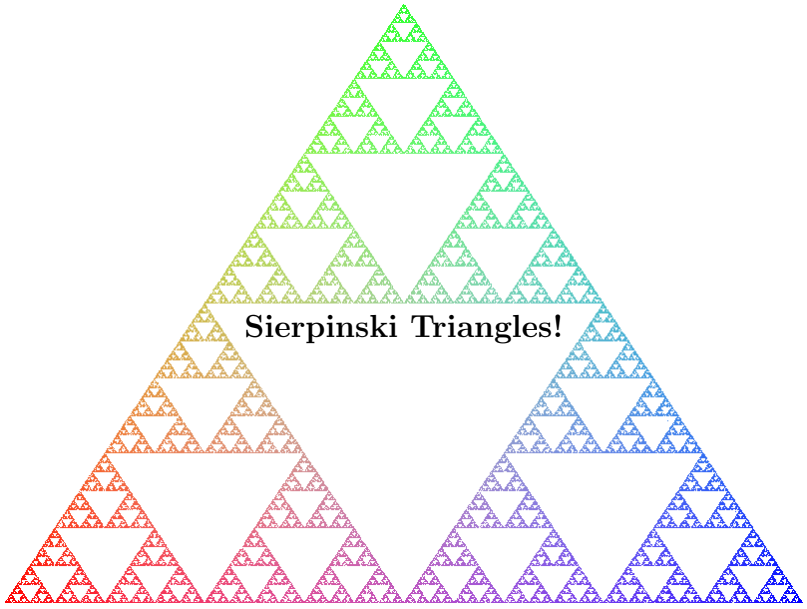

CSCI 141, Fall 2013, Project #1

Due date: Turn your program in to Canvas by Friday, November 8.

Problem description: The Sierpinski triangle is a self-similar, or fractal, figure that can be constructed in many ways. One way is the following algorithm, also called the **chaos game**:

1. Pick a random point inside the triangle.
2. Pick a random corner of the triangle.
3. Move the point half the distance to the corner.
4. Plot the point.
5. Go back to step 2 and repeat.

Obviously, you can repeat this as many times as you want, to get more refined pictures. Larger pictures will require more iterations.

Step One: Use the chaos game to write a program to produce a Sierpinski triangle in a **pygame** window. Use the `image.py` program I've provided on the website as a starting point, and the functions I've provided to create an image, show it, and save it.

At first, just make a black on white figure. The corners of the triangle should be calculated from the width and height of the window, giving the biggest possible triangle, as shown in examples at right. The first is from an 400×100 window and the second from a 100×400 window.

Step Two: Now colorize it by position! Here's what to do to get the figure above, with red, green, and blue corners blending through the interior.

1. Before the loop begins, for each corner, calculate the maximum distance a point can be from each corner. This will just be the maximum of the two distances to the other corners. Note that these two distances might be very different with a deformed triangle.
2. Now, inside the loop, calculate the distance of each point from each corner.
3. Divide each of these by the maximum possible distance from that corner. This gives you a number between 0 and 1 for each corner.
4. Scale each distance by 255 to get the red, green, and blue components of the color for that point.

Style: Use functions for every separate task. Finding where the corners are located, finding the initial point (make sure it's inside the triangle), finding the next point, finding the distance from a corner, finding the new color, *etc.*

Global variables should appear at the top of the program, and be documented with comments justifying their use as global variables. They might include things like: `screenWidth`, `screenHeight`, `numberOfIterations`, *etc.*

Don't forget to document the name of the program, author, class, purpose, *etc.* in a comment block at the start.

Set up the program so that when the TA runs the program, a colored Sierpinski triangle on a black background on a canvas of 400×300 will (gradually) appear.