COSC 231 **Program #4** **GUI for Fractals**

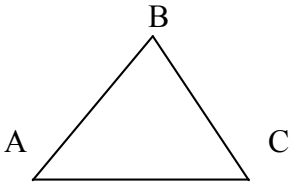
Distributed: 29 March 2011 **Due**: 12 April 2011

Write a GUI that will display, with certain user-controlled parameters, either the Sierpinski gasket or a Mandelbrot set.

Sierpinski triangle (gasket)
The Sierpinski triangle generation algorithm is given in Savitch's book (3rd edition) on pp 1061-2. That text is reproduced here.

Write a program that draws a Sierpinski gasket. A Sierpinski gasket or triangle is a type of fractal. It is an example of how an orderly structure can be created as a result of random, chaotic behavior.

The creation of a Sierpinski gasket is fairly simple. There are three points that form the corners of a triangle. In the figure below, they are labeled as A, B, and C.



To draw the Sierpinski gasket follow the algorithm:

1. Randomly pick one of the three corners. Call this the current point. (Alternatively, you could also randomly select any point inside the triangle).

2. Randomly pick one of the three corners. Call this the target point.

3. Calculate the midpoint between the current point and the target point.

4. Draw a single pixel at the midpoint.

5. Set the current point to the midpoint.

6. Go back to step 2 and repeat many times (e.g., 500 iterations).

It would seem like you should get a random mess of dots since the algorithm randomly picks a target corner each time. Instead, if this process is repeated many times, you will get a very orderly picture with a repeating structure: < picture from Savitch is not reproduced here>. Look at
http://en.wikipedia.org/wiki/Sierpinski_gasket >

To draw a single pixel at coordinate (x, y) use the `drawline` method where the start and endpoints are both (x, y).

Mandolbrot set
There are several explanations of the Mandelbrot set. A simple one, including the iterative algorithm for determining if a number is a member of the set, is given at http://www.ddewey.net/mandelbrot/
Here is another simple explanation:
http://t16web.lanl.gov/Kawano/gnuplot/fractal/mandelbrot-e.html

Each point, C, in the complex plane is a member of the Mandelbrot set if after iterating (forever), the magnitude of Z (Z = a + i*b, magnitude(Z) = sqrt(a^2 + b^2) is less than 2.

That is, each point C has its own Z value.

Z iterates as follows:

$Z_0 = 0.0 + i * 0.0$

$Z_{n+1} = Z_n * Z_n + C = (a_n + i*b_n) * (a_n + i*b_n) + (x + i*y)$

Represent a point C = x + iy on the drawing surface as (x, y).

Khan academy on youtube has a nice and sufficient explanation for complex numbers that will take a little over 20 minutes to view.

http://www.youtube.com/watch?v=kpywdu1afas Complex numbers (part 1)
http://www.youtube.com/watch?v=EQviquyrDxA&feature=related Complex numbers (part 3
http://www.youtube.com/watch?v=dWjg6fiKNOw&feature=related Complex numbers (part 11)

/***/

So, in pseudo-code:

```
// initialize z
for (int row=0; row <=WIDTH; row++)
   for (int col = 0; col <= HEIGHT; col++) {
      (z[row][col]).real = 0;
      (z[row][col]).img = 0;
      }

// iterate for long enough
for (int t = 0; t < MAX_ITERATION; t++) {
   for (int row=0; row <=WIDTH; row++)
   for (int col = 0; col <= HEIGHT; col++) {

      compute real and imaginary values of z for point (row, col);

      if ( sqrt (
                  ( (z[row][col]).real ^2) +
                  ( (z[row][col]).img  ^2) )
                ) < 2.0) color (row,col) INSET;
      else color (row, col) OUTSET;

      }   // for int col
   }      // for int t
```

The GUI interface will have a drawing surface and the following buttons:

1. Clear – to clear the drawing surface.

2. Set Points – activate the mouse listener to select the bounding box for the Sierpinski Triangle or for the Mandelbrot set.

3. Start Sierpinski – Draws the Sierpinski triangle at the current speed and color option and starting with the current target point and current point. Fractal generation will reset when the user hits this button. A reset means the drawing surface is cleared and values for fractal generation are (re)initialized.

4. Start Mandelbrot - Draws the Mandelbrot set at current color option. Fractal generation will be reset when the user hits this button.

5. Stop – Halts the current graphic generation. The current state of the fractal generation is saved so that if the user subsequently hits the Start button, the fractal generation continues from the current state. Do NOT clear the drawing surface.

6. Speed – Lets the user choose the speed of the iteration (e.g., the length of the time to wait before returning to step 2. The speed can be changed during fractal generation. Fractal generation will not cease or reset. *Use a slider to control speed.*

7. Color choice – A user can select a colors for the fractal. Use a ColorPicker to actually select colors. The color choices are placed in an array.

If the user is drawing a Sierpinski triangle, the color of the generated point will change every 100 pixels, cycling through the color array.

If the user is drawing a Mandelbrot set, the colors indicate the number of iterations to determine that a number (x,y) is not in the Mandelbrot set. Points that are in the Mandelbrot set are given the value BLACK. Initially, all numbers are in the Mandelbrot set.

The color choices cannot be altered during fractal generation – make sure that control is deactivated!

The buttons must be arranged in a rational and user-friendly way. Nicely aligned. Related functionalities grouped together in geography and in style.

Simplifications:

- You can assume the user will not change the size of the GUI during execution.

Options:

- You can pick other fractals to implement rather than Sierpinski triangle and Mandelbrot set. One fractal should be discrete in the sense that Sierpinski triangle is (not all points on the plane are marked), the other is continuous in the sense that Mandelbrot set is (every point on the plane is in the Mandelbrot or it is not). Possibilities to consider are: (replacing Sierpinski triangle) dragon curve, Koch snowflake, Sierpinski carpet, and (replacing Mandelbrot set) Julia set, Burning Ship, Nova fractal.

Deliverables:

- URL
- Hardcopy of GUI and fractal code
- Demo and code walk-thru

Grade based on:

- Satisfying the program specs given here	75%
- GUI layout	5%
- Elegance of code:	10%
- Coding standards	10%