Fractal: Math and Art

Chujiao Ma

May 9, 2007

Introduction

In essence, fractals are complex geometric patterns. However, it is used more often than many people are aware of. Fractals are being used in applications from image compression to movement of particles in physics to biological analysis. Not only is it useful in technological applications, fractals also shows up in many forms in nature, from trees to clouds to leaves to general landscapes-thus enable beautiful arts to be produced.

In this project, I focused on the mathematics of fractals, the programming aspect, and the art they produced. The project is separated into three main components. The first part is to understand the math and derive equations with which to code. The second part is to learn the soft wares necessary and then implement the coding for a simple fractal. The third part is to grasp how to produce different image, codes and produce more complex imagery for final art work, such as leaves, trees and landscape.

Mathematics

Fractal images are the result of coloring and graphing mathematical functions. The two main kind of functions used in fractals are iterative function system (IFS), and equations involving complex numbers. For this project I focused on the fractals generated by complex numbers because the complex fractals generate a colored image while the IFS generate a plot that is only one colored.

First of all I learned about complex numbers and how they generate fractals. A complex number is frequently written as $x + i * y$, where x is a number and represents the real component of the equation while the term i*y represent the imaginary component of the complex number. As for how they generate fractal images, it is it is easier to learn using examples. After researching online for a week on the concept of fractal math and how they produce images, I decide to start with an existing equation that generates image and see how it works. The first fractal I examined is the Julia set, the equation of which is in form of $Z_{n+1} = Z_n^2 + C$, where C is a complex number and a constant. Z_0 is a point on the xy plane, where $Z = a + i * y$. Using the equation $Z_{n+1} = Z_n^2 + C$, where n is the iteration number, we calculate how many iterations it takes for the Z to

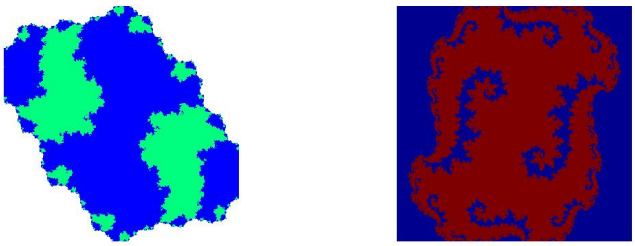

Figure 1: The first image above was produced with the constant c= -.1+.4*i while the second image was produced with the constant c= .25+.15*i in the Julia program I found.

approach infinity, usually when Z exceeds a 2 by 2 plane, it is confirmed that it will approach infinity. The number of iteration it takes to approach infinity is calculated for every point on the xy plane, and each iteration number is given a different color. Since some points on the xy plane will have different iteration number than others, the colors will also be different. Thus, the image for the Julia set is generated. By using different equations and constants, different fractal images can be generated.

Programming

The mathematical concept of it seems to be simple enough, but the programming component proves to be more difficult. First of all I chose to use MATLAB because it is already installed on our computers, and of the few programming languages I understand, it is the one that I am most comfortable with. I start by downloading and examine the program for Julia set taken from MATLAB M-File database online. Then based on my understanding of the Julia set program, I experimented with changing the constants first, depending on how fast it approaches infinity, or if at all, the coloring of the image sometimes turns out well and sometimes only turns up as a solid color. Fig. 1 shows two of the more successful results of changing constants within the Julia set program.

After experimenting with the constants in the given Julia program, I then start writing my own program. One of the images that I liked was called the Burning Ship, after some searching the equations are given, but no programs were available. Instead of an equation for the complex number like the Julia set, the equation for the burning ship have an equation for both x and y components, thus requiring x and y values to be calculated separately and then combined within the for loop of the program. Due to my limited knowledge of MATLAB command, the debugging took longer than expected. However, the program was finally debugged and image generated successfully as shown in Fig. 2.

