GEOMETRIC COMPUTING IN THE UNDERGRADUATE COMPUTER SCIENCE CURRICULA

John L. Lowther and Ching-Kuang Shene
Department of Computer Science
Michigan Technological University
Houghton, MI 49931-1295
(john|shene)@mtu.edu

Abstract

Geometric computing is a rapidly evolving interdisciplinary field involving computer science, engineering and mathematics. It has relationships to many other areas within computer science such as computational geometry, computer-aided design, computer graphics, computer vision, geometric/solid modeling, robotics, and visualization. Due to its interdisciplinary nature, geometric computing also serves as a vehicle for engineering students to approach important topics in manufacturing processes (*e.g.*, NC-machining, molding, and milling), geometric design/modeling (*e.g.*, feature-based design and constraint solving), and computational metrology (*e.g.*, sampling strategy and tolerancing design). Unfortunately, in a typical computer science curriculum, computing with geometry is virtually missing in spite of its tremendous impact to computer science and the other fields mentioned above and its importance to increase students' employability.

This paper investigates this issue and suggests a possible remedy by designing an elementary and interdisciplinary intermediate level geometric computing course. The rationale, goals and objectives, and the impact and significance of the proposed course are discussed. A review of previous work, including textbooks, software systems, and other media materials, is included. Finally, after presenting the merit of designing this course, this paper proposes a sample ten-week syllabus with software modules for lab use.

1. INTRODUCTION

Geometric computing is a rapidly evolving interdisciplinary field involving computer science, engineering and mathematics. It has relationships to many other areas within computer science such as computational geometry, computer-aided design, computer graphics, computer vision, geometric/solid modeling, robotics, and visualization. Due to its interdisciplinary nature, geometric computing also serves as a vehicle for engineering students to approach important topics in geometric and mechanical design, manufacturing processes (*e.g.*, NC-machining, molding, milling, cutting, stereolithography, and photochemical machining), geometric design/modeling (*e.g.*, feature-based design and constraint solving), and computational metrology (*e.g.*, sampling strategy, tolerance checking and statistical tolerancing). Since geometric computing deals with geometric objects, geometry and topology have already provided a solid foundation and many techniques.

Unfortunately, in a typical computer science curriculum, computing with geometry is virtually missing in spite of its tremendous impact to computer science and the other fields mentioned above and its importance to increase students' employability. In this paper, we shall investigate this issue and suggest a possible remedy by designing an intermediate level interdisciplinary geometric computing course that addresses this problem. We shall consider software support for this course as well.

In this paper, Section 2 discusses the rationale of introducing such a course. Section 3 presents the course's goals and objectives. Section 4 addresses its potential impact and significance. Then, Section 5 reviews existing work that is related to our work, including textbooks, software systems and other media materials. This is followed by a discussion of our course design in Section 6, including the merit of our design (Section 6.1), a brief sample of a ten-week course syllabus (Section 6.2) and possible software modules for lab use (Section 6.3). Finally, Section 7 has our conclusion.

2. RATIONALE

A geometric computing course is important to computer science and should have a position in computer science curricula. The following sections will focus on several issues, namely: recommendations by renown computer scientists on adding continuous computation back to computer science curricula (Section 2.1); the fundamental role geometric computing plays in the new emerging manufacturing technology and which has been identified as vital to this country in order to regain the competitiveness in the global economy (Section 2.2); the opportunity a geometric computing course provides for consolidating many existing but scattered materials in a coherent way (Section 2.3); and, due to its interdisciplinary nature, the foundation a geometric computing course provides to students in many departments/disciplines (Section 2.4).

2.1 Recommendations from National Research Council

Over the past decades, computer science curricula have been changed emphasizing their discrete nature. Non-discrete topics have been gradually shifted to other disciplines. The *ACM/IEEE Computing Curricula*[1] is a good example in which only very limited hours are spent on computing with the continuum, although it does manage to preserve some basics of symbolic and numeric computing. This trend has also been observed by the Computer Science and Telecommunications Board and National Research Council:[2]

> *Moreover, as discrete mathematics (e.g., logic, set theory, graph theory) has found its way into the CS&E curriculum, continuous mathematics (e.g., calculus, differential equations, statistics) has been slighted. This is unfortunate, because continuous mathematics is essential in important subfields in CS&E such as performance analysis, computational geometry, numerical analysis, and robotics. Furthermore, continuous mathematics is the language of many scientific and engineering fields, and an adequate understanding of continuous mathematics is needed to approach computing applications in such areas with confidence.*

2.2 Geometric Computing is a Foundation of the New Manufacturing Technologies

---

[1] Allen B. Tucker (editor), *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*, ACM Press, New York, 1991.

