WELCOME TO CS 451 — COMPUTATIONAL GEOMETRY

When and where: T,Th 3:00-4:20 in Computer Science Bldg, 302

Your instructor: Bernard Chazelle, userid= chazelle, 404, x5380

Grading: Based on homework assignments and final project

Textbook (strongly recommended): *Computational Geometry, Algorithms and Applications,* de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., Springer-Verlag, 1997. A more elementary –and narrowly-focused– text is "*Computational Geometry in C*", Cambridge Univ. Press, 2nd ed., 2001, by J. O'Rourke. Two other good books, less comprehensive but more advanced, are "*Algorithms in Combinatorial Geometry*", H. Edelsbrunner, Springer-Verlag, 1987, and "*Multidimensional Searching and Computational Geometry*", by K. Mehlhorn, Springer-Verlag, 1984. The classic, "*Computational Geometry: An Introduction*", by F.P. Preparata and M.I. Shamos, Springer-Verlag, 1985, is still good but is showing signs of age.

This course is intended for people who like mathematical reasoning, computers, and doodles. The word *geometry*, here, bears a fairly different meaning from the one you (sometimes painfully) associate with your high-school years. There, you were taught the arcane secrets of highly regular geometric structures (*e.g., regular polygons, platonic solids, conic sections, etc...*), here you will learn to deal with real-world objects which engineers, chemists, statisticians, as well as yourself constantly encounter. Indeed, one fundamental feature of computational geometry is to make minimal assumptions about the shape of the objects under consideration. A polygon may be arbitrarily shaped, and so may an arrangement of points in the plane, to take two common examples. The two key questions of computational geometry are:

1. *How do we represent geometric objects?*
2. *How do we manipulate them?*

The latter question brings to mind perhaps the most fundamental feature of computational geometry: to be *algorithmic*. In the same way as mathematicians are happy knowing that rational numbers *can* be divided, while computer scientists find rest only once they know *how* to divide them, geometers suffer the same dichotomy. Whereas a pure geometer will be content with Chvatal's theorem, stating that no more than $\lfloor n/3 \rfloor$ stationary guards are necessary to watch over an entire n-wall gallery, a computational geometer will insist upon locating where to station the guards. There step in all the algorithmic techniques, fancy data structures, and programming tricks that have made your journey in the land of computer science so idyllic.

But there is much more to come, so here is a brief overview of what's coming up.

1. **Introduction** — Representation of geometric objects; elementary abstract data types; similarity of simple polygons.

2. **Convex Hull Algorithms** — Jarvis march; Graham-scan; divide-and-conquer; prune-and-search method. Probabilistic methods. Lower bounds.

3. **Point Location** — Strip method; Kirkpatrick's algorithm; chain method with fractional cascading.

4. **Voronoi Diagrams** — Divide-and-conquer; Fortune's sweep-line method. Minimum spanning tree. Higher-order Voronoi diagrams; parabolic lifting map.

5. **Polygons** — Guard-placement; ray-shooting; triangulation.

6. **Duality** — Brown's dual transform; polarity. Equivalence between convex hulls and halfspace intersections.

7. **Intersection problems** — Polyhedral intersections; segment intersections; ray-shooting.

8. **Multidimensional searching** — Orthogonal range searching; line arrangements; ε-net theory; partition trees.

9. **Optimization problems** — Linear programming; convex programming.