Computational Geometry Education for Computer Graphics Students

I. Kolingerová[†1]

[1]Centre of Computer Graphics and Data Visualization,
Department of Computer Science and Engineering,
University of West Bohemia, Pilsen, Czech Republic

Abstract

The paper surveys main features of computational geometry and presents an argumentation that a course oriented to the applied computational geometry should be a part of computer graphics curriculum as it teaches effective algorithmic methods and helps to develop an abstract thinking. A possible contents of the course and forms suitable and interesting for computer graphics students are discussed. The students' feedback for such a course has been mostly positive.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages and systems K.3.2 [Computer and Information Science Education]: Computer science education K.3.2 [Computer and Information Science Education]: Curriculum

1. Computational geometry – a tool for computer graphics

Computational geometry is a study of algorithms and data structures for geometrically formulated problems. Typical tasks solved by computational geometry are convex hulls, point location, intersections, visibility, triangulations and other partitions in 2D and 3D, motion planning. These problems are inspired by applied sciences, such as robotics, databases, image and pattern recognition, cartography, GIS, biology, civil engineering and many others, see a more detailed list in [App96]. However, most of the problems in the scope of computational geometry are inspired by computer graphics. With some simplification, computational geometry can be understood as fundamental research for computer graphics.

Main advantages of this scientific discipline are its exactness in language as well as in algorithms, systematic use of complexity evaluation, beauty of its algorithms and efficiency of the proposed solutions. Like mathematics, also computational geometry not only teaches tools but also trains an abstract reasoning.

Computational geometry has also its disadvantages, such as its sometimes too high abstraction, which causes difficulties to more practically oriented engineers. If an engineer unused to high formalism of computational geometry tries to read some paper from this area, he/she usually cannot get too much of it because of 'somebody else's language', an absence of illustrations, sometimes also an absence of implementation results. Computational geometry sometimes also shows an unhealthy distance from practical life (it usually does not care whether the solved problem is academic or not). Due to these features, computational geometry has sometimes low influence on applied research and solutions of practical problems. Asymptotically optimized algorithms which are in focus of computational geometry also do not necessarily work well in practice – if the question of expected complexity and expected type of input data are not properly considered in the algorithm proposal. Sometimes the algorithms vitally depend on complicated data structures or on theoretically sound but never implemented algorithms. Special cases are often not discussed or the published algorithm has been proved but not implemented.

† supported by Ministry of Education, Youth and Sports of the Czech Republic - project No. LC 06008

Most of these negative features are gradually disappearing with growing maturity of the computational geometry discipline. Experts in computational geometry worked hard to cure the problems. Computational geometry has already grown out into the stage where it is able to serve to applied sciences as a marvelous tool, see [deB97], [Dob92]. Still, there is a human factor working against it on the side of applied sciences community. The application experts are sometimes not willing to accept high abstraction and formalism of computational geometry and avoid its materials to the harm of both sides. Therefore, the efficient and beautiful algorithms of computational geometry are not so widely known and used in practice as they would deserve.

One way how to bring computational geometry to experts from applied sciences is to present it in a more accessible, a bit applied way, with the stress rather on algorithms and their practical use in various problems than on the proofs, and include its fundamental methods and 'tricks' in engineering education, namely in computer graphics area. If young computer graphics students get some survey and explanation of computational geometry methods and are taught its way of thinking, if they see that this discipline provides a set of beautiful tools and tricks which can be useful in practical computer graphics problems, they loose their fear of this discipline. In a better case, the student will not avoid the computational geometry resources any more and will be able to work with them in the future, to look for and find a new inspiration and new brilliant ideas. In the worse case, he at least knows some fundamental computational geometry methods for a future use.

For these reasons, computational geometry is in many cases a part of the computer graphics undergraduate and graduate study. Unfortunately, a usefulness of computational geometry is not so widely recognized to find its expression either in the proposed computer graphics curriculum and knowledge base [LO06], or in recommendations from the CGE06 Computer Graphics Education Workshop [BCF*06].