[2] Juris Hartmanis and Herbert Lin (editors), *Computing in the Future: A Broader Agenda for Computer Science and Engineering*, National Academy Press, Washington, D.C., 1992.

Recent geopolitical factors have led to a shift from defense-based industries to consumer-based production. As a result, manufacturing has been recognized as a priority for US competitiveness in the new global economy.[3] Although the computing communities are paying increasing attention to manufacturing problems, the pace of this work is not sufficient for this country to meet the demands. Friedman, Glimm and Lavery[4] listed six emerging manufacturing technologies of which intelligent machines and solid modeling are basic technologies that underlie other emerging manufacturing technologies.

Geometry is a key component of manufacturing. It enters at two distinct levels: in the design of the manufacturing *process* and in the manufacturing of *products*. Although computer science students might not directly participate the work of designing and manufacturing products, they will be part of the underlying software development process.

2.3 Geometric Computing Knowledge Are Scattered in Many Courses

Computer science curricula do have some basic elements of geometric computing scattered throughout many courses and lacking a coherent point of view. For example, in calculus, students would learn some curve/surface representations (*e.g.*, parametric and implicit forms), continuity, and curvatures. Other representations such as procedural representations (*e.g.*, fractals) and polyhedra only appear in a computer graphics course. An application oriented computational geometry course may introduce to the students the way of representing and generating a polyhedron (*resp.*, polyline) from a surface (*resp.*, curve) and some other tessellation techniques (*e.g.*, Voronoi diagrams and Delauny triangulations).

There is a more serious problem regarding accuracy and robustness of algorithms. Except for some introductory material or in a numerical analysis/methods course, students might never learn what will happen when a real world object, which is usually part of the continuum, is converted to a discrete computer representation. Students may never know the impact of finite precision to algorithms dealing with geometric objects. As a key example, writing a program for solving a quadratic equation that could handle extreme cases without losing significant digits might not be as easy and as trivial as one may expect at a first glance.[5]

Therefore, a coherent way must be found for organizing these missing or scattered but rarely taught materials. A geometric computing course seems a very reasonable choice.

2.4 Bridging the Gap Among Various Departments

---

[3] Chee Yap, *Report on NSF Workshop on Manufacturing and Computational Geometry*, Department of Computer Science, New York University, January 1995.

[4] Avner Friedman, James Glimm and John Lavery, *The Mathematical and Computational Sciences in Emerging Manufacturing Technologies and Management Practices*, SIAM, Philadelphia, PA, 1992.

[5] Forman S. Acton, *REAL Computing Made REAL*, Princeton University Press, 1996.

Computer science focuses on the "algorithmic" or the "computational" aspects of geometric computing, mathematics supplies the necessary foundation, and engineering disciplines offer applications. Although different disciplines may approach the same problem with different viewpoints, the fundamental geometric aspects are all the same. Therefore, designing an intermediate
level geometric computing course would help students in different disciplines to consolidate fundamental geometric knowledge and skills. For example, computer science students would learn the way of handling geometric problems, mathematics students would know the application and computation aspects of their geometric knowledge, and engineering students would understand the underlying geometric nature and skills for their design and product manufacturing process.

Rather than separated courses, a single course in which all important issues are covered, including basic concepts, fundamental skills, computational techniques and applications in various fields, would be more helpful and save time and effort. In such an interdisciplinary course with students coming from various departments, cross discipline discussion will not only inject ingredients into the course, but also generate new research problems and projects. After taking this course, equipped with interdisciplinary knowledge of geometric computing, students will be able to handle their future study within their home department in a more flexible and creative way.