From this point, I start to experiment with colors in MATLAB as well as

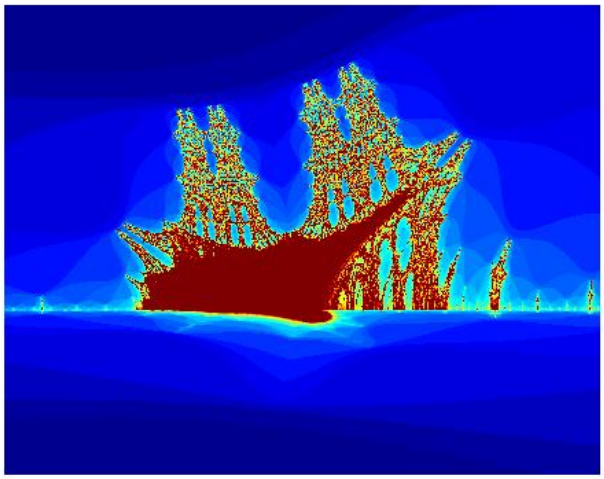

Figure 2: The image above was using MATLAB with its default colormap.

attempting to change or write my own equations, which is one of the difficult part and what I worked on for the rest of my project. Depending on the constants and equation used, it is very easy to get a function that either approaches infinity too quickly, doesn't approach infinity at all, or approaches infinity at similar iteration, all of which usually produces a solid black screen when the program is run. By taking sines or cosines different components of complex number and zooming in or out on certain parts, I came up with a few original images. However, since MATLAB is a program that is more suited to math, the available colormaps are limited and some of which are not exactly pretty. The resolution of the image are also not ideal and does not translate into .jpeg as well as I hoped. To increase the resolution of the image, I separated the xy plane into more points with which the program needs to calculate, however, there is a limit to MATLAB's memory, which seems to be at 3000 by 3000 points before it crashes, and the program takes a long time.

Results

One of the first original image I came up with is the Forest Sky made with the equation $Z = cos(Z)^2 + c$, and is shown in Fig. 3. Since there is no existing colormap with various shades of green together, I tried to write an equation for one myself and failed, so I decide to take existing colormaps and combine the sections of color that I wanted together to create a new colormap, which I used to color Fig. 3 so that it looks more like a forest.

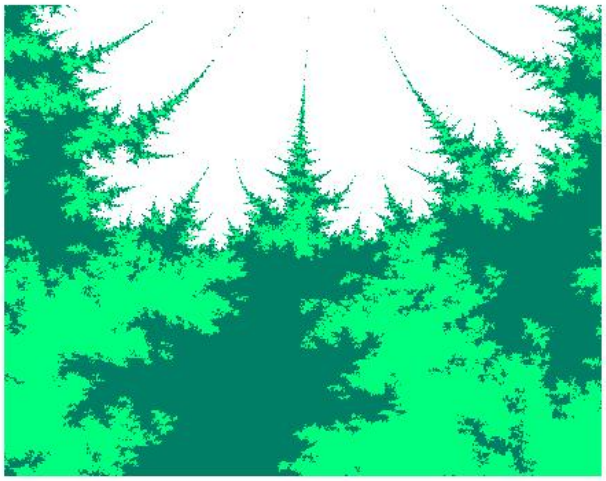

Figure 3: The Forest Sky was a zoomed in section of the image generated with equation $Z = cos(Z)^2 + c$.

Another image I made came from the equation $C = sin(C + cos(C) + C$. It does not look like an landscape, but rather an abstract art form, on which I decide to experiment different colors on. The result of which are shown in Fig. 4.

The last significant original image I came up with was Forest Pond, shown in Fig. 5 of which I also have to make the colormap myself. The colormap consist of different shades of green and different shades of brown.

On a side note, I tried to create combined images in MATLAB, which can be done, but not if the two images require different colormaps. Thus I tried to combine the images in Photoshop, shown in Fig. 6, which did not turn out as well as expected.

Due to time constraints, I was not able to came up with more significant equations for fractal images. Fig. 7, the Ocean Wave at Night, was the last image I generated for this project.

