The Beginning Computer Graphics Course in Computer Science

A report from the working group on computer graphics in computer science 2004 SIGGRAPH/Eurographics Computer Graphics Education workshop Zhejiang University, Hangzhou, China, June 2-6, 2004

Steve Cunningham, U.S. National Science Foundation, Arlington, Virginia, USA
Werner Hansmann, Universität Hamburg, Hamburg, Germany
Cary Laxer, Rose-Hulman Institute of Technology, Terre Haute, Indiana, USA
Jiaoying Shi, Zhejiang University, Hangzhou, China

The CGE04 workshop [04] was the third in a series of computer graphics edutaion workshops sponsored by ACM SIGGRAPH and Eurographics. These workshops focus on considering problems in computer graphics education and making recommendations based on those considerations. This report covers the discussion of teaching computer graphics in computer science and focuses almost entirely on teaching the beginning course in the subject.

Computer graphics courses grew up following a very few key textbooks that were based on the technology of the 1970s and 1980s, when graphics standards were 2D and clumsy, graphics was an expensive and nonstandard capability, and there were no standard graphics tools. Faculty who taught graphics grew up in that environment and it was difficult to get people to think about the computer graphics course in a new and more general setting. Computer graphics had the reputation of being very difficult and very mathematical, as it originally was.

In 1999 we agreed that the "graphics pipeline"—the computation process that takes 3D geometry in Euclidean modeling space and converts it to pixels on a screen—was key to the beginning graphics course [99]. This implied that the fundamental modeling paradigm for the beginning course would be polygons, or what we identified at the workshop as "triangles with vertex attributes." We also noted the importance of including interaction techniques in that course and on recognizing that graphics students need to know the visual effect of some choices in the pipeline processes.

In 2002 we described particular aspects of the subject that a student should know on completing a beginning graphics course [02]. Many of these expanded steps in the pipeline and so reinforced the 1999 statement; some involved newer capabilities in more modern graphics systems and represent newer capabilities. These topics are

* Transformations
* Modeling: primitives, surfaces, and scene graphs
* Viewing and projection
* Perception and colour models
* Lighting and shading
* Interaction, both event-driven and using selection
* Animation and time-dependent behaviour
* Texture mapping

This is a much more specific statement about course topics than the 1999 workshop produced. While it continues to look at "triangles with vertex attributes," it is clear that the attributes being considered are colors, normals, or texture coordinates. In the future we expect the list of vertex attributes to expand to include such concepts as vertex shaders. We noted that computer graphics courses play very different roles in different universities, so while we present these topics as a general goal, we expect that when a program clearly has a different need, it could be expected to take a different tack for the beginning course.

In 2004, we started by asking participants think of questions about teaching computer graphics. This was very free-form and nearly two dozen questions found their way to the board. From this list, we started grouping topics and focusing on what was most important to the group, and we found ourselves looking again at the beginning graphics course. We looked particularly at the concept of whether computer science students should have access to learning about computer graphics — everybody's computer has this capability and it seemed wasteful to teach computer science without giving students a chance to learn about it. Our first statement, then, was this:

> **Every undergraduate computer science student should have the opportunity to have at least a meaningful introduction to computer graphics.**

We were explicit about the word "every" and intended this to mean all postsecondary institutions: two-year colleges, polytechnics, liberal-arts schools, regional comprehensive universities, and major research universities. This is at variance with the ACM/IEEE Curriculum 2001 recommendations that identify only a very small amount of graphics as important in computer science. This has enormous implications because most postsecondary schools do not currently teach computer graphics, either through a graphics course or as part of another course, and most computer science faculty do not have a background in the subject. We realize that this statement means that we must define a course that can be taught without lot of difficult faculty development. Fortunately, the graphics capabilities of modern computers and the availability of sound graphics APIs give instructors enough tools that this is now possible.

