CS106 W20 - Assignment 05

Advanced Shapes

Due: Friday, 7th February, 11:59 PM

There is no starter code for this assignment.

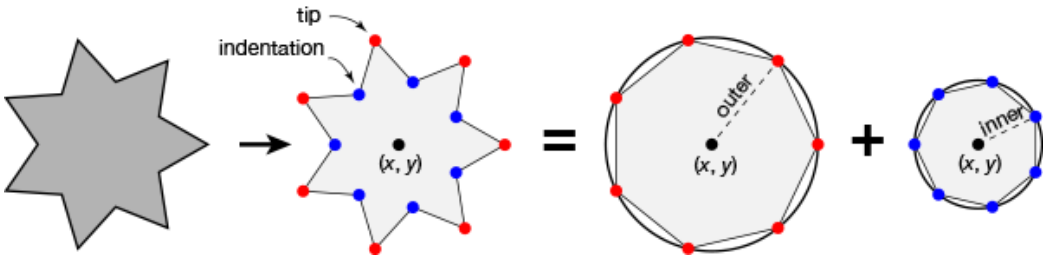
Question 1: Stars

A *star polygon* is a shape with *n* tips that stick out, alternating with *n* indentations. The shape's edges are all the same length, and the tips and indentations are all evenly spaced.

We can describe a star polygon using exactly five numbers:

- *x*, *y*: The coordinates of the centre of the star
- *outer*, *inner*: the distances from the centre of the star to every tip (*outer*) and every indentation (*inner*)
- *n*: the number of tips the star has (which is the same as the number of indentations!)

The points of a star are actually pretty easy to describe. In a star with *n* tips, those tips are the corners of a regular polygon with *n* sides. The indentations are another regular polygon with *n* sides, rotated by 180.0/*n* degrees relative to the first one. When drawing the star, you just alternate between these two families of points. The diagram below breaks things down a bit for you, using the example of a 7-pointed star.



Write a sketch called Stars that reads a list of stars from a text file called stars.txt in the sketch's data/directory, and draws each one to the sketch window. Once the stars are drawn, the sketch doesn't need to do anything else. Each line of the input file describes a single star by giving the numbers *x*, *y*, *outer*, *inner*, and *n* in order, separated by spaces. For example, if stars.txt looks like this:

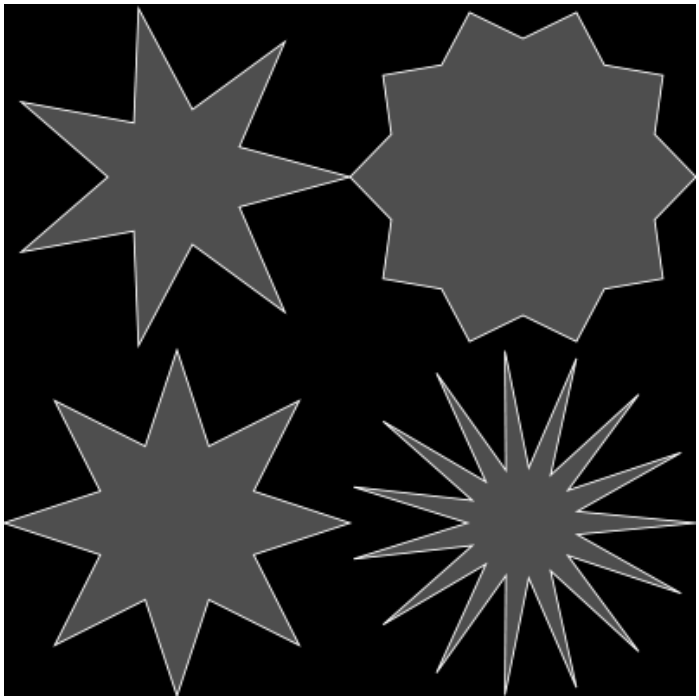
```
125 125  125 50   7

375 125  125 100  10

125 375  125 60   8

375 375  125 40   15
```

Then the sketch window might look like this:



Your sketch must satisfy the following requirements:

- The sketch window is of size 500×500.
- When the sketch starts, it reads the file stars.txt and draws every star described in that file.
- The stars are drawn using beginShape(), vertex(), and endShape(). Every star has both a fill colour and an outline colour.