Computer graphics programmers, not counting computer graphics applications, come from two main areas, mathematics and computer science. Also computational geometry courses can be either more theoretically or more practically oriented, where by theoretical we mean a classical style of computational geometry with proofs and mathematical formulations of knowledge and by practical a style oriented to algorithms, to the questions related to their implementation and applications. An example of a mixed course see in [Min].

In this paper we will show how a course of practically oriented computational geometry, suitable mainly for undergraduate computer graphics students, can look like, in what form and extent it could be taught, what kinds of 'entertainment' can be used to make it more attractive for students. We will also mention some positive and negative experience that was collected teaching this course at two different types of faculties at different universities. The course has been developed after experience with five years of teaching a classical computational geometry course at a technical faculty and tested in other five years in a technical faculty, and, at the same time, in four years in a mathematical faculty, in all cases for undergraduate students oriented to computer graphics, with an exceptional participance of individual graduate students. Faculties providing this course were: Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic and Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic.

Main differences between the course presented in this paper and a traditional style of teaching computational geometry are: more applied character of our course which is in such a case more accessible to the computer graphics students, new forms of project work given to the students and more competitive style of student assignments, which makes the course more interesting for the students.

As far as we know, the paper oriented to computational geometry teaching for computer graphics students has not been published before, although papers addressing other parts of computer graphics education exist, such as inclusion of virtual reality [Zar05].

The paper is organized as follows. Section 2 presents the recommended contents of the course lectures and explains the most important points concerning partial topics. Section 3 shows a style of seminars, section 4 gives details about students' course projects. Section 5 reports teacher's experience from both types of faculties and feedback collected from the students. Section 6 concludes the paper.

2. Computational geometry course contents

Let us introduce a proper content of lectures for an undergraduate computational geometry course, suitable for computer graphics students. A proper placement of the course is the latter cycle of the European curriculum (an advanced course in the vocabulary of CGE 06 Workshop) [LO06], [BCF*06]. The course as described is primarily intended for undergraduate students but its contents could be also used for graduates if needed.

A suitable extent of this course is one semester in the length of 13-15 weeks, 2 hours for lectures plus 2 hours for seminars per week. To be able to take such a course, the student should have fluent knowledge of programming and fundamental knowledge of data structures. It is useful if the course participants have already some knowledge of computer graphics fundamentals - it is then easier for them to make visualization or user interface to their implemented algorithms. It is not absolutely necessary - the teacher may alternatively provide some visualization programs or libraries for user interface, made by previous students.

The content is as follows:

1. Introduction - computational geometry scope, degeneracy and robustness questions, application areas and examples, algorithmic complexity and algorithm evaluation, main computational geometry techniques
2. Geometric search – point location in convex and non-convex polygons, planar subdivision, range search
3. Convex hulls in 2D and 3D
4. Voronoi diagrams, their applications and generalizations
5. Planar triangulations of point sets and their applications
6. Tetrahedrizations and their applications
7. Polygon triangulations, convex and trapezoid partitions, art gallery problem
8. Intersections: line segments, polygons, polyhedra, half-planes and half-spaces
9. Motion planning
10. Trends, tasks and news in computational geometry.

Items 1-5, 8 of this course content are fundamental and can be recommended into any computational geometry course, items 6-7, 9-10 could be replaced by other topics according to the particular specialization of the students (e.g., GIS specialization may bring more stress on 2D algorithms with polygons). Let us explain the contents in more detail.

Introduction

Level of abstraction of an average technical student is not as high as desired. Many students are not used to think about algorithms from more viewpoints. Namely, from the viewpoint of algorithmic complexity, which gives a general idea about its behavior regardless a particular computer influence; from the viewpoint of numerical stability and robustness; from the viewpoint of implementation difficulty, and, which is for beginners the most difficult way of thinking, from the viewpoint of suitability to a particular application problem, considering expected input data. It is reasonable at the beginning of the course to state all these viewpoints and introduce a proper way of thinking when an efficient solution to some problem is searched. Of course, it is not enough to state all these big truths at the beginning, they have to be recalled again and again and the students inspired to think in this way.