3. GOALS AND OBJECTIVES

The primary goal of this on-going project is to design a comprehensive and elementary geometric computing course to address interdisciplinary issues for students in computer science, engineering and mathematics. A secondary goal is to develop accompanying software systems for students to visualize and to experiment various concepts in geometric computing. To achieve these goals over the next few years, the following six-fold objectives are planned:

1. Design course material for an elementary and comprehensive course for computer science, engineering, and mathematics students.
2. Design and implement lab programs and experiments illustrating important concepts (*e.g.,* loss of significance digits, numerical instability, and robustness of computation algorithms) and visualizing and interrogating geometric objects.
3. Initiate a cross disciplinary research/education effort to absorb other issues and application areas to further enhance this course. Some possibilities include applications in computer-aided design and terrain modeling in geographical information systems (GIS).
4. Design a number of interactive animation sequences showing the fundamental concepts of geometric computing. These will include, but not be limited to, losing significant digits in floating number computation and its impact to the robustness of geometric algorithms, tracing a curve/surface interactively, showing the curvature, inflection, umbilic, and other geometric characteristics of a curve/surface.
5. As a long term effort, produce several video programs so that some of the most time consuming computation and the most complex concepts will be presented to our students as well as educators/students in other institutions. These topics include spline curve and surface design, knot insertions, degree reduction/elevation, and characteristics of implicit surfaces (*e.g.*, singularities, curvature computation, curve tracing, and surface intersection), and other

topics.

6. The ultimate goal is to design a virtual reality system for students walking into the virtual world so that they could gain direct experience of the critical geometric and topological concepts to further enhance their understanding of geometric computing.

4. IMPACT AND SIGNIFICANCE

Given the current trend of 3D computer graphics and the need of geometric computing skills in manufacturing, the addition of an intermediate interdisciplinary geometric computing course to computer science curriculum will have an important impact. It will not only address the need of other areas but also expand the knowledge of computer science students. As a result, the problems and recommendations of National Research Council are addressed and computer science curricula will be improved.

Such a geometric computing course will improve students' understanding of the use and the theory of computers in the non-discrete world. It would also consolidate some basic knowledge in computer science (the algorithmic and computational aspects), mathematics (geometry and topology), and engineering (design, manufacturing, and robotics). Although it is difficult to estimate the number
of universities where this course might be adopted, there will be a high possibility that some universities would shift their current course offerings by incorporating some topics from our geometric computing course.

The innovative part of our course's software systems involves providing to the students an interactive environment so that they can visualize many fundamental concepts and carry out trial-and-error experiments. There are programs available that illustrate some of these concepts; however, these programs either address only a few issues and are not up to our expectation or have a totally different focus (Section 5.2). High powered computer-aided design and computer aided geometric design systems, although good for real world designers, may cost too much for an education environment and may not permit student experimentation. Moreover, these systems in general do not have pedagogy as their main concern. Therefore, our courseware and interactive animation systems will be working examples to other institutions. This, in turn, will stimulate the development of pedagogical software exploring other aspects of geometric computing.

Although our effort may be a small step toward the right direction, its impact on adding continuous mathematics topics to computer science curricula, promoting interdisciplinary courses, and consolidating materials from various disciplines may be significant in the foreseeable future.

5. PREVIOUS WORK

This section reviews previous works in three categories: namely, textbooks and course materials, system systems, and other media materials.

5.1 Textbooks and Course Materials

During the past decade, there have been a number of textbooks published to address different aspects of geometric computing.[6] These are excellent senior and graduate level textbooks focusing on specific topics (*i.e.*, design and manufacture, CAGD, geometric or solid modeling). There are some computational geometry textbooks as well,[7] focusing on the theory/algorithmic side of discrete geometry with an emphasis on computational complexity issues.

The new wave of texts are more accessible than the previous ones, although they still are not up to our expectation. Of these new textbooks, Boehm and Prautzsch[8] provides a complete view of the necessary geometric background for geometric design. Fiume[9] covers both symbolic and numerical computation; however, the geometric content which forms the fundamental part of geometric computing is at best minimal. Laszlo[10] and O'Rourke[11] focus on the computational geometry. Finally, Bowyer and Woodwark[12] more or less addresses our need. To be used as a comprehensive elementary textbook, the following two shortcomings must be addressed: (1) its focus still is too narrow (*i.e.*, computer-aided design related), and (2) more materials such as representations of shapes, algorithm robustness, symbolic and algebraic computations must be added.

