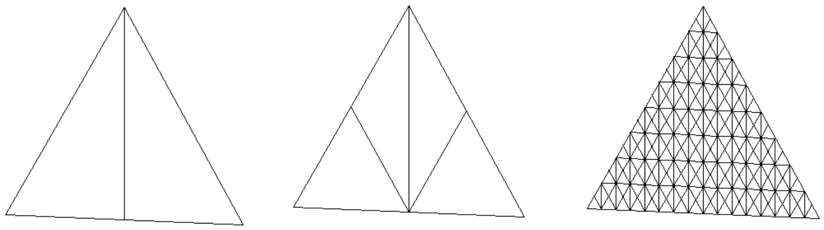
Laboratory 7: Recursive Doodlings

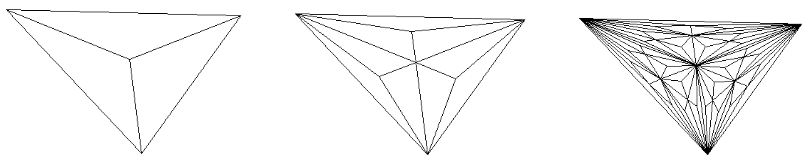
Objective. To experiment with recursion using graphics.

Discussion. We are all used to drawing silly "doodles" on napkins and scraps of paper. The purpose of this assignment is to construct computerized doodles in the drawing window. We will focus on triangle-shaped doodles that are constructed in a variety of ways.

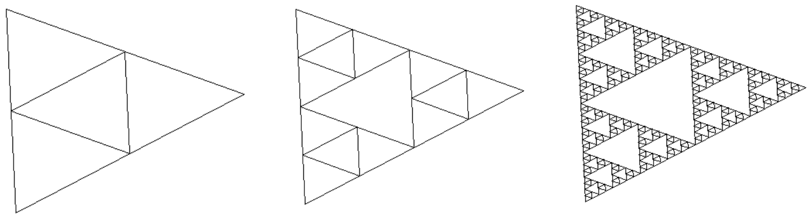
Our first technique is to draw a triangle and then connect one "top" vertex to the midpoint on the opposite side. The result is two triangles that can be further subdivided. When each of these triangles is halved, the newest vertex (the one that was once a midpoint) is considered the top of each. The doodling continues, recursively, until the distance from the top to the midpoint becomes shorter than, say, 10 pixels. Here is the progression of the figures drawn with this technique:



The second technique divides the triangle into three subtriangles by connecting the midpoint of each side to the *centroid* (a point that is the coordinate-wise average of the three vertices). If the centroid is far from each of the vertices, each of the three triangles is further subdivided. Here is the progression of the figures drawn with this technique:



Our final technique divides the triangle into four subtriangles by connecting each of the three midpoints. The three corner triangles are further subdivided if each of their sides is sufficiently large. This pretty doodle has the following progression:



Procedure. Follow these steps to complete at least two of the three doodle methods:

1. Download the file `lab/doodle.java` from the book's web site. Peruse this file. It contains working code for determining the distance between two points, the point halfway between two others, and a utility method for reading three vertices from the drawing window. Computing the centroid of three points is similar to computing the midpoint of two.
2. Pick one of the three doodle technique, and write a recursive method, `doodle`, that accepts three points and draws a doodle within that triangle. It is useful to declare a constant for the minimum length side that may be easily changed.

   Make sure that your recursive method has all the features we might expect: a base case, self reference, and progress.
3. Test your program with triangles of various sizes.
4. Complete another program that implements one of the other doodle forms.

Thought questions. Consider the following questions as you complete the lab:

1. What is "progress" in these recursive methods?
2. Suppose you change the color of the pen before each recursive call within one of your doodles. Can you predict the color pattern?
3. What happens if you change the order of the actual parameters in the recursive calls? Which of the black-and-white doodles would be different?