Homework #3: Point location, polygon triangulation [60 points]
Due Date: in class, on Friday, 27 May 2011 – ** no late days for this homework **

*Note: All papers referenced below are available from the 'Lecture Notes" part of the class web page.*

- **The Common Theory Problems**

**Problem 1. [10 points]**

Consider a subdivision of the plane consisting entirely of rectangles aligned with the $x$- and $y$-axes. Such a subdivision is clearly monotone for any direction $\theta$, $0 < \theta < \pi/2$ (in other words, a line in the direction $\theta$ always cuts each region along at most a single segment). As was shown in class in the monotone subdivision point location lectures, for each fixed direction $\theta$, the corresponding "above" relation between rectangles is acyclic. Prove that there is actually a linear ordering of the regions that is consistent with the "above" relations *for all* $\theta$, $0 < \theta < \pi/2$, *simultaneously*.

**Problem 2. [10 points]**

We discussed briefly in class the point location method due to David Kirpatrick which constructs a hierarchy of coarser and coarser triangular subdivisions over the original triangulation [*SIAM J. Comp., 12 (1983), 28–35*] by repeatedly removing large subsets of independent vertices and re-triangulating the resulting holes. Adapt his method to perform point location on subdivisions consisting entirely of rectangles aligned with the axes. Your method should build a hierarchy of subdivisions that are all of the same type as the original: edges must be vertical or horizontal and regions rectangular. Your asymptotic preprocessing, space, and query bounds should be the same as Kirkpatrick's. (*Hint:* You may want to consider removing elements other that vertices in the coarsening process.)

- **The Additional Theory Problems**

**Problem 3. [10 points]**

Read Section B1 of the "Ruler, Compass, and Computer" paper (included under "Lecture notes" on the course web page). Show how the interval stabbing structure presented there can be adapted to report all the intervals containing the query point, not just count them. The time for the reporting operation should be $O(\log n + k)$, where $k$ is the number of intervals reported (such an algorithm is called 'output sensitive"). Then show that counting and reporting can also be done (within the same time

bounds) when the query object is not a point but another interval, and we are interested in the original intervals intersecting the query interval. The more ambitious of you can now try to use this method plus a sweep line idea to give an $O(n \log n + k)$ algorithm for reporting all intersecting pairs among $n$ axis-aligned rectangles in the plane (again $k$ is the output size) [5 extra points]. This is a very useful operation in design rule checking for VLSI circuits.

**Problem 4. [10 points]**

Let $S$ be the set of vertices of a simple polygon $P$ in the plane and call a diagonal $AB$ of $P$ *extreme* if both $A$ and $B$ are vertices of the convex hull of $S$. Show that, unless $P$ is convex, $P$ can be triangulated without using any extreme diagonals.

**Problem 5. [10 points]**

Let $P$ be a simple polygon on $n$ sides. Show how to compute the vertical trapezoidalization of $P$ in linear time, starting from an arbitrary triangulation of $P$ (we will cover the other direction in class).

**Problem 6. [10 points]**

Let $P$ be a simple polygon of $n$ sides. Consider a particular side $e$ of $P$ as a luminous neon tube, casting light towards the interior of $P$. The illuminated area of $P$ is another polygon $V$, called the *weak-visibility* polygon of $P$ from $e$. Every point of $V$ has the property that "it can see" some point of $e$. Show how to compute $V$ in linear time, starting from an arbitrary triangulation of $P$.

- **The Programming Problem**

**Problem 7. [40 points]**

**Option A: Alpha Shapes**

*Alpha Shapes* are an important early attempt to "understand the shape of a cloud of points." In many applications we want to infer an unknown shape based of a set of sample points drawn for the surface of the shape. The question is "how to connect the dots together" to recover the shape. Clearly there can be many solutions to such a problem. Alpha shapes are so named because of a parameter $\alpha$ that controls how the points are connected.

The goal of the problem is to produce and document a robust implementation of unweighted 2-D alpha shapes together with a user interface that allows interactive exploration.

Read a couple of the original alpha shapes papers (available through the "Lecture notes" part of the class web page):

1. H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. ACM Trans. Graphics 13 (1994), 43–72.
2. H. Edelsbrunner. Alpha shapes — a survey. In Tessellations in the Sciences (to appear)
3. K. Fischer. Introduction to alpha shapes.

Many implementations of alpha shapes exist on the web and you are free to build upon anything you can find (this is real-life software development), including the implementation provided by CGAL
`http://www.cgal.org/Manual/latest/doc_html/cgal_manual/Alpha_shapes_2/Chapter_main.html`.

However, writing your own is also not very difficult.

You should aim for an interactive interface analogous to the one in
`http://cgm.cs.mcgill.ca/~godfried/teaching/projects97/belair/alpha.html`.

In particular, your interface should allow the user to either (1) interactively click to locate points in a workspace, move points, or delete points, or (2) read a 2-D point set from a file and then interactively edit it as in (1). A slider should be provided that allows the user to interactively change and experiment with the value of $\alpha$.

Unlike homework 2, this time we really care for robustness and you should do (and document) all you can to avoid crashes.

Develop and experiment with a number of ideas for automatically selecting the most meaningful values of $\alpha$ — and provide a user interface where the user can cycle among a small number of $\alpha$ values chosen by you based on the data.

In the problem grading, the correctness, efficiency, and robustness of the alpha shape implementation will count for 20 points. The quality of the user interface will be worth up to 10 points, and the ability to automatically select good $\alpha$ values another 10.

For an extra credit of 5 points, you can add the ability to display alpha hulls as well as alpha shapes.

For an extra credit of another 10 points, you can implement (and provide an interface for) weighted alpha shapes. You should allow the weights to be either entered by the user, read from a file, or to be estimated automatically using, for example, the local density of the points.

**Option B: Build Your Own Project**

If you are working in an applied area, you have the option of submitting a project that is related to, or part of, another piece of work that you are doing and which involves geometric algorithms. Such a project should clearly demonstrate the use of computational geometry methods and provide an assessment of their effectiveness. In your write-up you will need to provide enough background about the other area to allows us to evaluate the appropriateness of the methods you have chosen to implement.

A one page proposal about such an option B project should be e-mailed to the instructor and the CA as soon as possible and no later than Monday, 16 May 2011. Approval of the project proposal by the instructor is needed for you to proceed with this plan.