Lecture #1: Monday, 28 March 2011
Topics: Course Introduction [Revised Dates]
Lecturer: Leonidas Guibas

## Introduction to Geometric Algorithms

*Computational Geometry* is now a bit over thirty years old. In the broadest sense, the field is the study of geometric problems from a computational point of view. At its core is a set of techniques for the design and analysis of geometric algorithms, for the development of certain key geometric data structures, and of tools for the robust implementation of these on current computer hardware, using familiar computer languages. In what follows we present a few facts about this 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, molecular modeling, geography, etc. Not only have these more applied areas been a source of problems and inspiration for computational geometry but, conversely, numerous techniques from computational geometry have been found useful in practice as well. A special challenge in the field is the need to deal with data that has both continuous aspects (point coordinates, plane equations), as well as combinatorial aspects (incidence structures, graphs, polytopes) — and to keep these consistent.

The mix of continuous and discrete representations gives rise to new challenges and indeed the level of mathematical sophistication needed for work in the field has risen sharply in the last two decades. Nevertheless, many of the new algorithms are quite simple and practical to implement — it is only their analysis that requires advanced mathematical tools. We intend this course to cover a mix of the theoretical and practical aspects of geometric computation.

We note also that computational geometry problems in spaces of modest to high dimensions, such as nearest neighbor searching, have recently become quite important with the increasing need to index all kinds of documents, both text and multimedia — fueled by the explosive growth of search engines on the web.

## Course Outline

There is now so much material in computational geometry that at least 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. At least in low dimensions, figures and diagrams can greatly help with intuition.

Below is a list of topics to be covered — but we do not promise to cover them in the order listed, and not all may fit in one quarter. Beyond CS268 (this course), there is CS468 (Topics in Geometric Algorithms). It is offered many quarters with different material and can be repeated for credit.

- *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. Approximate Voronoi diagrams.

- *Triangulations*

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

- *Geometric searching*

  Point-location in planar subdivisions; fractional cascading and other efficient data-structuring techniques; three-dimensional analogs. Balanced-aspect-ratio and balanced-box-decomposition trees and their applications.

- *Geometric optimization*

  Smallest enclosing balls and ellipsoids, LP-type problems, decimation, parametric search.

- *Visibility and shortest path problems*

  Visibility graphs and their uses; Euclidean minimum spanning trees; shortest path problems amidst obstacles.

- *Geometric sampling techniques*

  Random sampling for partitioning; randomized incremental algorithms; $\epsilon$-nets; making randomized algorithms deterministic; cuttings and their applications. Core sets and applications.

- *Partition trees and range searching*

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

- *Curve and surface reconstruction*

  Reconstruction from sample points; sampling conditions; crust and cocone algorithms; the witness complex and its applications.

- *Robustness in geometric computation*

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

## Bibliography

The main text for the course is the CS268 lecture notes, class notes written by the lecturer and/or scribed with the help of students in previous years of this class (some when it was numbered as CS368). These will be posted on the class web site (more below).

There are now several excellent 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 well motivated by a practical application — this is the recommended textbook for this class. The text *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. The monograph *Curve and Surface Reconstruction : Algorithms with Mathematical Analysis* by Tamal Dey (Cambridge University Press, 2007) is focused on recent work in surface reconstruction. 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, 2004 — second edition), and in the *Handbook of Computational Geometry*, edited by J. Sack and J. Urrutia (North-Holland, 2000).

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 (Wiley-InterScience, 1995). 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 *ACM Transactions on Algorithms*, the *SIAM Journal on Computing*, and others. There are three specialized journals primarily devoted to this field, *Discrete and Computational Geometry*, the *International Journal of Computational Geometry & Applications*, and *Computational Geometry, Theory and Applications*. Almost all of this material is now available on line through your Stanford account.

There is an annual conference, the *ACM Annual Conference on Computational Geometry*, now in its twenty-seventh year, whose proceedings are a useful reference for much of the work in the area. There is also a Canadian conference in the field that started twenty or so years ago, as well as annual workshops in Europe and the Far East. Other well established theory conferences, such as *STOC*, *FOCS*, *SODA*, or *WADS/SWAT*, also get a good share of high quality geometry papers — the last two usually contain a substantial fraction.

An on-line geometry paper data-base containing several thousand papers is available and can be found on-line at `http://compgeom.cs.uiuc.edu/~jeffe/compgeom/biblios.html`

A closely related area is that of *computational topology* which looks at algorithms for the computation of classical topological invariants. Most of the above journals and conferences contain material in computational topology as well. We expect to cover more computational topology in future editions of this course.

## Office hours, address data, etc.

- **Instructor:** **Prof. Leonidas J. Guibas**
  office: Clark S293
  tel.: (650) 723-0304
  e-mail: guibas@cs.stanford.edu
  office hours: Monday, 1:00 – 2:30 pm

- | | |
  |---|---|
  | **Course Assistant:** | **Daniel Chen** |
  | office: | Clark S297 |
  | phone: | (650) 725-6521 |
  | email: | danielc@cs.stanford.edu |
  | office hours: | Tuesday, 9:00 – 11:00 am and |
  | | Thursday, 2:00 – 3:00 pm |

- | | |
  |---|---|
  | **Course Admin** | **Monica Niemiec** |
  | office: | Gates 368 |
  | tel.: | (650) 725-9494 |
  | e-mail: | mniemienc@graphics.stanford.edu |

## Handouts and Course Notes

When we cover material that is not well represented in the extant set of course notes of the (optional) textbook, the instructor may call for student volunteers to help in scribing the corresponding lecture(s).

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. 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.

These composition tools can also be quite useful in preparing homework solutions.

## Web page

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

`\http://graphics.stanford.edu/courses/cs268-11-spring/` or
`http://cs268.stanford.edu/`

This URL 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 breadth but not depth. The schedule will be as follows:

| **Homework** | **handed out** | **due** |
|---|---|---|
| #1 | Monday, 11 April | Monday, 25 April |
| #2 | Monday, 25 April | Monday, 9 May |
| #3 | Monday, 9 May | Friday, 27 May |
| **Programming project proposal** | | Monday, 16 May |
| **Midterm** | in class | Wednesday, 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 assignment in lieu of some of the theory problems. We will be using the CGAL programming environment (`http://www.cgal.org/`) in this course. 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. Switching among tracks is permitted only up to the first homework due date. The programming track will require a final project; all such 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 groups of up to three students — in fact collaboration is 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 may work in groups of up to three 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 two classes. 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 Wednesday, 1 June, 2011.

Homework solutions will be handed out in hardcopy in class. All other handouts and class materials will be available on the web.