**Tendron**

In this lab, you will use recursion to generate patterns / images. Your code will eventually draw a "plant" recursively, modifying the structure as the elements get smaller, going through smaller branches and finally to leaves. Every time the Tendron program runs, it generates a new and different plant since growth occurs randomly.

The main program, **Tendron.java**, sets up the graphics world for the image. When asked to paint, it creates a new Cluster, passing it the graphics context (the g parameter, essentially the paintbrush) to draw with. Then it calls the display method of the Cluster class, telling it to draw a cluster starting at the midpoint of the screen using n short segments.

A cluster is a group of seven tendrils. Each tendril is made of a series of short line segments, and as each segment is drawn, it may turn a bit left or right, giving a more life-like appearance.

At the end of each tendril, a new cluster is created. The new cluster will have fewer line segments, therefore each tendril will be shorter than before. Recursively, these tendrils will again spawn clusters until a minimum tendril length is reached. Below is the final result:



When drawn, the tendrils together look very plant-like for such a simple algorithm. Notice the effect is enhanced by using the tendril length to determine the line color used when drawing.

**The Assignment**

You will complete the Cluster and Tendril classes.

The Cluster class has a constructor that stores a reference to the Graphics context to be used later (for drawing), and one public method, display. The display method creates seven "tendrils" by creating Tendril objects and then invoking *their* display methods.

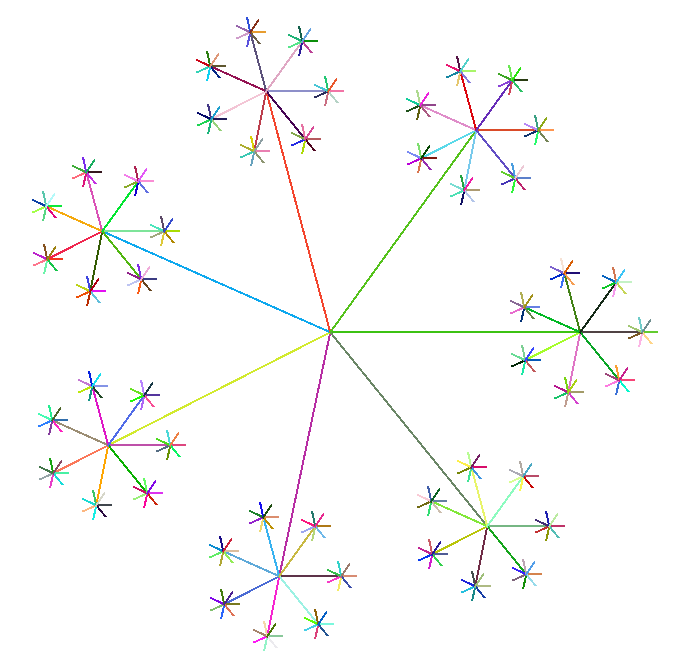
The Tendril class also has a constructor that stores a reference to the Graphics context passed to it by Cluster. This class does the actual drawing through its only method, display. The display method draws the tendril using the drawLine(int, int, int, int) method of the Graphics class. The drawLine method and the setColor method are the only methods you need to draw your image.

For example, if you were passed the graphics context in your constructor and saved it as Graphics g, you could draw a red line from the point (10, 15) to the point (30, 75) with these two lines of code:

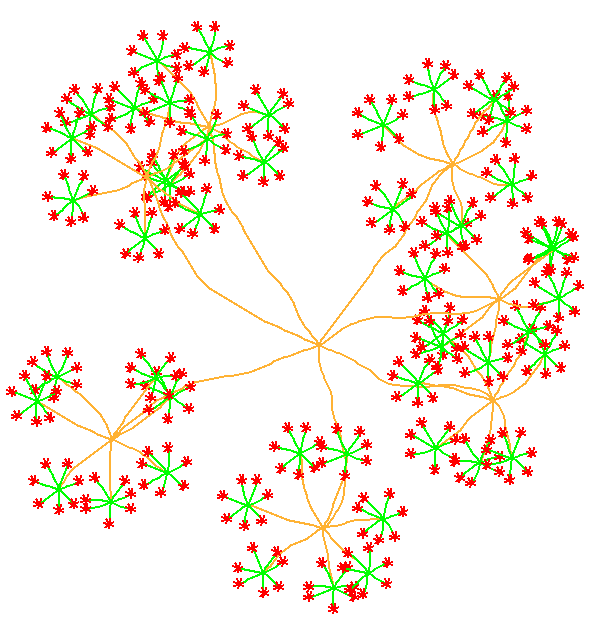
g.setColor(Color.RED);

g.drawLine(10, 15, 30, 75);

Before attempting to code the plant-like display method, it may be helpful to start with a pinwheel method that simply draws straight lines, so that you understand the recursive elements before worrying about the slightly more complicated math required to make lines move randomly. Example of pinwheel method call, using a random color for each tendril:

