Teaching Computer Graphics without Raster-Level Algorithms

Panelists: Edward Angel, University of New Mexico
Steve Cunningham, Grinnell College
Peter Shirley, University of Utah
Kelvin Sung, University of Washington, Bothell (Moderator)

SUMMARY

Classical computer graphics textbooks (e.g. [1-4]) introduce the field by covering the details of raster-level algorithms. Many computer graphics educators have long recognized that depending on students' backgrounds and needs (e.g. major vs non-major) [5,6] alternate approaches may be more appropriate [7-8]. With the recent advances in the field, advent of powerful hardware, and sophisticated APIs, the field of computer graphics has become much larger and richer. As educators, we must make tough decisions about what to include in a one semester/quarter class. This panel presents three distinct approaches to teaching introductory Computer Graphics without covering raster-level algorithms. These approaches are suitable for a wide-range of students with different backgrounds and needs. To ensure neutrality and balanced of viewpoints, the panel also discusses the merits of teaching raster-level algorithms and situations where the classical approach better aligns with and serves students' needs.

Categories and Subject Descriptors

I.3.0 [**Computer Graphics**]: General.

General Terms: Design, Experimentation.

Keywords: Curriculum Issues, Courseware, Graphics/Visualization.

Edward Angel: Teaching computer graphics concepts with a high-level API. I've been teaching computer graphics for computer science and engineering seniors and graduate students for over 20 years. About 10 years ago, I switched to the top-down approach that is the basis of my textbook [9]. My philosophy is best described in the preface using an automobile analogy due to John Kemeny, a pioneer in computer education: "You don't have to know what's under the hood to be literate, but unless you know how to program, you'll be sitting in the back seat instead of driving. That same analogy applies to the way we teach computer graphics. One approach, the algorithmic approach, is to teach everything about what makes a car function: the engine, the transmission, the combustion process. A second approach, the survey approach, is to hire a chauffeur, to sit back, and to see the world as a spectator. The third approach, the programming approach that I have adopted here, is to teach you how to drive and how to take yourself wherever you want to go."

Although there is no one "right" approach, the top-down programming-oriented approach appears to be the most popular among CS departments. One strength is that it is a fun class for the students while serving as a capstone course that allows them to apply their knowledge from classes in programming, data structures, algorithms, and software engineering to an area that truly excites them. My class has evolved to one in which the students start with 2-3 beginning projects over the first half of the semester and then each student spends the rest of the semester on a self-selected term project. There are no written examinations. One criticism of this approach is that students tend to be weaker on graphics algorithms than with the bottom-up approach. Nevertheless, I prefer this approach in an environment in which most students will not go on to graduate study in the graphics field (although many have done so successfully). As I look to the future, I believe that recent advances in computer graphics, such as the advent of programmable pipelines, should reinforce the importance of top-down approach.

Steve Cunningham: Teaching computer graphics by focusing student work on an application area. The beginning course in computer graphics is changing, driven partly by the availability of high-performance graphics systems in PCs and partly by the interest of students in games and animation. The change is reflected in the fact that this panel is focused on "top-down" graphics, or graphics courses where the development of algorithms, projections, and scan conversion is no longer a major part. In practice, top-down probably means "graphics-with-an-API" where the API is OpenGL, DirectX, or something else.

I believe that computer graphics should be a general skill and capability that our students will use throughout their careers [10], but by itself a "top-down" approach does not focus on what graduates can do with their graphics. If our graphics courses are self-referential, and focus only on graphics without a significant contact with the ways graphics is used, then we have simply built another silo in the academic barnyard and our students will not understand what a wonderful tool graphics is.

There are certainly graphics courses that focus on using the subject. There are books on games graphics and class projects with a games focus, so games may be a way to organize a beginning course. Personally, I have chosen to organize a course around graphical communication, with an emphasis on communication in the sciences. In effect, this becomes a scientific visualization course without the graphics tool emphasis often found in such courses. What can we say about science in only a beginning graphics course? Quite a lot. We build models of scientific processes, and modeling becomes a natural way to describe the geometry that underlies the science. We consider time-varying behavior, and we naturally have animations. We want a user to interact with a model to understand the science better, and we get event-driven

SIGCSE'06, March 1-5, 2006, Houston, Texas, USA.
ACM 1-59593-259-3/06/0003.

interactive graphics. We want more realistic environments to present scientific activities, and we get texture mapping. In short, we can cover everything you would find in any first course, and presenting science lets us motivate all of them. And we have given the student a good background that he or she can use to go on to more advanced studies.

Kelvin Sung: Integrating computer graphics concepts in moderately complex applications. Many Computer Science students are looking for an education with a direct applicability. This is especially true in a high-profile field like Computer Graphics, where students are familiar with a lot of the popular applications (e.g. graphical editors, games, special effects, etc.). Many students are motivated and enthusiastic about the computer graphics field because they want to understand the theories behind and how to build these fascinating applications. As educators, we would like to concentrate on fundamental principles and competencies where there is potential applicability throughout students' life and in their careers. Many educators align these needs by introducing case studies to relate fundamental principles to real-world applications. Modern graphics applications require the collaboration of many concepts. For example, the functionality of the grouping and ungrouping operations are based on multiple graphics concepts (coordinate transformation, hierarchical modeling) where the behaviors of these operations are integrated with user interaction issues (selection, direct manipulation).

