Homework #2: Voronoi and Delaunay diagrams [60 points]
Due Date: Wednesday, 25 February 2009

- **The Common Theory Problems**

Problem 1. [5 points]

The edges of both the Delaunay and Voronoi diagrams are line segments. Give a simple necessary and sufficient condition on a pair of sites A and B so that AB is a Delaunay edge *and* AB intersects its dual Voronoi edge.

Problem 2. [10 points]

Delaunay's theorem states that the triangulation of a set S on n sites in the plane is a Delaunay triangulation if and only if every edge passes the `InCircle` test with respect to its two adjacent triangles. This gives a linear-time algorithm to verify that a triangulation is Delaunay, and it also suggests the following algorithm to fix it up, if it is not: Start with any triangulation of the n sites. If an edge fails the `InCircle` test, then swap it with the other diagonal of the quadrilateral formed by the two adjacent triangles (this edge must pass the test). Continue in this fashion, until all the edges pass the `InCircle` test. Make this idea into a rigorous algorithm and prove its correctness. Prove that your algorithm always terminates in $O(n^2)$ steps. (*Open problem:* Can this method be parallelized in an interesting way? What processor/time bounds can you get?)

Problem 3. [10 points]

Second-order Voronoi diagram:
Given n sites in the plane, suppose we partition the plane according the nearest *and* the second-nearest site. Thus all points in the same region have the same nearest and second-nearest neighbor. This is sometimes called the *second-order* Voronoi diagram. Show that it is also of size $O(n)$ and it can be computed in time $O(n \log n)$.

- **The Additional Theory Problems**

Problem 4. [10 points]

Davenport-Schinzel sequences of order 2 and triangulations:
Let P be a convex polygon with n vertices. A triangulation of P is a collection of $n-3$ non-intersecting diagonals connecting pairs of vertices of P and partitioning P into $n-2$ triangles. Set up a correspondence between such triangulations and $DS(n-$

$1,2)$ sequences, as follows. Number the vertices $1,2,\ldots,n$ in their order along the boundary ∂P. Let T be a given triangulation. Include in T the edges of P too. For each vertex i, let $T(i)$ be the sequence of vertices $j < i$ connected to i in T and arranged in *decreasing* order, and let U_T be the concatenation of $T(2), T(3), \ldots, T(n)$.

(a) Show that U_T is a $DS(n-1,2)$ sequence of maximum length.

(b) Show that any $DS(n-1,2)$-sequence of maximum length can be realized in this manner, perhaps with an appropriate renumbering of its symbols.

(c) Use (a) and (b) to show that the number of different $DS(n-1,2)$ sequences of maximum length is $\frac{1}{n-1}\binom{2n-4}{n-2}$ (where two sequences are different if one cannot obtain one sequence from the other by renumbering its symbols).

Problem 5. [10 points]

We discussed in class the lifting map $\lambda(x,y) : (x,y) \mapsto (x,y,x^2+y^2)$ from points in the xy-plane to points on the paraboloid of revolution $z = x^2 + y^2$. As we mentioned, the downwards-looking faces of the convex hull of the lifted images of a collection of sites in the xy-plane correspond to the Delaunay diagram of the sites. What do the upward-looking faces correspond to? Is there an analogous Voronoi diagram? How fast can it be computed?

Problem 6. [15 points]

We want to do an analysis of the randomized incremental algorithm for Delaunay triangulations discussed in class, but based on the appearance and disappearance of Delaunay edges during the process, rather than that of triangles.

We defined the *weight* of a triangle Δ to be the number of sites inside the circumcircle of Δ and related, in the class analysis, this weight with the probability that Δ would ever appear as a Delaunay triangle during the random process. How should we define the corresponding notion of the weight of an edge $e = AB$? Below we suggest one possibility, but feel free to explore your own definition, as long as it leads to the same eventual result.

One way is to let the weight of an edge AB be the w for which there is a circle through A and B which contains exactly w other sites to the left of AB and w sites to the right. Show that this notion of weight is well defined (i.e., such a w always exists and is unique). Prove an edge of weight w arises as Delaunay at some point of the random process with probability at most $4/(w+1)(w+2)$. By emulating the argument given in class for triangles, show the combinatorial-geometric result that in any collection of n sites, the number of edges of weight at most w is $O(n(k+1))$.

