COP 3502: Intro to CIS I **Spring 2014**

Problem Set 2: Fractal Art

Due: February 15th 2014

Last Updated: February 08th 2014

2.1 Introduction

> "*I find the ideas in the fractals, both as a body of knowledge and as a metaphor, an incredibly important way of looking at the world.*" Vice President and Nobel Laureate Al Gore, New York Times, Wednesday, June 21, 2000, discussing some of the "big think" questions that intrigue him.

Fractals are images which display "self-similar" patterns. What this means is that they are "the same from near as from far". No matter how far one "zooms in" to an image, similar patterns continue to emerge. Fractals can be found in nature in such diverse places as food and animal cells, water ripplings, lightning bolts, and snowflakes. They can also be more abstract as we saw with the Mandelbrot Set and Burning Ship fractals. Or, in the case of this assignment, the Sierpinski triangle.

2.2 The Assignment

First identified by the Polish mathematician Waclaw Sierpinski in 1915, the pattern resembles similar patterns dating back to Italy in the 13th century. Though the pattern may appear complex, it can be generated using a short recursive program. Your assignment is to create just such a program using the ideas and tools we've been using in class up until now.

For the Check

You will create a java program in a single file, `Sierpinksi.java`, that recursively generates the Sierpinski triangle. You will need a function `filledTriangle` that draws a single triangle to the screen in a certain location. Think about the recursive process. You will need a function `drawSierpinski` that for each triangle you draw, it draws three additional triangles on each of the edges. This process would generally continue forever, but we eventually want our programs to stop. Your program should take in a single command line argument, N, that determines how many times `drawSierpinski` should recurse. The goal of this assignment is to familiarize yourself with the drawing of shapes to the screen and where to place triangles and the recursive ability of function calls.

API: Application Programming Interface

As stated above, your `Sierpinski.java` program should have three primary functions:

- `filledTriangle(double x, double y, double len)`: Use `StdDraw.filledPolygon()` to generate a triangle whose bottom point whose bottom vertex is centered at (`x`, `y`) and has a side of length `len`.

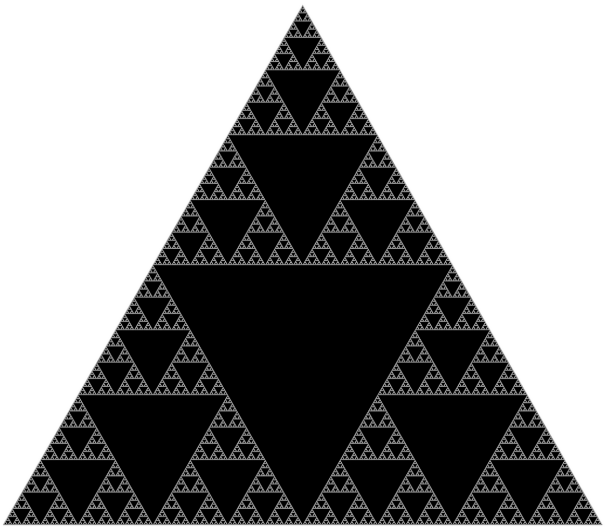


Figure 2.1: The Sierpinski Triangle

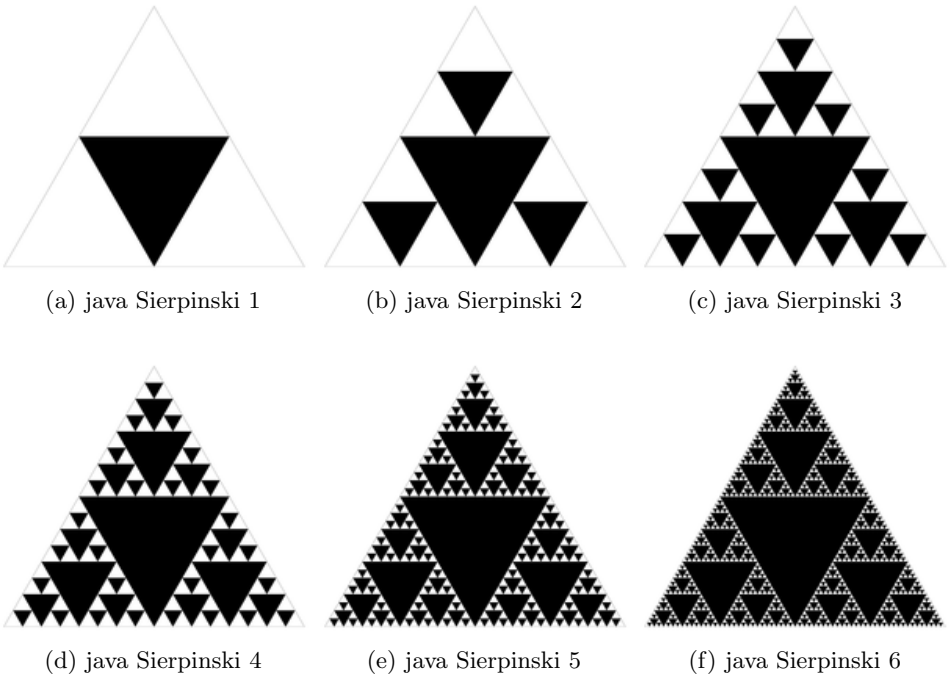
- `main(String[] args)`: You'll need a `main` function that takes in two command line arguments. The first is an integer specifiying the amount of recursion and the second is a double specifying the length ofyour initial triangle side.

