22C:31 Programming Project

Due via ICON by midnight, Friday, December 11th

A pair of line segments AB and CD in the plane are said to *cross* if AB and CD have a common point that is not one of the 4 endpoints A, B, C, or D. Let P be a set of points in the plane. A *triangulation* T of P is a maximal set of pairwise non-crossing line segments connecting pairs of points in P. Here "maximal" refers to the fact that it is not possible to add a line segment connecting a pair of points in P to T without violating the pairwise non-crossing property.

Any triangulation T of P has a *weight* defined as the sum of the Euclidean lengths of the edges in T. The *minimum weight triangulation* of P is a triangulation with smallest weight. The MinWtTriangulation problem is the following:

MinWtTriangulation
Input: A set P of points in the plane.
Output: A triangulation T of P of minimum weight.

The MinWtTriangulation is NP-hard (i.e., unlikely to have a polynomial-time solution) if the input is an arbitrary set of points in the plane. However, if the input is more restricted, then there is a nice dynamic programming algorithm that solves MinWtTriangulation in polynomial-time. Specifically, if P is the vertex set of a convex polygon, then there is a "2-dimensional" dynamic programming algorithm to compute a minimum weight triangulation of P. The algorithm runs in $O(n^3)$ time using an $n \times n$ table when given a set P of n points.

Your task is to write a program that takes as input a sequence of vertices of a convex polygon, say in counter-clockwise order, and produces as output a minimum weight triangulation of P. Your output should be an image that shows the minimum weight triangulation computed by your program. The image should be labeled by the weight of the triangulation. You may assume that each coordinate of a point is a real number in the range 0 through 400 (this upper bound is pretty arbitrary). You may also assume that P does not contain wierd degeneracies that often makes programming with geometric objects somewhat tedious. For example, you can assume that no three points in P are collinear.

It will be a good exercise for you to figure out this dynamic programming algorithm on your own and resist the temptation to look on web or in other sources. But, I know that there are several descriptions and animations of this algorithm on the web. Even if you do consult these websites, make sure that the code is completely yours.

Extra credit. There are three ways of getting some extra credit for this programming project:

(i) If you allow the point set P to be specified in arbitrary order rather than in order around the convex polygon, then your program has to do additional work to first correctly order the points. You will get extra credit if your program accepts P in arbitrary order.

(ii) If you allow some interaction between the user and your program, for example, maybe the user can add points by clicking on a canvass or delete points and then recompute the minimum weight triangulation.

(iii) To get a sense of the running time of your program, it would be helpful to run it on relatively large polygons - say with 100 or 200 vertices. It is a pain for the user to have to specify such large polygons in input files, so if you allow for the input point set P to be generated randomly by your program, in addition to being provided in an input file, that will fetch you some extra credit.

What to submit. You should feel free to use a programing language and environment of your choice. As of now, you should submit (i) well-documented source code, (ii) executable, (iii) documentation that tells us how to run your program. As we get closer to the due date and I get a better sense of what sort of programming environments you are using, I might expand this list and also make it more specific. I will open an ICON folder for you to make the submission.
