From Pixels to Scene Graphs: The Evolution of Computer Graphics Courses

Contact
Dennis J Bouvier
University of Houston-Clear Lake
2700 Bay Area Boulevard
Houston, Texas 77058 USA
djb@acm.org

The field of computer graphics is expanding and evolving rapidly. Past computer graphics courses addressed low-level topics such as line-drawing algorithms[13] while the graphics industry has moved toward scene graph representations of three-dimensional scenes. This discrepancy has moved some educators to suggest that graphics courses exclude 2D topics for concentration on 3D. This paper suggests that such a switch is unnecessary and provides a course outline blending old and new topics to provide breadth and relevancy without abandoning the basics.

Since the time of Ivan Sutherland's doctoral work[11] much has changed in computer graphics, and much remains the same. One major change is the pervasiveness of graphics. Computer graphics technology was once reserved for a small number of researchers. Now, with significant computing power available on so many desktops, graphics has become commonplace. So ubiquitous is graphics that we tend to forget that WYSIWYG GUIs are composed of appropriately clipped lines, bit-mapped images, and scalable text. All are produced by some graphic algorithm. These WYSIWYG GUI features are commonly available as part of user interface programming libraries that are not even considered graphics packages.

This ubiquity serves as the impetus for training, or at least introducing, a larger population of students to computer graphics technology. The changing technological landscape provides the motivation to change the content of computer graphics courses. So, once again, the question is "What should be included in the introductory computer graphics course?"

Selecting computing course content presents a design tradeoff between providing technical relevancy and proper intellectual training. Many courses in computer science programs present similar course content design decisions. Often a wide variety of reasonable choices leaves room for a variety of opinions. For example, the selection of the programming language in introductory programming courses is a long-running debate destined not to end. The computer graphics course has a similar choice among graphics APIs.

A more important decision is the relative amount of time devoted to 2D versus 3D topics. This course content issue has been politely debated in a number of publications by a relatively small number of authors[3, 4, 5, 6, 10, 13]. This paper adds one more voice to the debate and a new point of view.

From pixels to scene graphs

Recent developments in computer graphics technology have changed the way in which graphics are used and perceived.

Comparing the past and the present situation forms a basis for reasonable discussion for the future. As this paper is focused on the future of teaching computer graphics, the historical look is focused on educational issues more than technology. Each of the recent papers that discusses the content of a computer graphics course provides a summary of technological history[3, 4, 5, 6, 10, 13]. The reader interested in this history is referred to these sources.

Computer graphics past

Historically, an introduction to computer graphics course usually started with the definition of a pixel. From the most basic of primitives, the course moved to 2D topics such as line and circle-drawing algorithms[13]. In courses at many institutions, students worked on graphics programming projects using non-standard graphics libraries, or with no library at all. These courses commonly included 3D topics only at the end of the course.

In the computer graphics courses of the 90s, OpenGL was commonly used for programming projects. Scene graph technology is only now becoming common in undergraduate and graduate level courses.

Computer graphics present

As SIGGRAPH attendees know, computer graphics technology is advancing at a pace that defies description in a few paragraphs. However, it is often easy to experience the results of these changes. Examples of graphics technology are often displayed on movie and television screens, in feature films and commercials. The technological advances are no longer only available to the select few; powerful 3D graphics hardware is available to the consumer. The hardware coupled with graphics application software has turned some average computer users into illustrators and animators. In addition, many computationally expensive graphics algorithms have been moved to the hardware. From MMX extensions in Pentium processors, to the ray tracing hardware (RenderDrive), and consumer 3D graphics rendering cards and chips (GeForce).

What is relevant to this discussion is not so much exactly what the advances are, but what impact these advances should have on computer graphics education.

With changing technology and its availability, an educator must consider the need for updating course content. A number of educators have offered their suggestions for changes. Brief summaries of two published papers appear in the next section.

Two previous course proposals

Using high-level graphics APIs in programming renders unnecessary detailed knowledge of low-level issues, such as line-drawing algorithms. For this reason, several educators now advocate elimination of the historically basic topics from the "introduction to computer graphics course."

All referenced papers argue that changing technology provides the rationale for a changing graphics course. However, Cunningham takes this a step further, observing that with the growth in graphics availability, through the growing number and variety of applications, there is a large population of computer graphics users to be served in education[5,6]. In addressing this larger audience, Cunningham proposes that "all course work is done entirely in 3D, with no reference at all to 2D graphics except perhaps in a few specific examples"[6].

Hitchner and Sowizral propose a complete change in course structure and move directly to scene graph based systems[10]. One argument presented for this change is that state-of-the-art implementations of graphics technology do not make use of traditional algorithms, "Scan Line polygon fill is not used in a custom parallel architecture built for rendering speed".[10]

Proposed Course Content

While the stated positions are valid, other course outlines also address advances in graphics technology without excluding traditional 2D topics. In this proposal, 2D topics are seen as more than just foundation for 3D topics, but relevant and meaningful topics on their own.