A survey of main general computational geometry techniques, such as sweeping, sorting, divide and conquer, locus approach, duality, combinatorial analysis, prune and search and geometrical transformations, helps to show at the beginning main tiles which will be in many variants used during the course. Also, existing libraries in this area, such as [CGAL], should be mentioned.

Geometric search

As the topic of point location is the first exposition how simple the algorithms for convex or star-shape polygons can be in comparison with those for general polygons, it is useful to introduce fundamental categories of polygons, such as convex, star-shape, monotonous and simple. Thanks to this information, students learn first to think whether they know something about the shape of polygons in their application and only then to propose or pick an existing algorithm. Another useful knowledge gathered in this topic is how to decide about the use of algorithms with and without preprocessing for a particular problem. It is also important to introduce orientation tests computed by determinant signs (geometric predicates) and to point out the question of numerical inaccuracy, its consequences for such tests, some possibilities how to solve the problem and existing libraries (e.g., [She96]).

The other part of the geometric search topic, range search, is devoted mainly to special data structures, such as k-D tree or trapezoidation. It should be pointed out to the students that efficiency of location data structures strongly depends also on the expected size of the query range – if comparable with the data size, no improvement can be achieved with any of these data structures.

Convex hulls

This topic is very useful, too, as it is an ideal example where to show the difference between the worst-case and the expected-case optimal algorithms, also a term of output-sensitivity can be explained here (e.g., the gift wrapping algorithm provides a good opportunity [PS85]). The student can also see some examples of algorithms which work well in 2D but cannot be used in 3D, such as the well-known Graham scan [ORo98]. Also, a wide range of convex hull applications helps to see usefulness of computational geometry algorithms.

Voronoi diagrams

Again, various interesting and even exotic applications can be presented for this topic, together with a wide scale of different algorithms, see [dBv*97] [PS85], [ORo98]. As most of existing algorithmic strategies have been used to design Voronoi diagram algorithms, students can be led to improve their understanding of general features of particular algorithmic strategies. Such an understanding helps to derive most of properties of the implemented algorithm in advance, according to the algorithmic strategy used in it. For example, Voronoi diagram construction by incremental insertion will have completely different properties than divide & conquer based program and most of their properties can be guessed in advance; however, for those students who are 'implementation oriented', such a possibility of a 'guess' made in advance can be quite surprising as they have not been used to such a level of abstraction before.

Planar triangulations

Triangulations are one of key topics for computer graphics. Therefore, it can be recommended to present not only the

traditional Delaunay triangulation and greedy triangulations, but also triangulations useful for special applications, such as constrained triangulations [Ang97], [Slo93], data dependent triangulations [DLR90] or multicriteria-optimized triangulations [KF01] and to point out their special properties. As for Delaunay triangulation (as well as for Voronoi diagrams), algorithms based on nearly all algorithmic strategies have been developed and also parallel algorithms of different categories exist, the topic can be used to show all common and general features of the particular algorithmic strategies, so that the students were able to use them for the problems, which they may face in the future.

As the students should be educated also on problems which have not been fully satisfactorily solved yet, the minimum weight triangulation should be mentioned to show such a problem and to point out that polynomial algorithms with higher than quadratic complexity are for large data useless.

Tetrahedrizations

3D Voronoi diagrams and 3D triangulations (tetrahedrizations) are very useful in computer graphics applications but it is a relatively difficult topic, more suitable for a graduate course. Therefore, it can be recommended to touch this topic only slightly and without details, just to know main problems in comparison to 2D, examples of 3D Delaunay triangulation use, such as surface reconstruction [ABK98], and an overall idea how these structures can be constructed.

Polygon triangulations

This topic is important for GIS and cartographic applications more than for a 'pure' computer graphics but contains the important and widely used algorithm of ear cutting [ORo98] and again a good example how much easier life can be with monotonous than with a simple polygon.