5.2 Software Systems

Designing software, especially animation systems, for course work has been very popular during the past decade. Ever since BALSA,[13] there have been many important and general algorithm

---

[6] Richard H. Bartels, John C. Beatty, and Brian A. Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann, 1987; Gerald Farin, *Curves and Surfaces for Computer Aided Geometric Design*, third edition, Academic Press, 1993; I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture*, Ellis Horwood, Chichester, England, 1979; Christoph M. Hoffmann, *Geometric & Solid Modeling: An Introduction*, Morgan Kaufmann, 1989; and, Michael E. Mortenson, *Geometric Modeling*, John Wiley & Sons, 1985.

[7] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987 and F. P. Preparata and M. I. Shamos, *Computational Geometry: an Introduction*, corrected and expanded second printing, Springer-Verlag, 1988.

[8] Wolfgang Boehm and Hartmut Prautzsch, *Geometric Concepts for Geometric Design*, A K Peters, 1994.

[9] Eugene Fiume, *An introduction to Scientific, Symbolic, and Graphical Computation*, A K Peters, 1995.

[10] Michael J. Laszlo, *Computational Geometry and Computer Graphics in C++*, Prentice Hall, 1996.

[11] Joseph O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1994.

[12] Adrian Bowyer and John Woodwark, *Introduction to Computing with Geometry*, Information Geometers, 1993.

[13] Marc H. Brown and Robert Sedgewick, A System for Algorithm Animation, *Computer Graphics*, Vol. 18 (1984), No. 3 (July), pp. 177-186.

animation systems, Zeus,[14] XTANGO[15] and POLKA/SAMBA, and Mocha[16] being the best examples. In addition to these general ones, there have been many others as reported in volumes of *ACM SIGCSE Technical Symposium on Computer Science Education*. See Price et al[17] for a complete list up to 1993. Moreover, *Journal of Visual Languages and Computing* contains more advanced uses of visual and animation systems, and *Journal of Parallel and Distributed Computing* has a special 1993 issue on visualizing parallel programs. None of these systems dealt with geometric computing.

Systems dealing with geometric objects do exist. For example, the Geometry Center at University of Minnesota maintains a collection of excellent geometric software packages for professional uses or for visualizing the geometry of some interesting geometric objects rather than teaching aids for a geometric computing course. One recent addition to the above list is Rockwood and Chamber.[18] This is a Windows-based multimedia tutorial, with a special focus on Bézier/B-spline curves and surfaces.

5.3 Other Media Materials

There is no shortage of video programs for education purpose. Most of these programs address one or two topics in an introductory level[19] or a collection of animations illustrating certain novel ideas (*e.g.*, ACM SIGGRAPH has published many selections annually). The *ACM Annual Symposium on Computational Geometry* video review usually includes novel animation video programs illustrating the discrete aspects of geometric algorithms. *Not Knot*[20] and *Outside In*[21] are two excellent video programs produced by Geometry Center. Although these programs would certainly serve as a model for our work, they do not address the need of a geometric computing course.

---

[14] Marc H. Brown, Zeus: A System for Algorithm Animation and Multi-View Editing, *1991 IEEE Workshop on Visual Languages*, October, 1991, pp. 4-9.

[15] John T. Stasko, TANGO: A Framework and System for Algorithm Animation, *IEEE Computer*, Vol. 26 (1990), No. 9 (September), pp. 27-39.

[16] James E. Baker, Isabel F. Cruz, Giuseppe Liotta and Roberto Tamassia, A New Model for Algorithm Animation Over the WWW, *ACM Computing Surveys*, Vol. 27 (1995), No. 4 (December), pp. 568-572.

[17] Blaine A. Price, Ronald M. Baecker, and Ian S. Small, A Principled Taxonomoy of Software Visualization, *Journal of Visual Languages and Computing*, Vol. 4 (1993), pp. 211-266.

[18] Alyn Rockwood and Peta Chambers, *Interactive Curves and Surfaces: A Multimedia Tutorial on CAGD*, Morgan Kaufman, 1996.

[19] University Video Communications, Films for Humanities and Sciences, and NOVA have published many one-hour programs.

[20] Geometry Center, *Not Knot*, Jones and Bartlett, Boston, 1991.

[21] Geometry Center, *Outside In*, A K Peters, 1994.

Knotty[22] is the only video program that partially addresses our need. The major contribution of this program is a rather complete animation of important characteristics of B-spline curves. Even though the surface part is weak, it could serve as a starting point for our course. In fact, these video programs do not require high-end technology to produce. The thrust behind these programs is usually careful planning, a good story board with colorful and flashy graphics, and, more importantly, labor investment.