Here are a few additional tips that will probably help you:

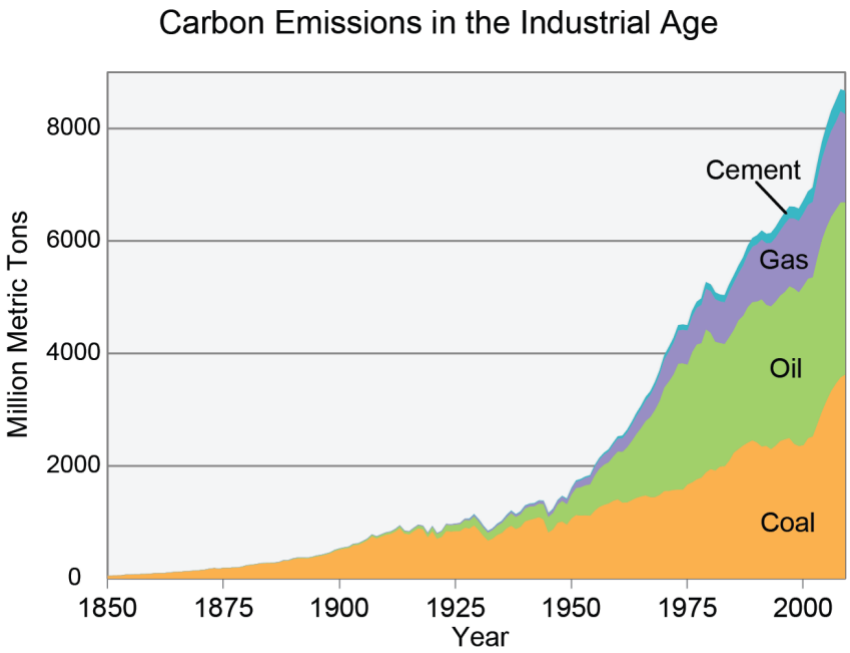
- The first tip of the star should always point to the right. Note that there's nothing in the description above that forces *outer* to be larger than *inner*. That's by design—if *outer* is smaller, then the tips and indentations simply trade places, which is fine.

- You are encouraged, but not required, to write a helper function that takes five numbers as arguments, as described above, and draws a star based on those five numbers.
- You will certainly want to use the polar() function from the lecture notes.

To test your sketch, we provide a zip file containing the simple test above, as well as six more complicated patterns of stars. Download the file star_tests.zip and unzip it into your sketch's data/ directory. You can temporarily rename different files to stars.txt, or change the sketch to read stars_test1.txt, etc. Have a look at star_tests.pdf to see whether your sketch is drawing the stars correctly.

Question 2: Graph

The website globalchange.gov collects data and reports on climate change with the participation of a number of US government agencies (at least for now...). A recent report includes this graph, showing carbon emissions of various kinds dating back to the industrial revolution:



Globalchange.gov makes its data available along with the reports, meaning that we can examine the numbers and reproduce the graph if we want. In this question, you will draw one part of the graph (total emissions, the topmost curve) based on their dataset.

