

# Re-Inventing the Introductory Computer Graphics Course: Providing Tools for a Wider Audience

Steve Cunningham\*  
Computer Science Department  
California State University Stanislaus  
Turlock, CA 95382 USA  
rsc@eos.csustan.edu  
<http://www.cs.csustan.edu/~rsc>

## Abstract

Traditionally, the introductory computer graphics course in computer science has focused on fundamental algorithms and techniques for creating images and animations. This was reflected in ACM/IEEE Curriculum 91 [TUC 91]. Computer graphics and similar subjects are expected to play a larger role in undergraduate computer science in the future [CUN 98], and this is being discussed in the ACM/IEEE Curriculum 2001 project. In the last few years the approach to teaching this course has changed to take advantage of more powerful graphics tools. This paper describes an approach to the introductory computer graphics course that increases its value as a tool for the student in the sciences, mathematics, or engineering as well as providing a sound introduction to the subject for the computer science student. This approach is compatible with the recommendations of the recent Graphics and Visualization Education Workshop [4] while focusing on serving an expanded audience.

## Introduction

Computer graphics has become an important tool for the working scientist, mathematician, and engineer. Its value is expanding as the sciences embrace visualization and as powerful computers become more and more commonly available, even on the desktop. However, the traditional computer graphics course focuses on fundamental graphical algorithms and processes, and is oriented towards the computer science student who wants to pursue advanced work in the field, not the student in science, mathematics, or engineering.

The graphics APIs now available allow us to refocus the first computer graphics course to place more emphasis on computer graphics programming. Using a high-level API opens computer graphics to any student with strong programming skills, and offers computer science an opportunity to serve science, mathematics, and engineering students by developing an introductory computer graphics course with dual goals. One goal is to introduce computer science students to basic geometric processes, graphics programming, and the graphics pipeline; the other is to give students in science, mathematics, and engineering a basic background in thinking and programming in 3D geometric terms.

## Developing the Course

The goals of the class as described above must be implemented when the course is offered. The first goal, introducing computer science students to basic graphics concepts and to graphics programming, is straightforward. We discuss the nature of modeling, of defining entities and their properties, of defining a scene, of viewing, and of rendering. We add issues of event-driven program design and of controlling programs with simple devices such as the keyboard and mouse, and we introduce simple animation. These are all supported by current APIs, and the student develops skills in programming graphics applications while learning about graphical concepts. This approach opens the beginning graphics course to a wide variety of new students, as described in [CUN 99].

The second goal of supporting science, mathematics, and engineering students (abbreviated as “science” in the remainder of the paper) is intended to give them the graphics tools that they will need for their fields [BRO 90]. As we describe later in this paper, the course will include projects that emphasize scientific concepts, so these students will not only learn computer graphics, but will also see how graphics can help them understand science.

Adopting these two goals has another consequence. By working with science students on interdisciplinary projects, computer science students will learn about communications and about the nature of applications. Each project will begin with a discussion about the underlying science and will close by asking for a conclusion about the science; in between students will need to understand how geometry is developed from a scientific problem, how that geometry is programmed to produce an image or animation, and how the graphic results tell something about the science. This in itself should be a valuable experience for an upcoming computer scientist, and of course is also the approach needed by the science student. The heterogeneous nature of the class offers opportunity for small-group projects with diverse groups of students, which can be challenging, but the benefits should outweigh any problems this might bring.

In order to serve this diverse audience, the course needs to include a collection of examples and projects that come from a broad set of scientific areas. We need examples and projects from mathematics, from physics, from chemistry, from biology, and from various branches of engineering. Creating a set of materials with this broad coverage will take some time, and the author has begun to work on such a project with the support of a grant from the National Science Foundation. These materials will be developed and tested in collaboration with other faculty having many different kinds of experience and with students from other disciplines. We expect to have a draft set of materials available by mid-2000 and a completed set a year from then. The materials will be distributed on the Web and will evolve as our understanding of the needs of science students grows.

The course outline is straightforward. It begins with an introduction to the concepts of geometry as used by OpenGL, and with simple examples of OpenGL programs to help students see how to build their first programming projects. It goes on to cover basic polygon-based modeling, including the use of instancing transformations; to discuss rendering including projections, viewing, lighting, and shading; and to work with event handling to provide user controls and animation. It ends with some features that students rarely actually use in a traditional introductory course, such as alpha blending, texture maps, and fog. All the work in the course is done in 3D, with almost no reference at all to 2D graphics. The use of OpenGL as the basic API supports both goals for the course and is consistent with current directions in graphics applications.

