Lecture #1:	Tuesday, 30 March 1999
Topics:	Course Introduction
Lecturer:	Leonidas Guibas

Introduction — Geometric Algorithms

Computational Geometry is, in its broadest sense, the study of geometric problems from a computational point of view. At the core of the field is a set of techniques for the design and analysis of geometric algorithms. In what follows we present a few facts about the area and then discuss what we propose to cover in the course.

Some of the excitement of computational geometry is due to a combination of factors: deep connections with classical mathematics and theoretical computer science on the one hand, and many ties with applications on the other. Indeed, the origins of the discipline clearly lie in geometric questions that arose in areas such as computer graphics and solid modeling, computer-aided design, robotics, computer vision, geography, etc. Not only have these more applied areas been a source of problems and inspiration for computational geometry but, conversely, several techniques from computational geometry have been found useful in practice as well. The level of mathematical sophistication needed for work in the field has risen sharply in the last seven to eight years. Nevertheless, many of the new algorithms are simple and practical to implement—it is only their analysis that requires advanced mathematical tools. We intend this course to cover the theoretical, as well as some of the the practical, aspects of geometric computation.

Course Outline

There is now so much material in computational geometry that probably a full-year course is needed to cover all the basic techniques. Nevertheless, given some algorithmic preparation, we can cover lots of ground even in one quarter. Below is a list of topics to be covered—but we do not promise to cover them in the order listed.

- *Geometric fundamentals*

  Computational primitives in two and three dimensions and their implementation; models of computation and lower bounds; geometric duality.

- *Convexity*

  Algorithms for convex hulls of point sets in two and three dimensions; convex polygons—properties and algorithms.

- *Arrangements*

  The combinatorics of line arrangements, including the zone theorem; sweep-line methods for arrangements—topological sweep; Davenport-Schinzel sequences; many-cell problems.

- *Proximity problems*

  Voronoi Diagrams and Delaunay triangulations; algorithms and applications.

- *Geometric searching*

  Point-location in planar subdivisions; fractional cascading and other efficient data-structuring techniques; three-dimensional analogs.

- *Partition trees and range searching*

  The ham-sandwich theorem; decimation methods; range-searching problems of various kinds.

- *Triangulations*

  Triangulating a simple polygon and applications to shortest-paths; reductions among geometric problems; decompositions of polyhedra; questions of optimality.

- *Random sampling techniques*

  Random sampling for partitioning; randomized incremental algorithms; ϵ-nets; making randomized algorithms deterministic; cuttings and their applications.

- *Visibility and shortest path problems*

  Visibility graphs and their uses; Euclidean minimum spanning trees and more on shortest path problems.

- *Robustness in geometric computation*

  Issues in topological consistency; handling of degeneracies; rounding of geometric structures; robust algorithms.

Bibliography

The main text for the course is the course reader, comprised of a number of survey articles plus lecture notes written by the lecturer or scribed with the help of students in previous years of this class.

There are now several excellent new textbooks commercially available that cover most of the same material as well. A very well written elementary introduction is *Computational Geometry in C* by J. O'Rourke (Cambridge U. Press, second edition, 1998). The book *Computational Geometry, Algorithms and Applications*, by M. de Berg, M. van

Kreveld, M. Overmars, and O. Schwarzkopf (Springer Verlag, 1997) has an excellent selection of topics, each motivated by a practical application. The newest book in English *Algorithmic Geometry*, by J-D. Boissonnat and M. Yvinec (Cambridge U. Press, 1998, translated from the French by H. Brönnimann) offers more advanced coverage at the level of a research monograph. An extensive survey of the field is given in the *Handbook of Discrete and Computational Geometry*, edited by J. Goodman and J. O'Rourke (CRC Press, 1997).

The previous generation of books includes the text by K. Mulmuley *Computational Geometry: an Introduction through Randomized Algorithms* (Prentice Hall, 1993). Also in that group are the excellent monograph *Davenport-Schinzel Sequences and their Geometric Applications* (Cambridge U. Press, 1995) by M. Sharir and P. Agarwal, as well as the survey text *Combinatorial Geometry* by J. Pach and P. Agarwal. Currently available older bibliography includes three books. The oldest one is volume III of K. Mehlhorn's *Data Structures and Algorithms*, titled *Multidimensional Searching and Computational Geometry* (Springer-Verlag, 1984). Next is the classic *Computational Geometry, An Introduction*, by F. P. Preparata and M. I. Shamos (Springer-Verlag, 1985), and most recent is *Algorithms in Combinatorial Geometry* by H. Edelsbrunner (Springer-Verlag, 1987).

