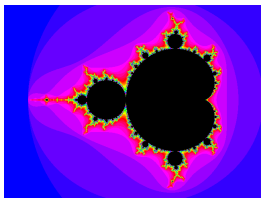
The MATLAB^{TM} Challenge is an effort to spur interest in programming among USI Engineering students. This challenge is avialable to all USI Engineering students.

Sierpinski's Gasket

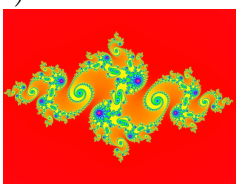
1 Introduction

Fractals are described as self-similar shapes; shapes that repeat when looking at any region of the image even if zoomed into different regions. In addition, unique to fractals are that they have non-integer dimension. The Mandelbrot and Julia sets are probably the most well known fractals. The "Barnsley fern" in an example of a fractal that may be observed nature and can be calculated using 4 separate coordinate transformations.

A) Mandelbrot Set



B) Julia Set



C) Barnsley Fern

Figure 1: Fractal Examples

A) http://teamlabor.inf.elte.hu/logosecsetvonasok/kepek/erdekes7/mandelbrot.gif
B) http://ubermari0.deviantart.com/art/Julia-Set-84023616
C) http://newtonexcelbach.wordpress.com/2010/10/03/faster-ferns/

2 The Challenge

Write a MATLAB^{TM} script to create Sierpinski's Gasket (including color) using one of the methods described below. Your code must allow for easy change of input variables to create various sized/shaped triangles.

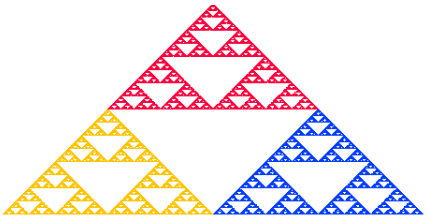


Figure 2: Sierpinski's Gasket plotted with red, yellow and blue triangles

2.1 Sierpinski's Gasket via Graphical Methods

One method for creating Sierpinski's Gasket is to:

1. Draw a solid triangle.
2. Find the mid-point of each side of the triangle.
3. Connect each of those midpoints.
4. Remove the "up-sidedown" triangle created by connecting each of the midpoints.
5. Repeat this process for each of the remaining solid triangles.

2.2 Sierpinski's Gasket - via "The Chaos Game"

The chaos game is played as follows: First pick three points at the vertices of a triangle (any triangle works–right, equilateral, isosceles, whatever). Color one of the vertices red, the second blue, and the third yellow.

Next, take a "six sided die" and color two of the faces red, two blue, and two yellow. Now select the coordinate of any point in the triangle. This point is the seed for the game. (Actually, the seed can be anywhere in the plane, even miles away from the triangle.) Then "roll" the die. Depending on what color comes up, move the seed half the distance to the appropriately colored vertex. That is, if red comes up, move the point half the distance to the red vertex and plot that coordinate point with a red dot. Now erase the original point (only this 1st point) and begin again, using the result of the previous roll as the seed for the next. That is, roll the die again and move the new point half the distance to the appropriately colored vertex, and then mark the point with the color indicated by the die.

A few words about the coloring here is in order. We have used color merely to indicate the proximity of the vertex with the given color. For example, the portion of the triangle closest to the yellow vertex is colored yellow, and so forth.

There is some terminology associated with the chaos game that is important. The sequence of points generated by the chaos game is called the orbit of the seed. The process of repeating the rolls of the die and tracing the resulting orbit is called iteration. Iteration is important in many areas of mathematics. In fact, the branch of mathematics known as discrete dynamical systems theory is the study of such iterative processes.

There are two remarkable facets of the chaos game. The first is the geometric intricacy of the resulting figure. The Sierpinski Gasket is one of the most basic types of geometric images known as fractals. The second is the fact that this figure results no matter what seed is used to begin the game: With probability one, the orbit of any seed eventually fills out the image. The words "with probability one" are important here. Obviously, if we always roll "red," the orbit will simply tend directly to the red vertex. Of course, we do not expect a fair die to yield the same two numbers at each roll.

Text adjusted from: "http://math.bu.edu/DYSYS/chaos-game/node1.html"

3 Challenge Rules

3.1 General Rules

1. Write your own code.
   (a) Your code must be original; All of your work must be your own!
   (b) You are encouraged to use subroutines where & when appropriate.
2. Please comment your code.
   (a) You must include your last and first name as the first commented line in your main code.
   (b) You must add comments to your code to describe the operation of different sections.
   (c) If your code is not understandable or easy to follow, you may forfeit your entry.
   (d) I strongly encourage you to include a flow chart with your submision.
3. Team entries are valid (all names must be inlcuded in submitted code), however, only one prize may be awarded.
4. Digital time stamps on submitted files will be used to determine order of entry.
5. "Elegance of code" will decide tie-breakers.
6. The MATLAB Challenge will be closed after the first successful entry.
7. Only one entry per team per month will be allowed.
   (a) There must be at least 30 calendar days between submissions.
   (b) You may re/submit as a team OR an individual - NOT BOTH.

3.2 Submission Instructions

1. Matlab code and required subroutines must be zipped into a single file.
2. Please use the file name structure: "LastName_FirstName_Year_Month_Day_MatlabChallenge01.zip"
   For example, my submission would be: "Davis_Julian_2014_09_07_MatlabChallenge01.zip"
3. E-mail the zipped file to Dr. Davis (jldavis2@usi.edu).
4. The E-mail subject line must include the text: "Matlab Challenge 01"

3.3 The Prize!

TBD.