6. COURSE DESIGN

6.1 Design Merit

Organizing a course on geometric computing and preparing a well-orchestrated and inspiring set of undergraduate class room notes is a challenging task. But it is not an impossible mission. At the turn of this century, Felix Klein wrote many short texts for German high school teachers and students explaining to them advanced mathematical concepts. Hilbert and Cohn-Vossen published one the best intuition based geometry lecture notes for German undergraduates, *Geometry and the Imagination*,[23] in which advanced geometric concepts are all treated with elementary and intuitive geometric arguments. These great works inspired our ambition of designing an undergraduate geometric computing course based on essentially the same principles. Modeling our lectures after these great works would not only enforce us to rethink and to consolidate the current state-of-the-art research, but also to embark on a new way of teaching geometric computing.

In a recent inspiring article, Schank, Korcuska and Jona[24] warned that there exists a hidden danger in the commercial juggernaut of current-generation multimedia software. This danger is simply that these systems often employ a "page-turning architecture," and as a consequence students only learn what Alfred Whitehead called "inert knowledge."[25] To address this problem, Schank et al suggested that multimedia systems can help in four ways: (1) building goal-directed learning systems, (2) making software failure-driven, (3) making software case-based, and (4) learning-by-doing.

Based on the above rationale, our course will address the fundamentals of geometric computing with an intuitive and elementary approach by emphasizing the geometric nature of the subjects and leaving the complexity of algebraic derivations and computations to software systems. Emphasizing the geometric nature does have its merit. Most of the algebraic derivations and computations are too complex or too difficult to be taught in an intermediate level course. However, students could still

---

[22] Jonathan Yen, *Knotty: A B-Spline Visualization Program*, Morgan Kaufmann, San Francisco, 1993.

[23] David Hilbert and S. Cohn-Vossen, *Geometry and the Imagination*, translated from the German edition *Anschauliche Geometrie* by P. Nemenyi, Chelsea, New York, 1952.

[24] Roger C. Schank, Michael Korcuska and Menachen Jona, Multimedia Applications for Education and Training: Revolution or Red Herring? *ACM Computing Surveys*, Vol. 27 (1995), No. 4 (December), pp. 633-635.

[25] Inert knowledge is not connected to the situations in which that knowledge would be useful and so is ultimately forgotten.

learn the fundamentals of geometric computing and, with the help of this basic understanding, they can pick up the necessary derivation part in later courses easily. Even though the students may stop after taking our course, they have already acquired enough fundamental knowledge so that they will feel confident when facing geometric computing problems in their career.

Through the use of the interactive animation systems, students will be able to (1) visualize the geometry, (2) understand the impact of finite precision, (3) experiment with geometric concepts, and (4) discover deeper and subtle properties. In particular, some topics will not be covered in class and students are encouraged to discover their uses and properties through lab experiments. Our systems will be designed to address all four suggestions made by Schank et al, although some are not so obvious (*e.g.*, how to make software failure-driven) and further research is required during the course of software development.

6.2 Course Contents

Our course is a 3-credit introduction to geometric computing course for sophomore and junior students in computer science, engineering and mathematics disciplines. This course includes guest lectures. The invited speakers are leading figures in geometric computing. A course syllabus for a 10-week quarter follows. It can easily be made into a 15-week semester syllabus. For example, contents for Week 5, 6, 7 and 8 can be expanded easily and robust (Week 9) and tessellations (Week 10) are also good candidates for expansion.

The laboratory part involves the use of the software modules that are currently being designed. Examples of these software modules are described in the next section.

Week 1: Week 1 motivates students and introduces fundamental concepts. These include an overview of the field and its applications, the "theme" of this course (*i.e.*, Geometry Representation Algebra Algorithm Program), complexity of geometric problems (*i.e.*, dimensional, geometrical, and combinatorial), and computing with REALs.

Week 2: This week reviews some geometric concepts to be used in later weeks. Topics include coordinate systems, points, lines, and planes, simple curves and surfaces, homogeneous coordinates, Euclidean transformations (*i.e.*, translations, rotations, and reflections), and affine and projective transformations.

Week 3: Representations of geometric objects are the major topics of this week. The wireframe model, polyhedron model, boundary representation, constructive solid geometry, and sweeps will be covered.

Week 4: Parametric curves and surfaces will be covered in Week 4. Major topics are polynomial curves, rational curves, continuity (*i.e.*, tangential, curvature and geometric), and parameterizations (arc-length and non-uniform speed).