Intersections

Effective intersection algorithms are very useful computer graphics tools. We can show to the students a typical output-sensitive algorithm, the sweeping algorithm for polygons intersection [ORo98] and to confront an input configuration where this algorithm is very useful to some where it is useless. We can also introduce a concept of sublinear algorithms here - an algorithm for convex polygon intersection based on a bisection [CLM06]. Besides others, the intersection topic relates to the duality transformations which can be quite new and inspiring for the students. Dual-based algorithms have many computer graphics applications, namely in fast intersection detections.

Motion planning

In the period of animations and virtual reality, this 'robotic' topic is not out of scope of computer graphics any more as these algorithms may serve also for path planning in the virtual reality applications. Some fundamentals (a point robot, a disk and a ladder in 2D) [ORo98], [dBv*97] are enough to show such 'exotic' tools as Minkowski sums, visibility graphs etc. Also, a relation between graph and geometry methods should be shown and pointed out.

Trends, tasks and news in computational geometry

This topic is an opportunity to sum up what is general and useful on the computational geometry methods, what are typical features of this discipline, its vices and virtues. Tasks for computational geometry, both in research and in communication with other areas, can be pointed out. It is also useful to show fresh trends and hot topics – in our opinion, data stream algorithms [Mut03], sublinear [CLM06], in-place and in situ algorithms [BIK*04], parallelization, kinetic data structures [Gui04] can be understood as examples of current hot topics.

A proper book for this course is, first of all, [ORo98], as it is written in a less theoretical way and is concentrated on algorithms and their implementation. Useful sources are also [dBv*97], [Las96]. The bible of this science, [PS85], should be used carefully as it is for technical students quite demanding reading. Good materials are also those concerning design of algorithms [Ski98], [DAD] or geometric algorithms, books, Web sites, downloadable software and people working in this area [Epp], [DSG], [Sno], [DS]. There is no lack of materials for the area of geometric algorithms and with some care (not to include too difficult reading for an applied audience), many nice sources can be found.

3. Seminars

Let us address the content and form of seminars suitable for this course. As the training of students is oriented to the design and analysis of algorithms, it is better to have the seminars in a classical classroom without computers, in groups up to 20 students. In most cases students are given some geometric task concerning the current topic and are expected to propose an algorithm of their own for the given task. Examples of such problems are: an effective point-inside-polygon test; an algorithm for an approximate convex hull of a planar set of points; an algorithm for construction of a tangent line to a convex polygon from a given point or for construction of a common tangent line of two polygons. Tasks should be rather simple.

The students work independently or in small teams. The teacher spends the time of the seminar by spinning around the students and inspecting the just-being-born algorithmic ideas of students in their hand notes, lets them to explain how their algorithm works and supports their feeble and sometimes slightly crazy original ideas. When the students are more or less finished, interesting (and not necessarily correct and complete) ideas are presented by their authors to

the rest of class. This method has several advantages: the student is pleased that he/she developed something new and is encouraged to future creative attempts, learns to present his/her ideas 'on stage', learns to explain clearly an algorithm and to defend it from the classmates if necessary. Even shy students find some courage to show off when they see presentations of their classmates - they see that nobody is perfect. In the other role of the audience, the students learn to listen, to notice eventual mistakes and to ask sound questions. The teacher has to tailor the discussion in case the presenting student is too much 'beaten' by the classmates, who have found some error in his/her algorithm. At the end, the teacher should formulate some conclusions about the presented solution and concludes the debate.

This rather old-fashioned style of seminars is for such a course very useful and to be run successfully needs good experience of the teacher and his or her good promptness as in all cases the teacher needs immediately respond to the student's rudimentary algorithms and to regulate the student's progress if necessary. A noticeable algorithmic knowledge and experience is needed to react quickly and give advice. And, last but not least, the teacher should be a good psychologist to keep the students working and to control the presentation and discussion. Results can be very interesting as the students are by permanent 'attacks' of the teacher provoked to think out something of their own, and for some of them it is brand new experience.

This type of work can be combined with the students' presentations of their assignements, algorithmic analysis and other less demanding forms; however, the algorithmic synthesis should be the main point.

4. Student projects

In their free time, students also solve some small projects as is usual in such a type of course. We have good experience with three main rules: 1. versatility of forms, 2. freedom to choose the topics from the wide offer according to the personal taste, 3. competition.

