# RECURSIVE SHAPES

***Not Quite Fractals, But Getting Close***

## CSC211 LAB 11

---

## Lab 11 Background

This lab uses the vast amount of knowledge you've gained this week on recursive methods!

You are required to create two class that meet the following requirements:

- use recursion to draw one of the five project classes below
- implement the **`FractalInterface`**. This interface contains a single method, **`drawFractal`**, that should call the recursive method that draws the shape.
- The constructor should initialize
- the position to draw the main shape (**`xCorner`**, **`yCorner`**)
- the length of the **`side`** (rectangles) or **`radius`** (circles)
- the **`shapeColor`** to draw the shape (can be random)

Each recursive method needs to have as parameters

- the location (x, y) of where to draw the object. Use two integers or a **`Point`** object.
- the current length of a side (for rectangles) or radius (for squares)
- the **`Graphics`** object (pen)
- the number of recursive levels

Each recursive call should

- draw the basic shape using the parameters supplied
- decrease the side or radius
- decrement the recursive level
- make additional recursive call(s) with the new values

The base case should be when the recursive level is 1. When you reach the base case, draw the simplest form of the shape. You can also do as is done in the demo and have a second base case with a minimum size for the side or radius.

Since none of these methods returns a value, they can all have an implicit base case.

There is a demo of the lab that shows how to implement a sample class, **`SierpinskiCarpet`**, that recursively draws the shapes.

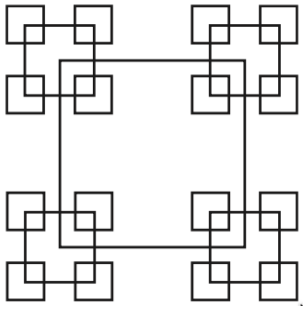
Once you have implemented your two shapes, change lines 20 & 21 in **`Lab11DemoPanel`** so they instantiate an instance of each of your objects. You should not need to make any other changes to **`CSC211Lab11Recursion`** and **`Lab11DemoPanel`**.

Test your recursive algorithms by setting the recursive levels to 1, 2 and then 3 to see how the shape is drawn and ensure you are correctly implementing the recursive calls.

## Project 1: QuadSquare Class

### Recursive Method: drawQuads

This class should draw the following recursive design:



Each call should:

- Draw a square.
- Then recursively draw a smaller square at and centered on each corner. (This requires four recursive calls).

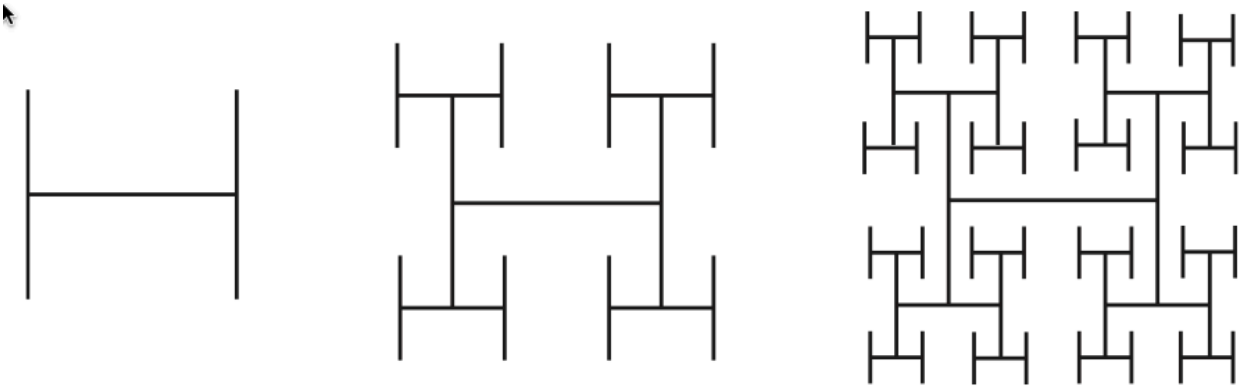
Recursively repeat drawing smaller squares at the corners of squares until the maximum number of recursive levels is reached.

***Base case***: draw a single square.

## Project 2 - HTree Class

### Recursive Method: drawHTrees

An *H tree* of order *n* is a recursive design based on the letter H. The following are H trees of orders 0, 1, and 2:



Each H is centered at a given point and consists of three line segments of equal length.

