Introduction to Fractals

1 Overview

In nature, in just about every place we look, we can find self-similar, scale-invariant patterns. If we take a cloud in the sky at view it up close, we can see similar geometric patterns as when viewed from afar. This same scale-invariant self-similarity can be seen on rough surface of mountains, the branching patterns in trees and bushes, and even in the structure of man-made road networks and cities layouts. In fact, even in the structure of your circulatory system, you can find blood vessels, which branch off into other vessels, and eventually into capillaries in self-similar manner. Fig. 1 is of a plant which displays self-similar branching patterns; each branch splits off into three or more smaller branches of almost approximately equal angles.

A fractal, a term coined by the pioneer of fractal theory, Benoit Mandelbrot, is a geometric shape that is produced by repeated application of a function which takes as inputs its outputs, and produces geometry that has properties of self-similarity and scale-invariance. By successively applying the fractal function, it is possible to produce arbitrarily complex geometry at any desired level of detail.

As computer graphics students, we are interested in visually reproducing the complex geometry and behavior of objects within the world. Fractals modelings offers a convenient approach to modeling objects which have the fractal properties of self-similarity and scale invariance. For example, to create the terrain for a flight simulator, it would be very impractical to generate all the terrain by hand; a typical map can cover an area of 10000 km^2. Modeling each individual convexity and concavity in the mountain to produce believable geometry would be very time consuming. Instead, through application of some simple fractal rules, it is possible to create infinite resolution mountains automatically, and online.

For this lab, you will be implementing the most widely recognized fractal – both in computer graphics and popular culture – the Mandelbrot fractal. The Mandelbrot fractal isn't useful in any practical sense (although it is aesthetically pleasing), but it does demonstrates that from a very simple procedural rule it is possible to create geometry that exhibits a remarkable degree of complexity.

Figure 1: A plant with irregular, self-similar, scale-invariant geometry. It would be very time consuming to generate the geometry of this model by hand.

2 Mandelbrot Fractal

The Mandelbrot set is defined as the set of all points in the complex plane which remain bounded after applying the following update rule to each point infinitely many times:

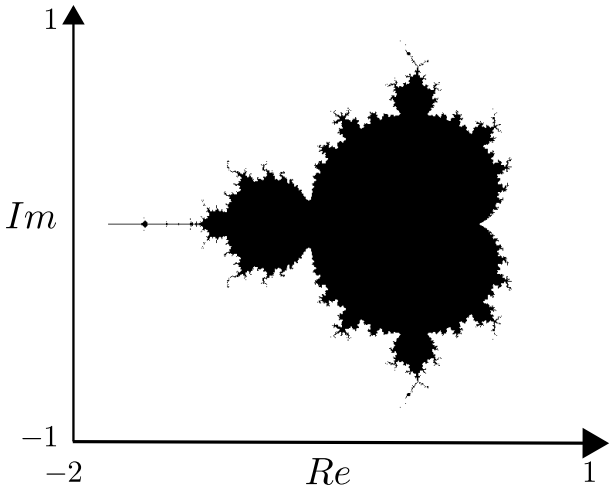


Figure 2: Visualization of the Mandelbrot set. Points colored black are within the set, and points which are white are outside the set.

$$z_1 = 0\,, \tag{1}$$
$$z_{i+1} = z_i^2 + c\,, \tag{2}$$

where z and c are complex numbers, and i denotes the iteration count. In other words, if a point does not diverge to infinity after infinitely many iterations of Eqn. 2, we say that point is within the Mandelbrot set.

As it turns out, the Mandelbrot set is fully contained within a closed disk of radius 2 about the origin. Any point which is outside of this closed disk can be proved to not be contained withing the Mandelbrot set. Therefore, if a point ever leaves the disk during some iteration, that point is known to diverge to infinity and we can cease iterating. If, on the other hand, the point does not exit the disk after a finite number of iterations, then all we know is that the point may be in the Mandelbrot set.

Since the Mandelbrot set is an infinite set, and we cannot iterate infinitely many times, we will approximate the set by iterating for some maximum number of iterations, n. The exact number of iterations will depend on how accurately we wish to visualize the boundary of the set, and is chosen by hand.

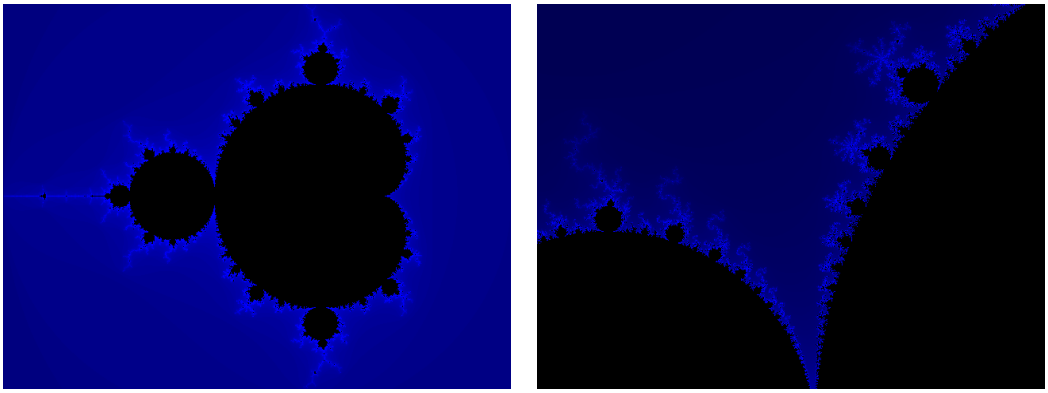
The Mandelbrot rendering algorithm proceeds as follows. For each pixel coordinate in our viewport, we will map it to the complex plane: this point is c in Eqn. 2. We will then iterate Eqn. 2 for a maximum of n iterations. If $|z_i| > 2$ after some iteration, i, we know that c is not within the Mandelbrot set, and exit early. If we exceed the maximum number of iterations, we consider c to be within the Mandelbrot set.

Fig. 2 is a visualization of the Mandelbrot set. We can zoom in to any arbitrary depth as see geometric substructures which are similar to the original zoomed-out fractal.

To make the Mandelbrot fractal more aesthetically pleasing, we can color the pixel as a function of i: the iteration count in which the corresponding point leaves the disk. The simplest coloring function is to scale some chosen color by i/n. This will increase the intensity of the color for points which diverge more slowly. Fig 3(a) shows the output of this coloring function with the color blue.

We can zoom into the fractal any number of times to generate more detail.[1] Fig. 3(b) shows a zoomed in portion of the fractal to the left.

[1]Technically, if we zoom in enough we will encounter floating point error and the output will be incorrect.



(a) The Madelbrot fractal Colored as a function of divergence iteration.

(b) A zoomed-in view of the same fractal to the left.

Figure 3: The colored Mandelbrot fractal