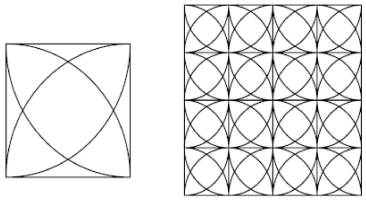
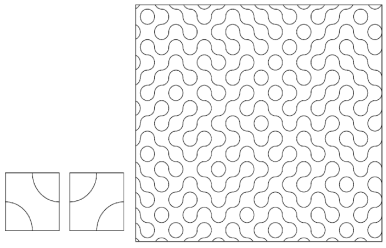
CS 371 – Assignment 1
Due: 11:59pm, Friday, September 12th

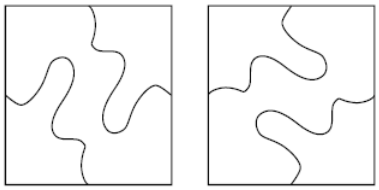
This assignment will require that you thoroughly understand viewports and parameterized curves. Computer graphics offers a powerful tool for creating pleasing pictures based on geometric objects. One of the most intriguing types of pictures is called a *tiling* – a picture composed of smaller objects that apparently repeat "forever" in all directions. For example, the figure below shows a basic tiling. A *motif*, in this case four quarter circles in a simple arrangement, is designed in a square region of the world. To draw a tiling over the plane based on this motif, a collection of viewports is created side-by-side that covers the display surface, and the motif is drawn once in each viewport.



On this assignment, you will write a program that allows me to select (using a command-line parameter as specified below) between two tilings. The first such tiling should be derived from the motif with four quarter circles illustrated above. The second tiling should be derived from the so-called *Truchet tiles* pictured below.



Notice that there are (at least) two different Truchet tiles to choose from. In this second tiling, your algorithm, as it walks around drawing a motif in a viewport, should choose at random from a pool of at least two Truchet tiles. Truchet tiles are not limited to being composed of circular arcs. For example, the two Truchet tiles pictured below will produce an even more interesting tiling.



The only requirement is that the curves that comprise an individual tile meet the edges of the tile at the midpoint. However, you should be forewarned, tilings look much better when the angle with which each curve meets the edge is chosen to avoid sharp corners in the resulting curve. Hence, you can get some golly-gee-whiz points by working hard at getting some more interesting Truchet tiles, but you can lose points if the visual appeal of your graphic is reduced by sharp corners. As you ponder how to produce these motifs, parametrized curves should come to mind. For instance, the motifs in the first two figures above are just circular arcs. Other more exotic Truchet motifs can also be created using parametrized curves.

Your program should allow the person running it to choose between at least the two motifs described above, that is, plain-vanilla circular arcs and some variation on randomized Truchet tiles. When you display the motif and its tiling, do it in a fashion that resembles the first two figures above. That is, display a large version of each motif on the left and then tiled smaller versions of the motifs on the right. If one runs your program from the command line, this choice of motifs should be established through a command line parameter as follows:

```
java -cp  /shared/naps/jogamp/jar/jogl-all.jar:/shared/naps/jogamp/jar/gluegen-rt.jar:. MyTiles 1

                                        OR

java -cp  /shared/naps/jogamp/jar/jogl-all.jar:/shared/naps/jogamp/jar/gluegen-rt.jar:. MyTiles 2
```

where *MyTiles* is the name of your main program class, and the command line value 1 or 2 dictates whether tiling 1 or 2 is used.

Your source code file *must be named* MyTiles.java. You should submit that one file and only that file to the "Assignment 1 Submissions" dropbox on D2L. If your source code file is named differently or if you submit multiple files to the dropbox, 10 (out of 100) points will be deducted from your score. On the campus Linux network, I must be able to compile your program using the command:

```
javac -cp  /shared/naps/jogamp/jar/jogl-all.jar:/shared/naps/jogamp/jar/gluegen-rt.jar:. MyTiles.java
```

If you develop the program on your own computer, you should still be sure to test it with this compile command on the campus Linux network. Even if you can compile the program on your own computer, you will automatically lose big points if this Linux compile command doesn't work.

I will print a copy and skim the source code. When I do that, there are some simple stylistic issues I will be checking.

Guidelines for the evaluation of your program

- Each motif should be developed in a separate method (just like I developed the Genie curve in a separate method in the demonstration program in class). Of course, you are strongly encouraged to modularize your program using other methods, such as circle-drawing routines.
- Your program must have an substantial introductory documentation block. By reading that I should learn:
  - Any options you have that might qualify for "golly-gee-whiz" points. Be sure you tell what what you're doing above and beyond "plain vanilla" and how I should run the program to access those options. If you fail to document these "golly-gee-whiz" features, you won't receive credit for them.
  - The math behind each of the parametrized curves you've used in drawing a motif. In particular describe how you broke the motif down into smaller more manageable parametrized curves.
  - Any internet sources from which you may have cut-and-pasted some code into your program. Remember that it is OK to do this as long as you cite the source.
- What other documentation should you provide? Each method in your program should have an introductory documentation block that clearly and accurately describes its role in the program
- Make sure that your program's internal logic is self-documenting through the judicious use of *meaningful* variable names and indentation. Use underscores or a mix of upper and lower case to achieve this self-documenting style, e.g., *sweep_angle* or *sweepAngle* are far superior variable names that *sweepangle* or the truly despicable single-character name *s*.
- Indent your code correctly – Use *http://www.prettyprinter.de* if it's not easy to do in your editor.
- In effect, what I'm asking you to do with these stylistic issues isn't anything beyond saying that your code must look and read as good as my sample programs that have appeared on the lecture slides. If you fail to do that, points will be deducted even if your program works correctly.

That's all – have fun!