- `drawSierpinski(double x, double y, double len, int n)`: This function should draw one triangle based on the first three input arguments and then recursively call itself three times to generate the next order of triangles.

Note: DO NOT change the size of the drawing window.

Above and Beyond

As in addition to your Sierpinski program, you will design and create your own fractal art and save it as `Art.java`. Your art should consist of objects that are drawn *recursively*. It should take in only a single command line argument N to determine the level of recursion in the same manner as your triangle above. It should stay within the predefined default drawing window at least for values of N between 1 and 7. Your art should not be something that could be easily replicated with simple iteration. You may use geometric objects of your own choosing (using `StdDraw.filledPolygon()` or even pictures (using `StdDraw.picture`). Your final picture may resemble a geometric pattern or something more random. Points will be awarded on the creativity and originality of your designs.



(a) java Sierpinski 1 (b) java Sierpinski 2 (c) java Sierpinski 3

(d) java Sierpinski 4 (e) java Sierpinski 5 (f) java Sierpinski 6

Figure 2.2: Example input/output values for your Sierpinski triangle program.

2.3 Where To Begin?

Drawing a Triangle to the Screen

Try to implement `filledTriangle()` by calling `StdDraw.filledPolygon()` using values specific to a triangle. The `filledPolygon` function takes in a collection of M (x, y) points, draws lines between those points, and fills in the resulting shape. The input to this function is an array of x values and an array of y values, where the first element of the x array is matched with the first element of the y array, and so on.

For example, if we want to plot a square of length 1 to the screen we need to provide the four points (0, 0), (0, 1), (1, 0), and (1, 1). We need an x array that contains the 4 x-values and a y array that contains the 4 y-values. As a reminder, arrays are *declared* using brackets after the variable and *assigned* by putting the values in curly braces. Our square would need the values:

```
double[] x = {0, 0, 1, 1};
double[] y = {0, 1, 0, 1};
```

Calling `StdDraw.filledPolyon(x, y)` with our arrays as input will plot a square to the screen. Your `filledTriangle()` function should set the arrays and perform the plotting for values appropriate to a triangle.

Recursion

Recursion occurs when a function calls it self on repeatedly smaller values of its input. The initial recursive order N is the command line input to your program. When using `drawSierpinski()` to plot your triangle, it should recursively call itself three times in the appropriate locations (you'll have to break out your geometry hats!) with a recursive order of N-1. Eventually, your recursion should reach the *base case* at which the recursion stops.

Your program should produce similar input/output results based on Figure 2.2.

Your Own Fractal Art

This part is designed for you to have fun. Try to run wild and come up with the weirdest, beautiful, scary, and/or creative patterns you can. As long as it follows the basic rules, nearly anything is allowed. For inspiration consult the following links:

- **Gallery of student submissions from another class**: http://www.cis.upenn.edu/~cis110/13sp/hw/hw03/gallery.shtml
- **Descriptions of Famous Fractal**: http://library.thinkquest.org/26242/full/fm/fm.html
- **List of Fractals by Hausdorff Dimension**: http://en.wikipedia.org/wiki/List_of_fractals_by_Hausdorff_dimension

For full credit on this part, your art must be recursive and it must be sufficiently different from the Sierpinski triangle. Just changing the triangle to a square, for example, will not receive credit. Remember that especially creative submissions can earn as much as a ++, which is equivalent to extra credit. Creative submissions include cool recursions as well as cool designs.

2.4 Submission Details

As for the previous problem set, all files will be submitted to E-learning. At a minimum, you can submit a single `Sierpinski.java` file with no README. **For grading purposes, your program MUST have this filename or you will lose points.** If you complete the "Above and Beyond", submit an additional `Art.java` that contains fractal art of your idea design. Again, your program must adhere to this naming convention. Additionally, you should include a README file containing a short description of your art and other thoughts that led to your inspiration.

In addition to the assignment due date of 2/15, `Art.java` submissions will be accepted until Wednesday 2/19. This extension is for those of you working on improving your original art to + and ++ levels. **You MUST submit a basic `Art.java` by the original 2/15 due date to receive any credit on the later submission**. This is to force you to start early and get practice writing recursive functions. The extra days should be used for improvement of your art, not the original creation. If you submit an improved version before the second due date, it will replace your original submission. We will be opening a second E-learning submission assignment for the second due date.

Your `Sierpinski.java` is still due on 2/15 or up to a day later with use of a late day. Your `Art.java` will not be accepted past the second due date.