I teach introductory computer graphics based on: (1) identifying and decomposing a moderately complex interactive graphics application into essential concepts; (2) designing and implementing case studies to teach/demonstrate these concepts; (3) requiring students integrate these concepts into their own moderately complex applications [11]. In this way, it would be clear to students how concepts learned in classes relate to the functionality and behaviors of familiar applications. With this approach it is important to identify a target application that demands a sufficiently large set of essential concepts common to many graphics software systems. In addition, one must remember that syllabus that is not based on "first principal" runs the risks of teaching students to be users rather than practitioners. As an example, in our approach, we would identify major functional modules in applications like Maya (e.g. user interaction module, or hierarchical modeling module) and examine how to design and implement these modules based on functionality supported by OpenGL or DirectX. This is different from learning how to use Maya. As newer versions of the software and/or APIs are released, the knowledge students gained must continued to be valid and applicable.

Peter Shirley: The case for traditional scan conversion approach to teaching computer graphics. I have taught five introductory computer graphics classes at Indiana University and the University of Utah. I was the TA for two introductory computer graphics at the University of Illinois. All of these classes but one would be considered "bottom-up" with students writing all code from scratch, down to line and triangle rasterization. I took two introductory graphics classes as a student. The first from William Kubitz was a hybrid course using GKS where there was a high-level 2D API. The second was from George Francis and used raw GL using mathematical visualization as the driving problem. These experiences as a whole have led me to believe that there is no "right way" to teach graphics, but the type of students in the classroom as well as whether the introductory course is part of a sequence influence the trade-offs in deciding between high and low-level approaches.

Kelvin Sung and I have argued that for mature programmers taking a single graphics course [11], the advantages of a high-level approach are amplified. However, my empirical experience is that the fundamentals are more effectively absorbed by students when they have to implement rasterization, viewing, and hidden-surface removal. It is impossible for students to implement such a pipeline without fully understanding barycentric coordinate, matrix transformations and homogeneous coordinates, and the raw simplicity of the z-buffer algorithm. Such students will ultimately be more effective users of 3D APIs than students that only partially understand those fundamentals. Unfortunately, that low-level material takes most of a full-semester course, leaving no time for interactive programming. The advantages of this low-level approach are most obvious in a multi-course sequence that moves from low-level to high-level programming. For this reason I think the details of student population and the number of graphics courses in a curriculum are crucial issues to consider before choosing an approach for a particular institution.

REFERENCES

[1] Edward Angel, "Computer Graphics," Addison Wesley, 1990.

[2] James Foley, Andries van Dam, Steven Feiner, John Hughes, and Richard Phillips, "Introduction to Computer Graphics," Addison Wesley, 1994.

[3] Donald Hearn and Pauline Baker, "Computer Graphics – C Version, ", second edition, Prentice Hall, 1997.

[4] Peter Shirley, "Fundamentals of Computer Graphics," A. K. Peters, 2002.

[5] Judith Brown, Robert Burton, Steve Cunningham, and Mark Ohlson, "Varieties of Computer Graphics Courses in Computer Science," 19th ACM SIGCSE Technical Symposium on Computer Science Education, pp. 313-313, February 1988.

[6] Rosalee Wolfe, "Bringing the Introductory Computer Graphics Course into the 21st Century, " Proceedings of the Graphics and Visualization Education (GVE'99) Workshop. pp. 3-6, 1999.

[7] Scott Grissom, Jack Bresenham, Bill Kubitz, and Scott Owen, "Approaches to Teaching Computer Graphics, " 26th ACM SIGCSE Technical Symposium on Computer Science Education, pp. 382-383, March 1995..

[8] Lew Hitchner, Steve Cummingham, Scott Grissom, and Rosalee Wolfe, "Computer Graphics: The Introductory Grows Up, " 30th ACM SIGCSE Technical Symposium on Computer Science Education, pp. 341-342, March 1999.

[9] Edward Angel, "Interactive Computer Graphics: A Top-down Approach using OpenGL," (Fourth Edition), Addison-Wesley, 2006.

[10] Steve Cunningham, "Powers of 10: The Case for Changing the First Course in Computer Graphics," Proceedings of the SIGCSE 2000 Technical Symposium on Computer Science Education, pp. 293-296, March 2000.

[11] K. Sung, and P. Shirley, "A Top-Down Approach to Teaching Introductory Computer Graphics," Computer & Graphics, Vol. 28, Issue 3, PP. 383-391, June 20042004 (full-length paper based on ACM SIGGRAPH 2003 Educator's Program Conference Paper).