This course is only possible because of advances in low-cost and high-function graphics systems. Computers with these capabilities are now accessible for even budget-strapped universities. Graphics programming APIs such as OpenGL and Java3D are readily usable by students who have programming skills but are not computing specialists. Further tools should soon be widely available.

My students are enthusiastic about this kind of course and my science colleagues are helping to assemble science-related projects, expecting that their students will want to take the course.

## **Course Projects**

A key to making this course effective for science students is creating a good set of projects that are meaningful to the student. We need projects that fit the students’ scientific areas. The author is developing a set of projects for the course with this orientation, and plans to have a set ready to distribute by the start of the academic year in 2000.

In the first course offering, however, we did not have any of these projects. Instead, the course projects illustrated simple modelings and renderings in the sequence in

which materials are presented and in which students would develop their skills. These skills, however, proved to be significant when the course provided good API support. In the first offering, these projects were:

- a) create 3D histograms that rotate continually, with only ambient light
- b) model and render a physical object with one light source, where both the object and the light source can be rotated using keystrokes
- c) create and display a pseudo-fractal landscape with semi-transparent water and both snow and tree lines, adding menus to control aspects of the landscape
- d) display a randomly-chosen N-body gravity problem with each body showing a short-term trace of its recent location, with time-step animation
- e) define and display a Bézier surface with selectable and moveable control points

Each of the projects included user-controlled motions or some form of animation, and each rewarded the student for good modeling work by providing very rewarding images. A few examples of student work on these projects are shown below in images from the last three of the course's five projects. Because each project also involved animation and/or user interaction, however, these static images only hint at the quality of the work involved.

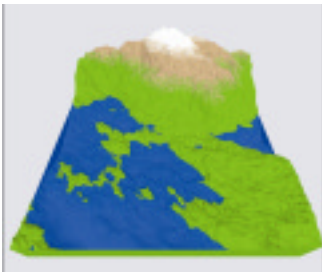


Figure 1  
pseudo-fractal landscape  
with alpha blending in water

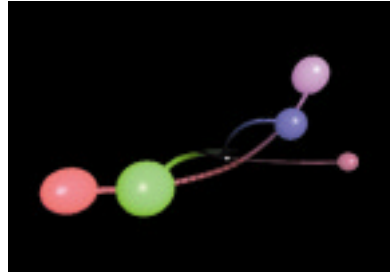


Figure 2  
n-body problem with  
fade-out trails on bodies

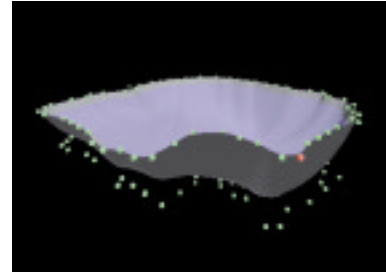


Figure 3  
Bézier spline surface  
with control points shown;  
one is selected for movement

In the first example, students use keyboard callbacks to move around the landscape and see it from different viewpoints, and use menu callbacks to change various features of the landscape. In the second example, the bodies move in real time driven by the idle callback, using simulated gravitational forces between the multiple bodies. In the third example, students move around the shape with keyboard control and can select and manipulate a control point with mouse callbacks and item selection, as shown; the shape of the surface changes interactively as the point is moved with keyboard control.

In the fully developed course, students will be able to choose the projects they work on so that their projects come from their own area of science. These projects will visualize aspects of their science and will reinforce their learning in that area. At the current time the author is developing the projects with the support of a grant from the National Science Foundation, and a first draft of the projects should be available in the summer of 2000 at the author's Web site, <http://www.cs.csustan.edu/~rsc>.

The process the projects illustrate is straightforward and is described in Figure 4. In the student's area of science, we choose a problem and describe to the student how geometric information may be developed from that problem. This geometric information may be data vectors, may be an abstraction of reality, or may be an expression of a piece of theoretical work, for example. The project then asks the student to use the graphics system to create an image based on that geometry, and finally the student is invited to draw a conclusion about the science (or about the problem in the science) from the image. This puts the students' work in computer graphics in the context of their understanding of science, giving an added value to their programming work.