Conclusion

Through this project, I gained a much better understanding of complex numbers and fractals as well as numerous commands in MATLAB involving imagery and colors. This project is very enjoyable and can be taken much further. One of the possible change is to use a program that is better suited for image generation, some of which that were recommended to me are C and Java. Another area that

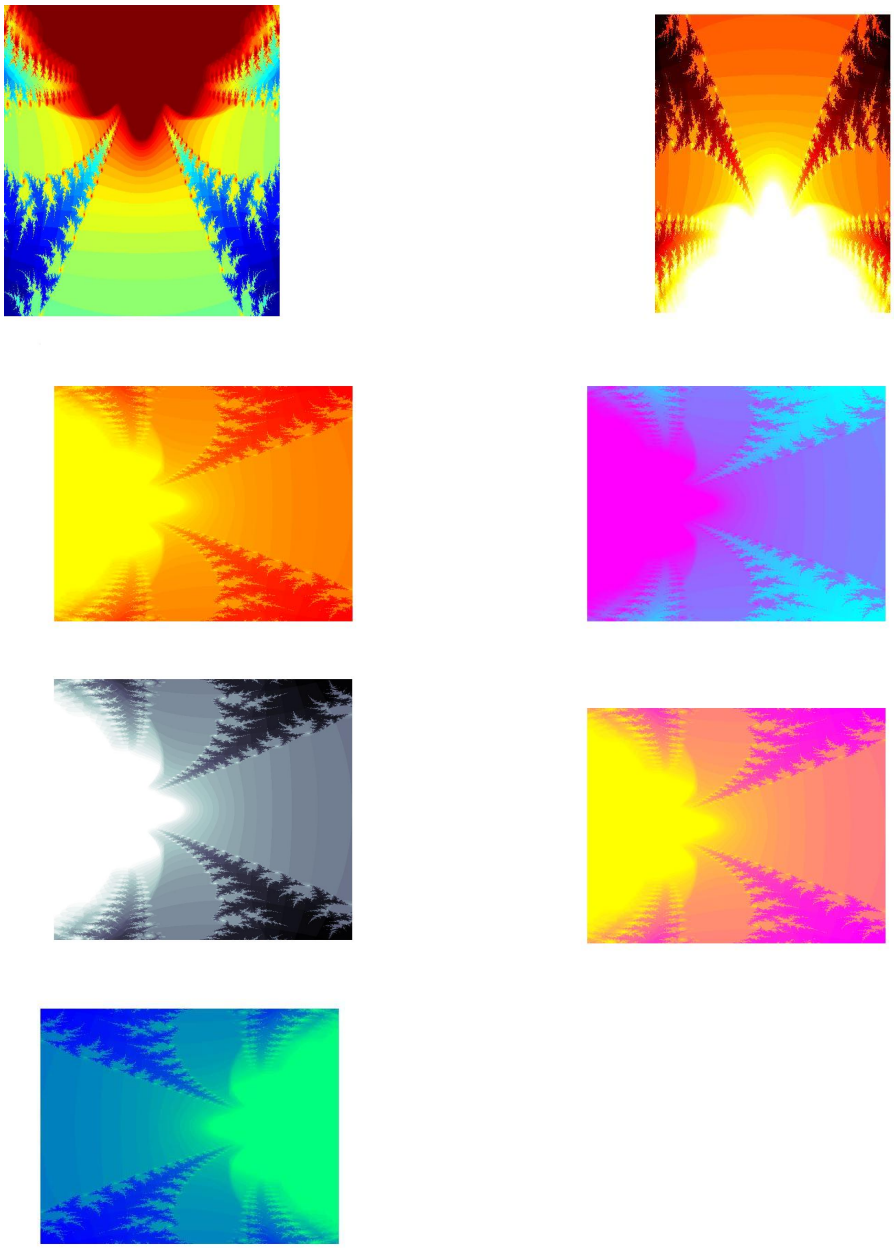

Figure 4: Different coloring and orientation of an image that was a zoomed in section of the image generated with equation $C = sin(C + cos(C) + C$, which was based on Mandelbrot, $Z = Z^2 + C$.