Versatility means that besides classical project of the form 'propose an algorithm for this task, implement it, make some experiments with it and formulate conclusions from the results', the students can pick other forms, such as presentations, algorithmic analysis, algorithmic synthesis without implementation, production of nice pictures, reviews of some older work of their predecessors, etc. It is also useful if instead of an abstract formulation of the problems, such as 'implement a convex hull algorithm', the tasks for students are packed into some attractive cover (e.g., in the style of a fantasy literature) or have a witty formulation from which the student has to derive what to do (a good example is the ACM programming contests problems formulation, see [ACM06]). An unusual form of the project work is the 'review' where the student should check an algorithm, implementation and documentation of somebody else. The student is asked to make it anonymously and the resulting review is provided to the author of the original project. The 'reviewer' has to be informed in advance that that reviewed project has been already checked and evaluated by the teacher, and so the review has no impact on evaluation of the project author. After such an explanation, the students have no moral problem due to their solidarity. Usually their reviews are very detailed and punctual and provide an interesting feedback to the author, a feedback from the same age group. (An interesting experiment was to allot a work of a 'mathematical' student to the 'computer' one and vice versa: the students from different universities and specializations have a bit different style, they appreciate and criticize different things. Reading of such a review is very valuable even for the teacher.)

Freedom means that the student can pick from many possible topics according to his/her taste. This point is closely connected with versatility: some students like to present and do not like to program and vice versa, so they can personalize the course duties according to their preferences. It is surprising that at the end most students choose the classical projects (to propose and implement something) because 'they know what they can expect of it' while some exotic forms, such as reviews or presentations of some scientific paper, are too novel for them and therefore threatening. Students can choose as many small assignments as they want; easier have lower evaluation and vice versa. There is of course some required lowest possible sum of points to get credits for the course.

Why a student should choose more work than absolutely necessary? The answer is a *competition*: At the end of the course, one or more students with the highest number of points get credits for the course without an exam – the 'exam without exam'. Also the most active student on the seminar, who collected the highest number of points from his work at the seminars, gets the exam. Even without this attractive price, the students, namely male, love competition; they just want to be better than their classmates and they are able to devote an extra work to win. If the same work load was allotted by the teacher, they would complain, but if they take it deliberately, it is something different. Usually the winner(s) make so much work that a preparation to the exam would be much easier but it would not be such a fun for them!

Some results achieved by the students when solving their training problems are quite interesting either as a good piece of implementation work (see examples of students' outputs in Figs.1-3) or as a new algorithm which deserves to be published on the international level. It is surprising that by a small modification of a well known and well solved problem, we come to the problems which has not been solved so often and are not solved to a complete satisfaction yet. Examples are a well-known problem of the convex hull of a set of planar points modified to an approximate convex hull,

the Delaunay triangulation problem modified to the Delaunay triangulation of moving points and many others. Some of students got so much attracted by such problems that they continued to work on them even after passing the course and later entered our research team.

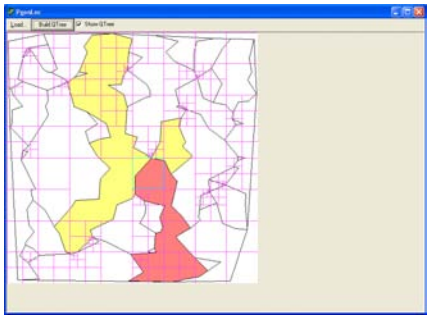

Figure 1: *A point location example - a planar subdivision together with a quadtree used to speed up a point location.*

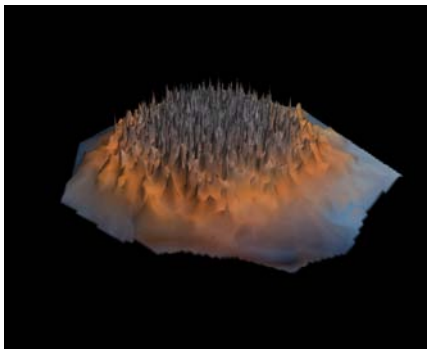

Figure 2: *A triangulation example - a fantastic landscape constructed on the Delaunay triangulation.*