Each H has a half-sized H centered on each of its four tips. (This requires four recursive calls).

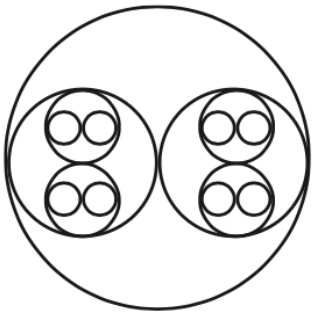
Recursively repeat drawing smaller H's at the tips until the maximum number of recursive levels is reached.

***Base case***: draw a single H.

## Project 3 - CircleEyes Class

### Recursive Method: drawCircleEyes

This class should draw the following recursive design:



Each call should:

- Begin with a circle,
- Inside that circle inscribe two circles whose diameters are one-half of the given circle's diameter and are tangent to both the larger circle and to each other. An imaginary line joining the centers of the three circles is horizontal.

You recursively repeat this process on each circle of the two inner circles.

At each successive level of the recursion, the imaginary line joining the inscribed circle centers alternates between horizontal and vertical so you'll need an additional parameter to track the lines direction.

***Base case***: draw a circle with two inscribed inside.

## Project 4 - GoldenRect Class

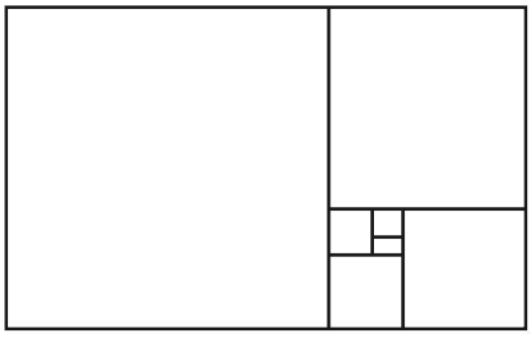
### Recursive Method: drawGoldenRects

A rectangle whose short side has length ***s*** is called a **golden rectangle** if the length of its long side is:

```
s * (1 + Math.sqrt(5) ) / 2.0
```

If you divide the rectangle into two pieces by drawing a line of length ***s*** to form an ***s***-by-***s*** square, the remaining piece will be a golden rectangle.

Recursively subdivide a golden rectangle to form a design such as the following one:



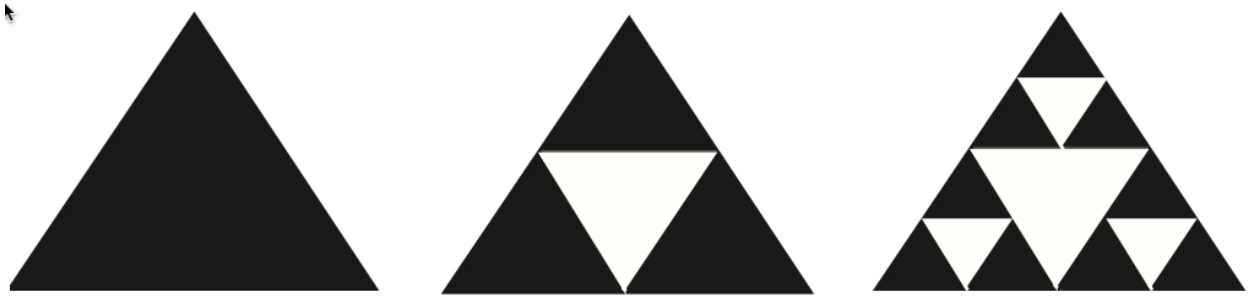
***Base case***: draw rectangle is a line dividing it into a square and rectangle.

**Base Case**

## Project 5 - SierpinskiTriangle Class

### Recursive Method: draw SierpinskiTris

A Sierpinski triangle is analogous to a Sierpinski carpet. To create one, you begin with an equilateral triangle. Connect the midpoints of the sides of the triangle to form four subtriangles, and remove the inner subtriangle. The result is a Level 1 Sierpinski triangle. Repeat the process on each of the remaining three subtriangles to get a Level 2 Sierpinski triangle, as follows:



Level 1 Level 2

Theoretically, you could continue up to any level, but in practice you will reach a point where you will not be able to see additional triangles.

Each call should:

- draw an equilateral triangle.
- connect the midpoints of the sides of the triangle and remove the inner subtriangle by drawing in an alternate color.
- recurse with each of the other three triangles

***Base case***: draw a level one triangle