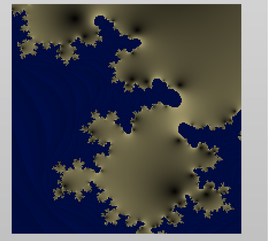


CS 423/523 Project 3: Generating Fractals with Genetic Algorithms

Lucas Nunno (lnunno), Mark Montoya (mmontoya), April Suknot (asuknot)



Introduction

In learning about Complex Adaptive Systems, one notion that we took particular interest in was the appearance of fractals throughout nature. Learning about fractals and genetic algorithms fueled a desire to use genetic algorithms as explained in [2], to create interesting images of fractals by evolving the different components of fractal equations like the equations described in [1]. The primary goals for the project were:

1. Generate increasingly interesting fractals through the use of genetic algorithms. This includes the use of crossover, mutation, and selection to generate new fractals.
2. Allow a user to select the fractals that they think are the "coolest" and generate new fractals with these as a basis.
3. Be able to zoom and inspect different areas of any fractal displayed.

Methods: Initial Approach

1. Starting with the Mandelbrot equation:

$$Z_{n+1} = Z_n^2 + c$$

2. Mutate the exponent of Z and the C value to create new fractals. This is done using a Gaussian distributed mutation.
3. Employ roulette selection to select candidates for the next generation.
4. Crossover these new parents.
5. Display these fractals to the user for selection.
6. Repeat process with selected fractals.

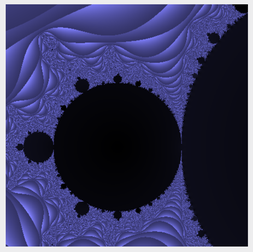


Figure 1: The Mandelbrot Set, the inspiration for the project.

Binary Expression Trees

We began our approach with the use of binary expression trees. These trees represent an equation as a binary tree where operations are branches (have two children) and constants (complex and real) are leaves. This would allow for intuitive crossover, substitution, deletion, and insertion of fractal equations.

Methods: Implementation

For our implementation, we were able to stick with much of what we had planned to use in our approach. However, we found that the expression tree functions performed on operators would cause the images to render as gradients. As a result, we decided to abandon the expression tree and only perform genetic algorithm functions on the constants.

Moving Forward Without Expression Trees

We found that even without the expression tree, we could still use mutation and crossover to evolve the Z exponent and the constant value in the Mandlebrot equation, based on the user's selection. An object named "Fractal" was created, and we used this to keep track of the color, Z exponent, and constant for each fractal shown displayed. From here, we were able to implement selection in which the panels selected by the user are used to populate the next generation.

Swing vs OpenGL

The language that we chose to use for this project was Java. As a result, we began with using the Swing library to draw the images. While the images were of great quality, the computations performed per pixel caused the program to run very slowly. Our solution to that was OpenGL's shading language (GLSL). Using GLSL, we were able to take advantage of the Graphics Processing Unit (GPU) to speed up rendering. This did come at a cost. We were unable to use Apache's Complex library. In spite of that, we were able to compensate with the use of De Moivre's Theorem [3]:

$$(a + bi)^n = r^n(\cos(n\theta) + i\sin(n\theta))$$

This allowed us to generate the same interesting fractals at a much faster rate. This made user interaction much more pleasant for zooming in and moving the panels around to help decide which images to select.

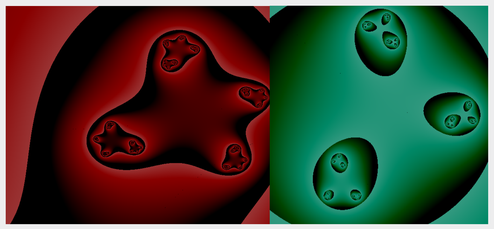


Figure 2: Images generated in early stages of testing fractals with the graphics components of the program.