5. Experience with this course and feedback from the students

Weak point of our experience with this course is a small number of students taking it; maximally 15 students on one faculty per one iteration. The reasons are different on both faculties: on the technical one the number of participants corresponds approximatelly to the number of students specialized on computer graphics each year, on the mathematical one the course is taught in a very impractical scheme for students but the only possible for the external teacher - 6 hours once in two weeks. On the other hand, the described system of seminars and project work has been verified also in different courses with up to 40 students per iteration. In all cases, the described system was run without a teaching assistant.

Figure 3: *A Voronoi diagrams example - a mosaic created from the Voronoi diagram of a bitmap.*

In our experience, there is no important difference in teaching this course in technical and mathematical faculties; the better mathematically educated students from the mathematical faculty feel this course as an applied one while the computer students from the technical faculty as a theoretical one, but there are excellent as well as $\cdots$ less excellent students in both groups. The students from the 'mathematical' group have fewer problems with complexity estimates, mathematical background and are not so curious on applications as the 'computer' group but all this can be expected and brings no surprise.

Problems have been encountered in case of several cartography students who have not been fluent enough in algorithms and programming; the students did not fail but needed too much time to debug their implementations and did not feel comfortable among computer graphics students.

The reader might also be interested whether there is some difference between boys and girls in their attitude to the described course, to its contents, to the type of selected projects and namely to the competition part of the work. As computer graphics in its technical and programming part is an area demanding on space imagination and sometimes close to hardware, it is usually not too attractive carrier for women. On the other hand, exactness and mathematical formalism of computational geometry, its stress to elegant algorithms fit well to many women, including the authoress of this text.

As the described course is taught mainly for students specialized on computer graphics, there is typically maximally

one girl and it is impossible to make any general conclusion. Our observation is that girls who are 'brave enough' to study computer graphics are usually strong in mathematics and programming, and have no problems with computational geometry course, often belonging to the best students in the course. No observations valid for more of them about their choice of topics, preferences to presentation or programming, relation to competition were collected.

Let us address the question how much the students like such a course. We could boast with the high level of students satisfaction with this course according to their anonymous feedback but we think this kind of feedback is a bit misleading: does it mean that the teacher is good? Or not demanding enough?

Therefore, we prefer to present some interesting points from an anonymous feedback to show how the students like such a type of course. Students usually appreciate a competition, although they know that they devoted to the coursework plenty of extra work due to their effort to win. They like a possibility to develop their own algorithms and it is surprising that for many of them it is a new experience; one would expect that it is a day-to-day life for computer graphics students. Some of them (not all!) like a possibility to present their ideas to public and to hear the ideas of their colleagues. Students appreciate that they are not pushed to memorize algorithms but to develop their thinking. They prize that the methods presented in the course can also be used for other problems. This feedback was collected anonymously, after the students had passed the exam.

6. Conclusion

Although computational geometry is not yet recognized as an important part of the computer graphics curriculum, it brings many useful methods to computer graphics community. In order to be able to use these methods, computer graphics experts need to be trained to it. In this paper we presented a computational geometry course suitable for an advanced undergraduate computer graphics students. Although such a course does not need an expensive hardware and software, it delivers fine tools to the hands of computer graphics people. It helps to develop an abstract thinking and to improve understanding of general and repeating aspects of computer graphics problems. Therefore, we can recommend such a course to be included in the computer graphics education.

References

[ACM06] *ACM Programming Contest Archive* http://www.acm.inf.ethz.ch/ProblemSetArchive.html

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new Voronoi-based surface reconstruction algorithm. In *Proc. Siggraph'98* (1998), pp.415–421.

[Ang97] ANGLADA, M.V.: An Improved Incremental algorithm for constructing restricted Delaunay triangulations. *Computers & Graphics 21*:215-223,1997

[App96] *Application Challenges to Computational geometry*. Computational Geometry Impact Task Force Report. TR-521-96, Princeton University, April 1996.