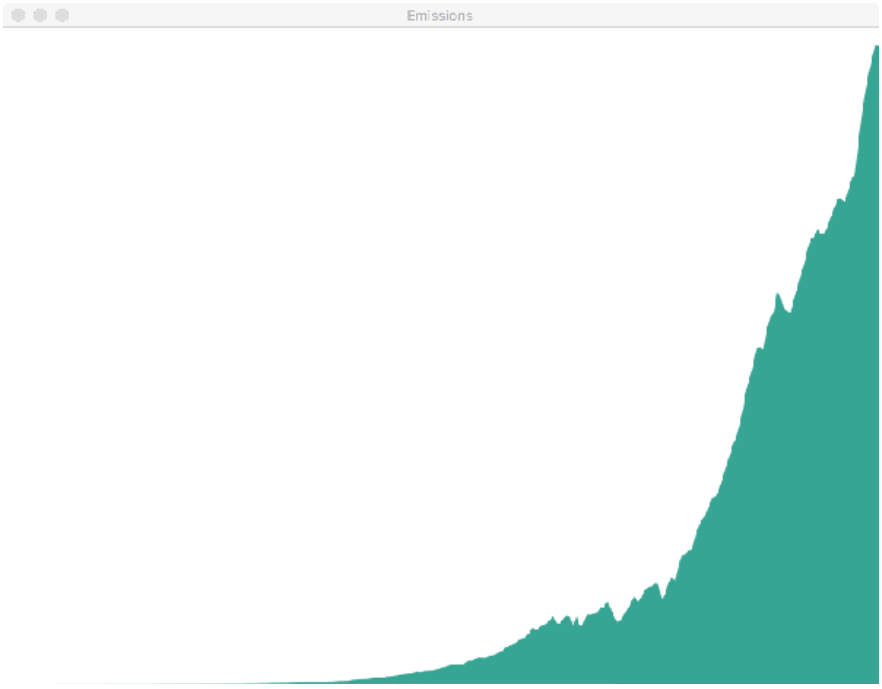
1. To begin, create a new sketch called Emissions, with size 800×600.

2. Download their dataset (if that link doesn't work, grab this local copy). This file contains a preamble with some information that we don't want to deal with in code. In a simple text editor (e.g., notepad on Windows, TextEdit on Mac), strip off everything above the actual data, so that the first line of the

file starts with "1751". Save the result, making sure it's still a plain text file, and store it as emissions.txt in your sketch's data/ folder. Do not modify the file in any other way.

3. Now write a sketch that reads emissions.txt and draws a graph of the total emissions from 1751 until 2009.

    I. Read every line of the file, and extract the number in the second column of that line (i.e., the number right after the year).
    II. Use the function map() to turn the year (or, equivalently, the line number in the file) into an *x* position.
    III. Use map() again to turn the total emissions into a *y* position, where the bottom of the sketch corresponds to 0 and the top corresponds to 9000.
    IV. That *x*,*y* combination gives you a point on the curve you want to plot.
    V. Use beginShape(), vertex(), and endShape() to draw all the points on the curve.
    VI. Add one last vertex in the bottom-right corner of the sketch window, so that the shape of the graph can be drawn with a solid fill colour. Draw the graph with a solid fill colour and no outline colour, on a white background:



Those are the only requirements. They lead to a very simple drawing, and the sketch can be written in less than 20 lines of code. You are welcome to experiment with extensions, as long as the core shape above is clearly visible in the sketch. Consider adding a title, names for the axes, labels along both axes, and so on. You can try to draw all four graphs as in the picture above, though you may encounter a few subtleties that make it tricky. We may award bonus marks to especially ambitious or informative enhancements.

Submission

When you are ready to submit, please follow these steps.

1. Please ensure that any sketches you submit compile and run. It's better to submit a sketch that runs smoothly but implements fewer required features than one that has broken code for all features. If you get partway into a feature but can't make it work, comment it out so that the sketch works correctly without it.
2. If necessary, review the Code Style Guide and use Processing's built-in auto format tool. You do not need to use the precise coding style outlined in the guide, but whatever style you use, your code must be clear, concise, consistent, and commented.
3. If necessary, review the How To Submit document for a reminder on how to submit to LEARN.
4. Make sure to include a comment at the top of all source files containing your name and student ID number.
5. Create a zip file called A05.zip containing the entire A05 with its subfolders Stars and Emissions.
6. Upload A05.zip to LEARN. Remember that you can (and should!) submit as many times as you like. That way, if there's a catastrophe, you and the course staff will still have access to a recent version of your code.