Week 5: Parallel to parametric curves, Week 5 addresses parametric surface patches.

Week 6: Part of Week 5 and Week 6 are dedicated to modern approaches of curve and surface design. Topics include Bézier curves/surfaces, the de Casteljau algorithm, B-spline curves/surfaces, and their important properties.

Week 7: Week 7 focuses on implicit curves and surfaces and their properties and uses. Point classification, level curves, tangent plane, normal, curvatures, and umbilic are example

topics.

Week 8: Week 8 will go deeper with implicit curves/surface, covering intersection computations, blending, offsetting, and their use in practice.

Week 9: After learning many topics in the previous weeks, Week 9 covers robustness issues, including problems with inaccuracy, imprecise geometric input, exact arithmetics, and robust algorithms design.

Week 10: Week 10 briefly covers the concept of tessellations such as grids, Voronoi diagrams, Delauny triangulations. If time permits, quad-trees and oct-trees will also be covered.

6.3 Software Development

The software system accompanying this course consists of the following five components:

MODULE-1: This module explains effect of losing significant digits and shows the impact of finite precision arithmetic on geometric computing.

MODULE-2: This module allows students to design and display polyhedra on the screen. Students can move the camera position and rotate the object about any one of the three coordinate axes.

MODULE-3: This parametric curve tracing system allows students to trace curves freely by varying a parameter. On the screen, the students can see the tangent vector, normal vector, bi-normal vector (*resp.*, the Frenet frame), curvature sphere, and inflection points. When the students are tracing Bézier and B-spline curves, the Bernstein/spline basis and the partition-of-unity characteristics will be shown. Students will also be able to see the effects of arc-length and non-arc-length parameterizations.

MODULE-4: This parametric surface tracing system permits students to see the tangent plane, normal vector, cross-section curvature, mean and Gaussian curvatures, isoparametric lines, umbilic points, singularities, and, if it is possible, geodesics and lines of curvatures. Students will be tracing the (u,v) parameter in the domain. When it is used for Bézier and B-spline surfaces, control points will be shown and allowed to be moved in order to change the shape of the resulting surface and to design a new one. Students will be asked to use this system to design their own favorite object, showing their understanding of the B-spline systems.

MODULE-5: This system is similar to the previous one, but for implicit surfaces. It will also be able to show the structure of an implicit surface, which is more difficult than the parametric part. An intersection curve tracing module is planned. Other operations such as intersection and offset may be performed with raytracing systems (*e.g.*, POV-Ray and Radiance).

Although students will design geometric objects and write some codes as lab experiments or homeworks, they are not writing the whole system. In fact, all mentioned systems will either let the students design the object directly through point-and-click technique or add their components through dynamic linking.

To produce video programs, pre-selected clips obtained from these modules will be saved in

digital format and later edited into several short sequences, one for each important concepts. A sound track will be added later when the digital format is converted to the conventional analog format on video tapes. At the beginning of this project, virtual reality will not be considered; however, it will be considered seriously in the later part of this project after the software modules are stabilized.

7. CONCLUSION

In previous sections, we have presented the rationale and design of an intermediate level introduction to geometric computing course for computer science, mathematics and engineering students. This course will be initially offered as a special topics course in computer science in the forthcoming Winter quarter. We have started course design and software development. A preliminary version of our software will be tested sometime this Fall and will be made available on the Internet to the public. Technical details of these systems will be presented elsewhere as well.

We believe that computer science students who are equipped with the basic knowledge and skills of geometric computing would have a better chance on the job market, especially at the time when new manufacturing technology, computer aided design and animation software development are entering a new era.

Students in engineering departments usually take engineering graphics in their first year and later computer aided design. Our course will also serve as intermediate step for these courses, especially for those students who want to know how their CAD system creates and manipulates geometric objects.

In addition to computer science and engineering students, our course may be very attractive to mathematics students as well. This course could provide a gateway for them to practice their geometric knowledge, to understand how finite precision computers process geometric objects, and to learn a skill in an applied and practical field for their career. With these basics in their hand, they could proceed to a practical computer aided design course or a more theoretical and fundamental geometric modeling course.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under grant DUE-9653244. The second author was also partially supported by the National Science Foundation under grant CCR-9696084, formerly CCR-9410707.