Briefly outline how combining these results (using the summation-by-parts trick shown in class) allows us to conclude that the expected number of edges that arise

during the random process is $O(n)$ (of course, this also follows immediately from the result for triangles).

Given an edge AB in a group of n sites, how fast can you calculate its weight? How fast can you calculate the *minimum* number of other sites whose deletion would make AB a Delaunay edge? Are those two quantities related?

- **The Programming Problem**

Problem 7. [35 points]

Kinetic Delaunay Triangulation:

The goal of the problem is to produce an animation of the Delaunay triangulation among moving points in the plane. The points will follow simple polynomially parametrized motions and will be confined to move within a specified square in the plane. The latter constraint will be enforced by having the point bounce (reflect) off the walls of the square container. When they do so, their entire polynomial trajectory is reflected around the bounding wall. The points themselves may be given a 'non-zero size' (think of small disks of a fixed radius), and in that case they will be required to bounce off each other when their disks collide.

Since the motion of the points will be simple and predictable in the short term, this is an ideal setting for using the framework of *Kinetic Data Structures*, which are event-driven structures that allow the efficient maintenance of geometric attributes. We remarked in class that a Delaunay triangulation can be certified locally: if every edge of the triangulation is locally Delaunay (passes its `InCircle` test), then the whole triangulation is globally Delaunay. This implies that the points can move and, as long as all the edge `InCircle` tests remain valid, the triangulation is still Delaunay. Since we know the motion of the points, we can predict when these `InCircle` certificates will fail. If we put these failure times into an event queue, we can then allow the motions to proceed until the first certificate fails. At that instant the Delaunay triangulation can be repaired with a single operation, the *edge-flip* we discussed in class. The new edge must then add the failure time of its `InCircle` certificate to the the event queue, and the simulation can proceed.

The above is a very simple example of *kinetic proof repair* for the certification of a geometric structure. More information about this approach and several worked out examples can be found in handout 3 and the papers

1. *Kinetic data structures — a state of the art report*, L. Guibas. Proc. 3rd Workshop on Algorithmic Foundations of Robotics, Houston, TX, 191–209, 1998.
2. *Data structures for mobile data*, J. Basch, L. Guibas, and J. Hershberger. J. Algorithms, **31**, 1–28 (1999).

These can be accessed on the web from

`http://graphics.stanford.edu/~guibas/kinetic_papers.html`

You will be provided with an input file giving the initial positions and parametric polynomial motions for the moving points. All motion will take place inside a specified square in the plane. A radius will be also given for the points; if the radius is non-zero, then your points will behave like disks of that radius. You may assume that initially all these disks are disjoint.

You will then need to design a data structure for representing the evolving Delaunay triangulation. Furthermore:

- You will need to compute the initial Delaunay triangulation of the point set; you may do so by using the relevant procedures available in CGAL and you can assume that no degenerary is initially present.
- You will need to implement a priority queue mechanism for handling the events generated by the kinetic structure; an example is the failure of `InCircle` certificates certifying Delaunay edges.
- For each Delaunay edge in the triangulation there will have to be an event scheduled in the event queue for the time when that edge stops being Delaunay next; you will be provided with a polynomial solver for calculating these event times.
- You will need to animate the evolving triangulation and draw it on the screen; some help on how to do this will be provided. As an optional part of the assignment, you may want to allow the user to see the Voronoi diagram as well as or in lieu of the Delaunay triangulation.

In addition to the above, you will need to detect collisions between the points and the four walls of the square and, if a non-zero radius has been specified for the points in the input file, between the points themselves as well. The collisions will be assumed to be perfectly elastic; the walls have infinite mass and all points have the same finite mass. You can implement these collision tests by scheduling collision events in the event queue between points and the walls, and among the points themselves. You may do so in a straightforward way, or use more sophisticated structures — for example, only points on the convex hull can collide with the walls. In addition, as we showed in class, the closest pair of points always corresponds to a Delaunay edge, so before two points can collide they have to become Delaunay neighbors. Note that, whenever a point collides, its motion changes and therefore the failure times for all certificates involving that point must have their failure time recalculated.

Details about the input file format, the polynomial solver, and drawing on the display will be available by following the programming link from the class web page. What you ought to hand in will also be discussed there.