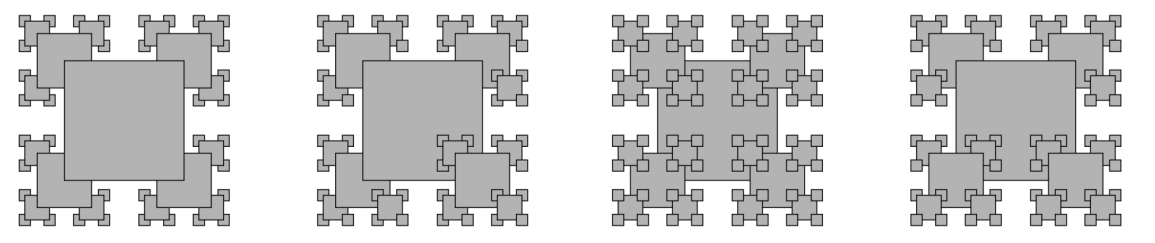


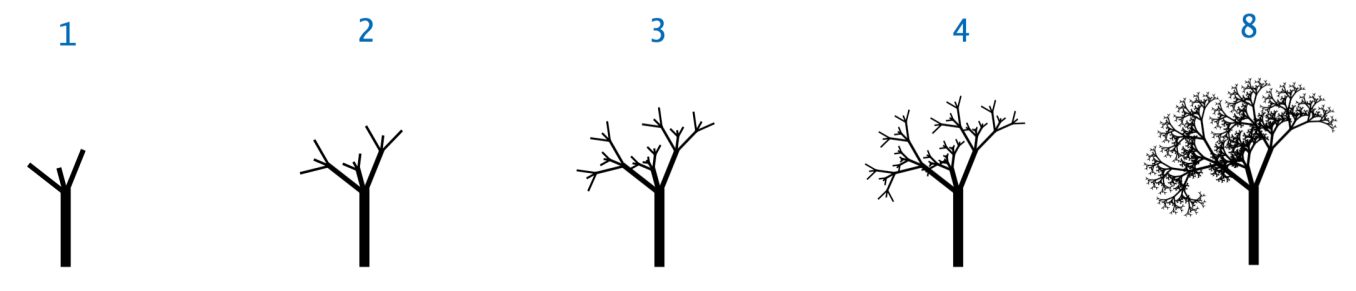
When drawing the "actual" tendril, remember to randomly change the direction (yes, you will need math for this) and to use an appropriate color for a life-like appearance. After each tendril is drawn, conditionally create a new, smaller cluster on the end, which will have its own seven tendrils, each of which may have a cluster at the end... you should end up with something like this:

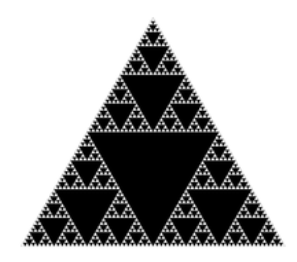
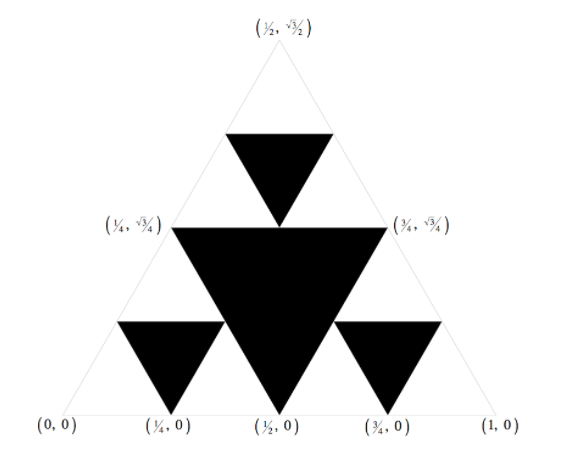


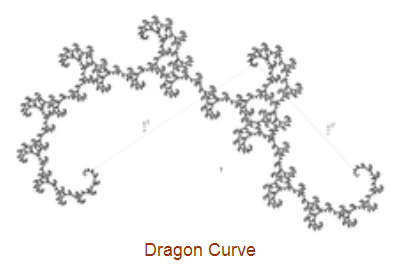
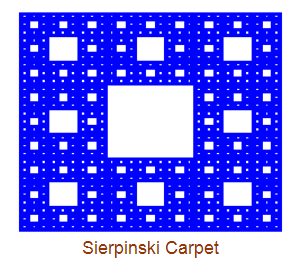
**(Optional) Other recursive graphics**

Simple recursive drawing code can lead to amazingly intricate graphics. Try the following challenges, and feel free to experiment with your own!

Types of recursive squares:

Recursive trees:

Sierpinski's triangle: 



Based on the Tendron lab

*http://www.rfrank.net/cslabs/1640-tendron/1640.html*