[BCF*06] BOURDIN J.J., CUNNINGHAM S., FAIRÉN M., HANSMANN W.: Report of the CGE06 Computer Graphics Education Workshop, Vienna, Austria, 2006 http://education.siggraph.org/conferences/eurographics/2006/cge2006

[BIK*04] BRÖNNIMANN H., IACONO J., KATAJAINEN J., MORIN P., MORRISON J., TOUSSAINT G.: Space-efficient convex hull algorithms. *Theoretical Computer Science 321(1)*:25-40, 2004.

[CGAL] *CGAL library* http://www.cs.ruu.nl/CGAL

[DAD] *Dictionary of Algorithms and Data Structures*. National Institute of Standards and Technology http://www.nist.gov/dads

[dBv*97] DE BERG M., VAN KREVELD M., OVERMARS M., SCHWARZKOPF O.: *Computational Geometry. Algorithms and Applications.* Springer-Verlag,1997.

[deB97] DE BERG M.: Trends and developments in computational geometry. *Computer Graphics Forum 16(1)*:3-30, 1997.

[DS] *Downloadable Software*. The Geometric Center, University of Minnesota http://www.geom.umn.edu/software/cglist

[CLM06] CHAZELLE B., LIU D., MAGEN A.: Sublinear geometric algorithms. in:*Sublinear algorithms*, Dagstuhl Seminar Procedings, Germany, 2006.

[Dob92] DOBKIN D.P.: Computational geometry and computer graphics. *Proceedings of the IEEE*, Vol 80, 9:1400–1411, 1992.

[DLR90] DYN N., LEVIN D., RIPPA S.: Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis* 10:137-154,1990.

[DSG] *Dan Sunday's Geometry Algorithms* http://geometryalgorithms.com

[Epp] EPPSTEIN D.: THE GEOMETRIC JUNKYARD http://www.ics.uci.edu/~eppstein/junkyard/triangulation.html

[Gui04] GUIBAS L.J.: Kinetic data structures. In: *Handbook of Data Structures and Applications* D. Mehta and S. Sahni, Eds, Chapman and Hall-CRC, 2004.

[KF01] KOLINGEROVÁ I., FERKO A. Multicriteria optimized triangulations. *The Visual Computer 17 (8)*:380-395, 2001.

[Las96] LASZLO M.J.: *Computational Geometry and Computer Graphics in C++*. Prentice Hall, 1996

[LO06] LAXER C., ORR J.: A Knowledge Base for the Emerging Discipline of Computer Graphics. Report of the SIGGRAPH Education Committee Curriculum Working Group, 2006 http://education.siggraph.org/conferences/eurographics/2006/CKBreport_CGE06.pdf

[Min] MING C.L. COMP290-72, Computational Geometry and Applications http://www.cs.unc.edu/~lin/COMP290-72

[Mut03] MUTHUKRISHNAN S.: Data streams: Algorithms and Applications. In:*Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* , Baltimore, Maryland, pp.413-413, 2003.

[ORo98] O' ROURKE J.: *Computational Geometry in C (2nd edition)* Cambridge University Press, New York, 1998.

[PS85] PREPARATA, F.P., SHAMOS, M.I.: *Computational Geometry: An Introduction* Springer Verlag New York Berlin Heidelberg Tokyo, 1985.

[She96] SHEWCHUK J.R.: Robust adaptive floating-point geometric predicates In: *Proceedings of the 12th Annual Symposium on Computational Geometry*, pp.141-150, 1996.

[Ski98] SKIENA S.S.: *The Algorithm Design Manual*, Telos/Springer Verlag, 1998.

[Slo93] SLOAN, S.W.: A fast algorithm for generating constrained Delaunay triangulations. *Computers & Structures* 47:441-450, 1993.

[Sno] SNOYEING J.: Computational Geometry Demos http://www.cs.ubc.ca/spider/snoeyink/demos/home.html

[Zar05] ŽÁRA J.: Virtual reality course - a natural enrichment of computer graphics classes. In: *Eurographics 2005 - Educational Papers*, Dublin, Ireland, The Eurographics Association, pp.25-32, 2005.