Noting technological progress, market changes, and the positions of other educators, the following course content outline is proposed:

Course Design Considerations

The following course outline is founded upon these tenants:

- Knowledge of lower-level layers provides an understanding of the basic problems, which provides the foundation for reasoning about what is possible and what is not, what is expensive and how to make it less so.
- Although much of the course is focused on 3D graphics, virtually all computer graphics systems use some form of 2D graphics display technology. For this reason, it is appropriate to include 2D topics in an introductory computer graphics course.
- A university course should not train students in a specific software product, but give students the opportunities to learn and explore the possibilities. Specifically, the student is expected to learn the details of commercial software products and/or APIs in programming projects, not lectures.
- The very dynamics of the market are a good reason to avoid radical changes in course content. The technological half-life of the current hot topics is probably shorter than the technology it replaced. Further, some technology becomes attractive after losing popularity. For example, vector graphics has become attractive once again as a way to reproduce graphic art over a network connection. Some Web products (for example, Flash) utilize vector representations to speed graphic descriptions to remote viewers.

Proposed course outline

The following list of topic areas shows the basic outline of the proposed course.

As the outline shows, the students study graphics topics from pixels to scene graphs.

Topic 1: Pixels, images, file formats

Virtually all computer graphics work is presented in 2D, and much is stored in files. These topics not only provide foundation for 3D topics to come, but are relevant on their own.

To begin the course, a foundation is laid with 2D fundamentals, including the definition of pixels and image plane. This material includes a few example image-file formats including at least one lossy and one loss-less format. Also discussed are vector vs. raster issues and compression techniques.

Topic 2: Line drawing, circle drawing

The basic line- and circle-drawing algorithms demonstrate creative ways to improve speed as well as the first occasion to talk about aliasing and anti-aliasing. The line drawing algorithm is used in polygon filling discussions. Line clipping in 2D can be included in this material as well.

Later in the course, it is noted that 3D graphics are projected to a 2D image plane before rendering; therefore, 2D line drawing and polygon filling is used in virtually all 3D graphics systems.

Topic 3: Color, lighting, shading

Prior to this point, only grayscale images have been discussed. Positive and negative color systems are introduced as well as a variety of color representations such as RGB and CMY. The discussion of color lays a foundation for shading.

Topic 4: 2D transformations
Two-dimensional translation, rotation, and scaling transforms, their matrix representations, and the use of homogeneous coordinates are presented.

Two-dimensional transformations are used in "paint" applications and in user interfaces, and are worthy of inclusion in the course. However, they are highly valuable in introducing 3D transformations. Two-dimensional images are easily drawn and 2D transformations are easily demonstrated in the class.

Topic 5: 3D transformations
Three-dimensional transforms and homogenous coordinates are easily introduced as natural extensions to the corresponding 2D topics. In both 2D and 3D transformations, make the distinction between point and coordinate transformations. At this point it is possible to introduce the OpenGL API.

Topic 6: Projection
Having homogenous coordinates as a tool, parallel and perspective projects are presented as matrix multiplication. It is important to explain the view frustum, 3D clipping, and perspective division.

Topic 7: Introduction to the graphics pipeline
The flow of data originating from the object vertex through 3D transformations and projection resulting in image plane points brings the course back to 2D line drawing and 2D polygon filling.

Topic 8: Hidden line and surface removal
A variety of algorithms is presented for solving occlusion problems. Typically, Painter's Algorithm, Binary Space Partition, Area Subdivision, and Z-Buffer are included in the discussions. Ray tracing is also included as an alternative. The discussion of ray tracing also includes shadows.

Topic 9: Animation
Basic animation issues and techniques are discussed. Time constraints usually allow very little detail on the use of key framing and motion capture. The focus is on animation using procedural techniques such as particle systems, motion capture, and inverse kinematics.

Topic 10: Scene graphs
Basic concepts of the representation of 3D scenes in a graph-data structure are presented. Programming examples are useful in illustrating the utility of this approach.

Topic 11: Texture mapping, bump mapping
Texture mapping and bump mapping are presented as alternatives to detailed 3D geometry.

Other Topics
The field of computer graphics is wide, and this course doesn't cover all possible topics. It would be easy to include discussions on transparency and interactivity. Image processing and text rendering are other possible topic areas. Typically, one or more of these topics is introduced at the end of the course, as time permits.

General comments
This course is intended for a 15-week semester. Most of the identified topic areas can be covered in about one week. The topic areas that included projection take more time than other topic areas. The timing of each topic area can be adjusted by including more or less detail on specific topics. Usually the discussions of clipping and arbitrary projections are short.