Results

While we were unable to approach the project as we had initially planned with the expression trees, we were able to meet our goals with our new approach. As can be seen in Figure 3 below, the user is able to select three panels for each generation to be used in the next iteration.

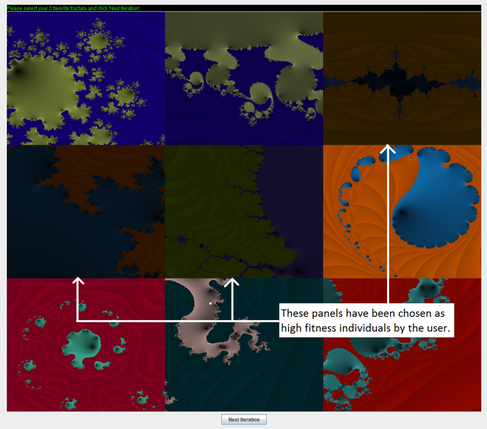


Figure 3: An image showing a user's selection on an early iteration of the algorithm.

After each generation, the fractals displayed take a form more similar to the images that the user selects. The user's selection has an effect on the color displayed as well as the fractal equation that is used for future generations. This notion is illustrated in Figure 4, in which it can be seen that the program will begin to display fractals very similar to the ones selected by the user.

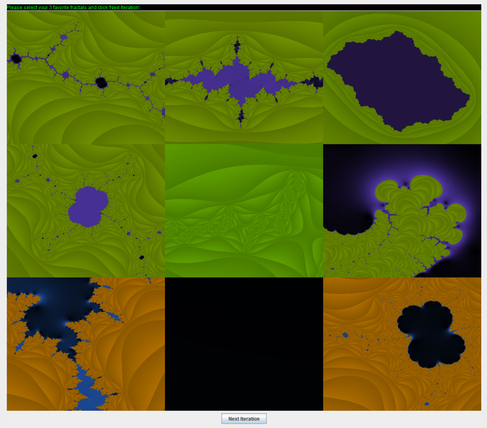


Figure 4: An image showing a later generation in which the algorithm has become more aware of the user's preference.

Conclusions

Fractals are a very common occurrence in many parts of nature, in evolving different fractals, we were able to see fractals that reminded us of leaves, coastlines, and many other things that we see in everyday life but may ignore the complexity of it.
In completing our implementation, we were able to realize our original goals:

1. We were able to implement the genetic algorithm to evolve the constants and Z exponents in using mutation, roulette selection, and crossover, in addition to a user-dictated fitness function for elitism.
2. The user is able to select his or her three favorite fractals each generation and see how those are used to populate the next generation.
3. With the use of the GLSL library, we were able to take advantage of the GPU to efficiently zoom in and move each fractal around in its home panel. In addition to this, users are able to save their favorite views to .png images.

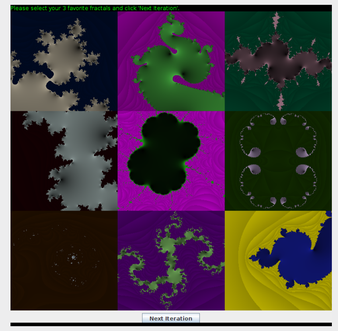


Figure 5: Another example of the diversity available in the early generations of the algorithm.

The project did have some degree of difficulty, especially with finding a successful approach in evolving the fractals. However, once we found an approach that allowed us to efficiently display the fractals with the correct form, we were able to experiment a great deal with the genetic algorithms allowing us to produce interesting and satisfying results.

References

[1] Michael McGoodwin. Julia Jewels: An Exploration of Julia Sets @ONLINE, March 2000. URL: http://mcgoodwin.net/julia/juliajewels.html
[2] Melanie Mitchell. *Complexity A Guided Tour*. Oxford University Press, USA, 2011.
[3] Bruce Simmons. De Moivre's Theorem @ONLINE, August 2012. URL: http://www.mathwords.com/d/demoivres_thm.htm