For the course itself, we identified the course goals using an outcomes-based definition of ***learning*** that has three components: what the student knows, what the student can do, and what attitudes and approaches the student has developed. These seemed to map onto some of the questions so we looked at the beginning course in these terms. The 2002 workshop had focused on what the student should *know*, and we saw little need to look further at this set of basic concepts. The other parts of this definition of learning opened new directions for us. The question of what a student could *do* led to a realization that many projects in graphics courses look only at making images without any thought of what the images might mean, and we decided that a course should help a student realize that the images he or she makes can communicate useful information. The question of what kinds of *approaches* to problems a student should learn led us to a consideration of how graphics is part of problem-solving and we were able to use a "visual mantra" from Mike Bailey (see figure) to illustrate this approach. These were important insights for our discussions.

In the end we used these three components to describe the nature of the beginning computer graphics course as we see it at the start of the 21st century. We believe that these will help faculty develop a course that gives the student important tools as well as a sound background for more graphics work.

1. What the student should know:

The student should understand the processing implied by the graphics pipeline and polygon-based modeling with vertex attributes, including the following components:

* Transformations
* Modeling: primitives, surfaces, and scene graphs
* Viewing and projection
* Perception and colour models
* Lighting and shading
* Interaction, both event-driven and using selection
* Animation and time-dependent behaviour
* Texture mapping

These knowledge topics are intended to be independent of the graphics API and hardware used in the course, and should be largely independent of such problem-structuring tools as the scene graph. In general we believe that this knowledge should be conceptual and should not require that the student implement the various algorithms and processes that go into these topics. This can be supplemented by student implementations of some of these topics as time permits, but such implementations should not supplant the visual communication and problem-solving discussed below.

2. What the student should be able to do:

The student should be able to use a modern graphics API to create a graphics application that can be integrated with other computer applications.

This learning is primarily focused in the projects that accompany a graphics course, and we suggest that these projects should not be artificial exercises that use graphics without reference to application areas but should integrate graphics with areas where the graphics is a key component. We further suggest that the projects should emphasize the quality of the communication or presentation that the students' images will create [VL].

3. What approaches the student should bring to a problem:

In the traditional problem solving process we see several steps: recognizing a problem, building a model of the problem from whatever knowledge is available, developing a tentative solution to the problem based on that model, and testing the tentative solution against the problem. Any shortcomings of the tentative solution are identified, and the model and solution are then revised to address these shortcomings. This cycle continues until a satisfactory model and solution have been found.

A student who has studied computer graphics should have experience with thinking visually about problems, and so the student should be able to take the model and create a visual representation of the problem or model that can help develop a tentative solution. In a graphics course, this corresponds to the part of the figure above labeled "model —> image". Thus the student with a computer graphics background should naturally think of creating a visualization for a problem as a way to think about it or communicate it to others, and this as a way to help create a tentative solution.

We recognize that there are circumstances when a beginning graphics course can have different kinds of learning goals that those above. An example might be a first course in a graduate sequence that leads to advanced degrees in computer science with a computer graphics specialization. But with these goals serving to define a standard beginning course, a program having a different kind of course should be able to give its own set of learning goals and use them to identify how their course differs from the standard course and why this difference is important in their program.

References: the Web sites below give reports of previous workshops and the Call for Participation for the 2004 workshop.

[99] http://www.siggraph.org/education/conferences/GVE99/index.html
[02] http://www.siggraph.org/education/bristol/bristol.htm
[VL] http://www.siggraph.org/education/vl/vl.htm
[04] http://www.cad.zju.edu.cn/cge04/

Workshop participants:

Qin Aihong, Shanxi University
Shanshan Chen, Fudan University
Xie Cui, Dalian Maritime University
Steve Cunningham, CSU Stanislaus, National Science Foundation
Peter Giles, Australian Film & Radio School
Werner Hansmann, University of Hamburg
Cary Laxer, Rose-Hulman Institute of Technology
Berndt Lutz, IGD, Fraunhofer Institute for Computer Graphics
Eric Paquette, ETS Montreal
Russell Pensyl, Beijing School of Software
I-fen Shen, Fudan University
Jiaoying Shi, Zhejiang University
Kelvin Sung, University of Washington Bothell
Elizabeth Sweedyk, Harvey Mudd College