Papers in computational geometry appear in a variety of computer science journals, including the *ACM Transactions on Graphics*, *Algorithmica*, the *Journal of Algorithms*, the *Journal of the ACM*, the *SIAM Journal on Computing*, and others. There is a ten-year-old journal, *Discrete and Computational Geometry*, that is primarily devoted to this field. Two more such journals began publication some years ago, the *International Journal of Computational Geometry & Applications*, and *Computational Geometry, Theory and Applications*.

There is an annual conference, the *ACM Annual Conference on Computational Geometry*, now in its fifteenth year, whose proceedings are a useful reference for much of the work in the field. There is also a Canadian conference in the area that started ten years ago, as well as annual workshops in Europe and the Far East. Other well established theory conferences, such as *STOC*, *FOCS*, *SODA*, or *ICALP*, also get a sprinkling of high quality geometry papers.

The survey papers included in the primary reader contain as a whole a reasonable set of references to the main research papers in the field. An on-line geometry paper data-base containing several thousand papers is also available and will be described more fully in the web page for the course — more about that in a later handout.

Office hours, address data, etc.

Instructor:	**Prof. Leonidas J. Guibas**
office:	Gates 374
tel.:	723-0304
e-mail:	guibas@cs.stanford.edu
office hours:	Wednesday, 10:00–12:00 noon

- **Teaching Assistant:** if any, will be announced
  Additional data to be handed out.

- **Course Secretary:** **Mina Madrigal**
  office: Gates 146
  tel.: 723-4137
  e-mail: `mina@cs.stanford.edu`

Handouts and Course Notes

Every student should have a copy of the primary course reader. There may be additional handouts during the course. When we cover material that is not well represented in the text or the reader, the instructor may call for student volunteers to help in scribing the corresponding lecture(s).

Web page

The most up-to-date information on the class is available at

`http://graphics.stanford.edu/courses/#cs368/`

It contains an evolving syllabus, and copies of handouts and homeworks, as well as links to useful resources on the web.

Homeworks, Exams, Grading, etc.

The course will have three substantial homework assignments. There will be no final exam, but there will be a midterm whose function will be to test breath but not depth. The schedule will be as follows:

homework	**handed out**	**due**
#1	Tuesday, 13 April	Tuesday, 27 April
#2	Tuesday, 27 April	Tuesday, 11 May
#3	Tuesday, 11 May	Tuesday, 1 June
midterm	in class	Tuesday, 18 May

Students may elect to take this class through either a *theoretical track* or through an *applied track*. Students in the theoretical track will have to do sets of 'paper-and-pencil' type problems as part of each homework. Those electing the applied track will substitute a programming project in lieu of some of the problems in the last two homeworks. Every student taking the class for credit is expected to do all the homeworks (according to the track they are in) and take the midterm. Programming projects will be demonstrated to the entire class at the end of the quarter.

Collaboration with other students in the class in doing the homeworks is permitted, in fact encouraged. However, each write-up must be individually composed and the names of the collaborators must be listed for each problem. Those in the applied track my work in groups of up to two students for the programming project(s). For the final grade we will count each of the homeworks and the midterm as 25% of the grade. *Please do the homework*—there is no other way to learn the material.

It is very important in this course that every homework be turned in on time. We recognize that occasionally there are circumstances beyond one's control that prevent an assignment from being completed before it is due. You will be allowed two classes of grace during the quarter. This means that you can either hand in two assignments late by one class, or one assignment late by one week. Any other assignment handed in late will be penalized by 20% for each class that it is late, unless special arrangements have been made previously with the instructor.

All course work must be handed in by Tuesday, 1 June, 1999.

Scribing

Most of the material in this course is covered well by the papers and notes in the course reader. But there will be some lectures for which that is not true. For those lectures that cover new material, we'd like to have some student volunteers act as scribes. If you do not already have experience, we recommend that you become familiar with LaTeX and some drawing program that produces postscript files, such as `Xfig` on Unix workstations, or `Adobe Illustrator` on a PC or Mac. We will provide you with LaTeX style files for the course notes, and macros for incorporating any figures you produce. Handout #3 in the reader discusses this process in more detail.

Scribes should provide a draft of the lecture notes to the instructor, in both hardcopy and electronic form, no later than a week after the lecture being scribed.