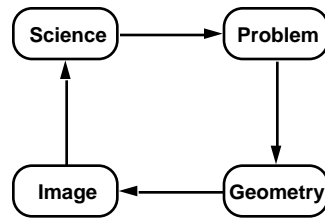


Figure 4: the process described by student projects

The language and systems used for projects is not an issue, though it happens that we use C or C++ on Macintosh and NT systems. In the future we expect to consider offering the course with APIs such as Java3D or perhaps Fahrenheit, depending on the availability of the systems and the amount of programming background needed to use them. Being independent of the language and systems is very important in making the course work for non-computer science students and cannot be overemphasized in permitting students to create interesting and useful images without using sophisticated programming techniques.

### The Sequel to this Course

This approach to the course is very promising. At the end of the introductory course, students have learned to think geometrically, to create programs that express their geometric thinking, and to write programs that implement modest amounts of animation and interaction. They have seen an overview of the graphics pipeline, have seen the effect and behavior of a number of graphics processes, and have had a modest description of some graphics algorithms, but have done little direct work with those algorithms. This gives my computer science students an excellent background for tackling a second graphics course containing the traditional fundamental algorithms content.

As I teach it, the second course focuses on familiar fundamental techniques. This may be done in many ways, depending on the instructor's experience and the goals of the students and the curriculum. These can include interpolation material such as scan-converting polygons, shading models, Z-buffering, texturing, and bump and environment mapping, as well as techniques such as antialiasing and ray tracing. With the experience and the basic use of key concepts my students got from the first course, they were able to develop a good understanding of more advanced graphics techniques and were able to move quite quickly through material that had previously been fairly difficult.

The value of the traditional computer graphics principles course does not seem to be reduced by beginning computer graphics studies with this new kind of introductory course. On the contrary, the combination of an introductory course based on graphics API programming and a subsequent fundamental principles course seems to be an excellent introduction to computer graphics content for computer science students. Additional advanced courses can be developed as needed when additional topics need to be covered.

### Summary

One of the challenges of computer science is to find ways to serve our universities and still maintain the quality of our courses and programs. Because computer graphics is so useful to so many different fields, it is a natural course to take on a service role, but because the course has traditionally made significant mathematical and programming demands on its students, it has rarely been used in this way.

The approach outlined in this note offers us a way to provide a course that benefits both the computer science student and the student from mathematics, science, or

engineering. It is accessible to faculty, is useful to computer science students, opens the door to further studies of more advanced computer graphics topics, and is a valuable service to other students whose professional lives will require continual use of computer graphics. And let's remember that when students have the opportunity to do work that they enjoy and that means something to them, the course is a lot of fun to teach and students really enjoy the high-quality work they produce.

## References

- [1] Brown, Judith R., Steve Cunningham, and Mike McGrath, "Visualization in Science and Engineering Education," in *IEEE Tutorial: Scientific Visualization*, Nielson, Gregory M. and Bruce Shriver, eds., IEEE Computer Society, 1990
- [2] Cunningham, Steve, "Outside the Box — The Changing Shape of the Computing World," invited editorial, *SIGCSE Bulletin* 30(4), December 1998, 4a-7a
- [3] Cunningham, Steve, "Powers of 10: The Case for Changing the First Course in Computer Graphics," submitted
- [4] Reports of the Graphics and Visualization Education 99 workshop are published online at [www.eg.org/WorkingGroups/GVE/GVE99](http://www.eg.org/WorkingGroups/GVE/GVE99) and [www.education.siggraph.org/conferences/GVE99/](http://www.education.siggraph.org/conferences/GVE99/) and have been submitted to *Computer Graphics* and *Computer Graphics Forum*.
- [5] Tucker, Allen B. and Bruce H. Barnes, eds., *Computing Curricula 1991: Report of the ACM/IEEE/CS Curriculum Task Force*, ACM Press/IEEE Computer Society Press, 1991

This work is partially supported by National Science Foundation grant DUE-9950121. All opinions, findings, conclusions, and recommendations in this work are those of the author and do not necessarily reflect the views of the National Science Foundation.

- \* For 1999-2000, the author's address is  
San Diego Supercomputer Center  
P.O. Box 85608  
San Diego, CA 92186  
rsc@sdsc.edu