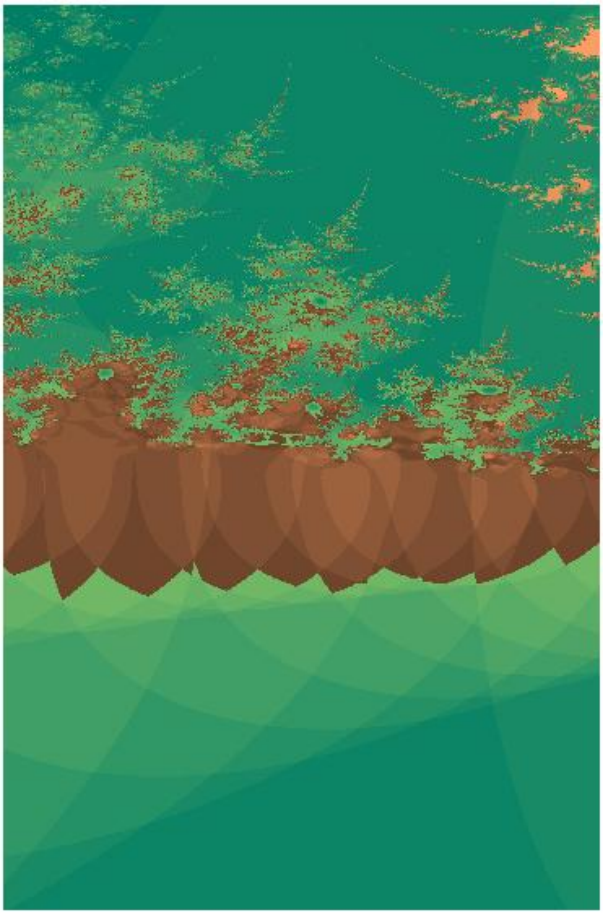

Figure 5: The Forest Pond was a zoomed in section of the image generated with equation $Z = (sin(Z) + cos(Z))./2 + C;$.

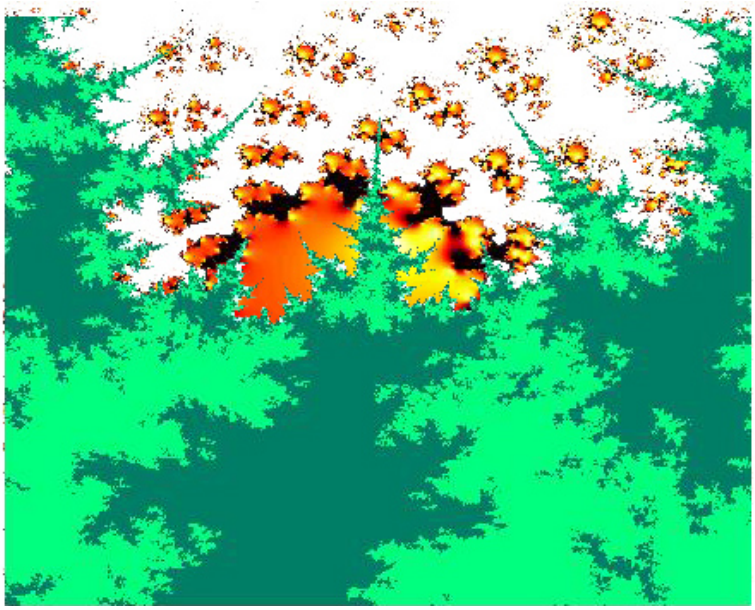

Figure 6: This image was the combination of two images, the green forest image was from an equation I wrote, and the orange fire sun image was from an equation which derived from the Julia set.

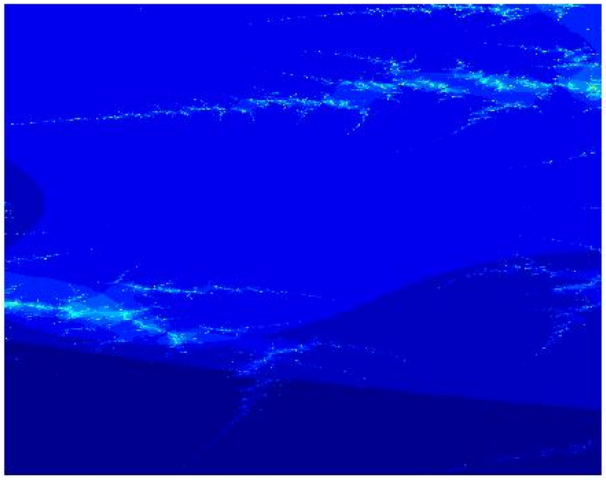

Figure 7: The Ocean Wave was a zoomed in section of the image generated with equation $Z = (sin(Z) + cos(Z)) + C$.

can be improved is the colormap, and learning how to write functions to compose desired colormap with smooth gradient would be very helpful. Lastly, with more research and experiments with equations, constants and boundary conditions many other fascinating images can be generated and numerous artistic possibilities discovered.