Some reordering of course topics can be done. In particular, the introduction of scene graphs (topic 10) can be moved to topic eight. However, the suggested order of topics allows sufficient time for programming projects as presented below.

Programming projects
The course as taught by the author requires a number of student programming projects. The projects reinforce the class material and expose the students to modern APIs such as OpenGL and Java 3D. The APIs are introduced in the lecture, but are learned in the programming projects.

Students are usually excited by the results of their work and quite often will go well beyond the requirements of the programming projects.

Project 1: Line drawing and file formats
The first project is completed without the use of any graphics API. The students are given an example program demonstrating how an array is used as an image plane and is written to a simple file format. The students are also provided with a utility for converting the file to other formats.

Students implement a line-drawing algorithm and use it to create a perspective line drawing of a 3D object without knowledge of projection techniques. In this project, students learn the details of line drawing and the difficulty of simulating 3D "by hand."

In addition, the students save their images to a file and convert it to a variety of file formats. The students then prepare a report of image file size and note degradation. This experience makes concrete the discussion of aliasing as well as the implications of using lossy file formats.

Project 2: Simple OpenGL
Using OpenGL, the students are required to produce a perspective projection of a static 3D scene. Students are given a simple example program that is used as an example and a skeleton.

Project 3: Simple Java 3D
Similar to Project 2, but using the Java 3D API to construct the 3D scene.

Other programming projects
The three programming projects outlined above also naturally lead to more programming projects. Quite often, the author requires three projects in addition to those described. A fourth is a color version of Project 1. The fifth and sixth are interactive versions of Projects 2 and 3.

Texts and support material
While a number of books may be used as the textbook for the course, all have some disadvantages. That is, no existing text covers all of the course topics for the proposed course outline. One possible text is Foley et al[7], or the more compact version[8]. This text requires the instructor to provide additional material on any software used in the course. Alternatives include recently published computer graphics texts that include the OpenGL API such as Angel[1] or Hill[9]. Another possibility is Watt[12] which includes no API and no 2D line drawing.

Missing from all of the suggested texts is information on scene graph software. Probably fewer than 10 books may be easily found that cover any scene graph API. Fortunately there is a free and readily available tutorial on Java 3D[2]. With this tutorial, it is easier for students to use the Java 3D API than Open Inventor for scene graph programming projects.

Also, few texts mention the issues surrounding image file formats. This material is provided to the students by this author.

Summary

A variety of factors motivate educators to make adjustments in course content. The growth in the number of graphics applications and advances in graphics technology has prompted a variety of educators to publish thoughts on the contents of computer graphics course content. While the views of these authors are valid, this paper provides another perspective.

The proposed course provides the student with sufficient background to read research papers, to enroll in upper level courses such as visualization, and to begin undertaking more complex undertakings in the field.

The proposed outline has been used as the syllabus for a senior-level computer graphics course taught by the author. The course remains popular with students and receives above-average reviews. The students express their interest and satisfaction with the course in a variety of ways, including a low drop rate.

References

1. Angel, E. (1997). *Interactive computer graphics, a top-down Approach with OpenGL*, Addison-Wesley.
2. Bouvier, D. *Getting Started with the Java 3D API*, URL: java.sun.com/products/java-media/3d/collateral.
3. Breshingham, J. (1999). Teaching the graphics processing pipeline: Cosmetic and geometric attribute implications. In *Proc. Graphics and Visualization Education* (99).
4. Cunningham, S. (1998). Outside the Box - The changing shape of the computing world, invited editorial. *SIGCSE Bulletin* 30 (4), 4a-7a.
5. Cunningham, S. (2000). Powers of 10: The case for changing the first course in computer graphics. In *Proc. SIGCSE Technical Symposium on Computer Science Education*, 46-49.
6. Cunningham, S. (1999). Re-inventing the introductory computer graphics course: Providing gools for a wider audience. In *Proc. Graphics and Visualization Education* (99).
7. Foley, J. et al. *Computer graphics*. Addison Wesley.
8. Foley, J. et al. (1993). *Introduction to computer graphics*. Addison Wesley.
9. Hill, (2000). *Computer Graphics Using OpenGL*. Prentice Hall.
10. Hitchner, L. & Sowizral, H. (1999). Adapting computer graphics curricula to changes in graphics. *Proc. Graphics and Visualization Education* (99).
11. Sutherland, I.E. (1963). Sketchpad: A man-machine graphical communication System. In SJCC, Spartan Books.
12. Watt, A. (2000). *3D computer graphics*. Addison Wesley.
13. Wolfe, R. (1999). Bringing the introductory computer graphics course into the 21st century. In *Proc. Graphics and Visualization Education* (99).