CMSC427 Fall 2017
Lab 1 – Validating polylines

Due by start of class Tuesday, Sept. 19th
Online as combined PDF plus a PDE file

Objectives of lab:

- Work with event driven programming
- Use vectors in polygon algorithms
- Be able to draw polylines for use in the class

Requirements:

Starting with the PolylineEditor code posted to the class web page (under the second day resources), modify it to meet these requirements.

1. Enable the user to delete a point in one of two ways:
   a. Hit the 'd' key while dragging a point
   b. Hit the 'd' key to delete the last point added if no point is picked
2. Enable the user to save or retrieve the polyline from a file, using a basic file format of your own design (such as #points on first line, and each point (x,y) on following lines). You can use a fixed file name "polyline.dat" rather than wrestle with Processing on string input.
3. Add to the Polyline class three Boolean methods.
   a. *isSimple*() that returns true if the polyline represents a simple polygon, ie, no intersections or duplicate points.
   b. *isConvex* (polyline p) that returns true if the polyline represents a convex polygon.
   c. *isCW*(polyline p) that returns true if the polyline represents a polygon that winds clockwise.

For 3a-c you should add key events that test the methods by calling them and outputting the results. You can output the result in the Processing console window or use another output mode that you'd like. To output in the console window this works: `println("isSimple: "+polyline.isSimple());`

For 3b and 3c you may assume that the polygon is simple (you have isSimple to determine this).

For 3c, if needed you may assume the polygon is convex (you have isConvex to determine this.) Assuming it is convex enables certain approaches. However, for a complete answer, you should be able to handle the case when the polygon is not convex, and has concave vertices.

Processing has the PVector type which supports dot product, cross product and other operations.

For 3a isSimple you may use a brute force approach to finding line intersections, although you are welcome to use a faster approach. We'll post a short note on finding line intersections on the class web site along with this assignment. You may also use a small real number fuzz factor as you'd like (are two points the same within the fuzz factor, or do two lines intersect within the factor.)

Since you may find analysis and code for these methods online, part of the assignment is explaining what you've done in your own words. It's ok to look for analysis online – not ok to copy code. For your own learning, it's good to start with your own analysis of the problems.

To submit you should create a Word or other document that includes the following:

A. A header with CMSC427 fall 2017 Lab 1 and your name.
B. A short narrative with a description of what you've done – what formula, what modifications you made to the original PolylineEditor code, your approach.
C. A copy of the PDE source code from Processing copy and pasted (hint – "copy as HTML" brings along the color coding").
D. A copy of your final image copied and pasted into the document.
E. Save as PDF and submit, along with a separate copy of your PDE files.

As a lab, the requirements for code are lightweight in that you don't have to validate your program against all possible inputs, or work on the most general solution. Consider your program a working prototype. You're free to extend, play with, revise, and otherwise make the assignment yours.