Bringing the Introductory Computer Graphics Course into the 21st Century

Rosalee Wolfe
DePaul University
Chicago, Illinois
wolfe@cs.depaul.edu

Abstract

The field of computer graphics has matured greatly since the formal statement of the introductory undergraduate course for computer science majors was created for ACM/IEEE Curriculum 91, and introductory courses need to reflect the substantive changes in the discipline. Recent discussions with graphics educators and a syllabus survey have found six trends in recent course offerings. Perhaps these findings will evolve into a basis upon which people can develop courses that fit their local needs as well as reflecting the changing field.
Keywords: *introductory graphics course, computer graphics curriculum.*

1. Introduction

Although curriculum is a recurring theme of conversation in the hallways at conferences involving graphics and education, *Curriculum 91*[1] marks the most recent formal discussion in the United States of the topics in an introductory computer graphics course. It was published in February 1991, nearly nine years ago. Nine years represents a significant percentage of the discipline's life span. Although the document was published in 1991, it reflects accepted practice from the late 1980s. So in fact it has been ten years or even longer since substantive discussions on this topic have taken place. This paper reviews past practice, surveys current practice and explores future possibilities for introductory graphics courses.

1.1. Graphics in the late 80s

The computing environment of ten years ago presented several substantial technology challenges to a graphics instructor. Graphics hardware was a heavy drain on university resources [2] [3] and graphics equipment was not as ubiquitous as it is today. *Curriculum 91* mentions "a high quality color display" as a special laboratory item. Intel-based PCs were expensive and many schools relied on graphics terminals connected to a mainframe via serial lines.

Another special need that *Curriculum 91* mentions is "a suite of graphics software tools." Graphics software was scarce, expensive and difficult to access. When trying to find appropriate software an instructor faced three difficult alternatives. The first was to find the considerable monetary resources necessary to purchase a graphics library from a vendor. At the time, these libraries ran into the tens of thousands of dollars, and many schools simply could not afford the purchase.

The second option was to use free software, typically developed at another university[4]. This was at a time before Internet use was as widespread as it is today, and net searching was not possible. In many cases an instructor learned about such software by reading a published article or by word of mouth. Often, just to obtain the software would require sending a magnetic tape through the mail and waiting weeks for it to return. While the price was right, most freeware demanded large amounts of time to install and even then the instructor often had to spend additional time developing custom software in order to accommodate the peculiarities of the school's hardware.

As a result many instructors chose a third approach to software tools. On their local system, they wrote just enough software to supply their students with the bare necessities for graphics work [5]. This included the functions GetPixel(), PutPixel() and maybe DrawLine() in addition to routines that began and ended a graphics session. This slim set of tools was all that students had for software development.

In this setting, students in an introductory graphics course would write an entire graphics package from the ground up. Class lectures would parallel the software development and cover such topics as those listed in Table 1.

The pace of coverage was somewhat dictated by the software implementation progress, and often the hardware added complexity to an already challenging software problem. Compensating for the low number of colors available on a graphics display required quantization and dithering. Merely viewing an image on a graphics terminal could be a time consuming

- basic concepts and hardware
- line and curve drawing
- window-to-viewport transformation
- clipping
- modelling transformations
- polygon rasterisation
- viewing transformations
- surface algorithms
- simple illumination
- shading
- interactive techniques

Table 1: Graphics courses in the late 1980s typically devoted 50% of their time to two-dimensional topics.

process, as it took over 20 minutes to display an image with a resolution of 800x480.

All of these factors influenced course content. Typically fifty percent of a late 1980s introductory graphics course covered two-dimensional issues[6, 7].

1.2. Graphics in the late 90s

In contrast, the computer hardware of the late 1990s has become so cheap that graphics has come into the mainstream of computing. Video cards with 24-bit per pixel capability are common on computers that retail for $1,000 or less. High school students with little or no programming background take pictures with digital cameras, download the images onto disk, use PhotoShop for retouching, and post the results on the Web. Students arrive at introductory computer graphics courses with a far greater acquaintance with rudimentary basics than they did ten years ago.

Equally as important, software for three-dimensional graphics is cheap and easily available via the Internet. Today many 3D modelling and rendering packages are available for free or for a modest cost via the Internet [8] and require very little effort to install. In addition, high-level 3D APIs such as OpenGL [9] now come bundled with compilers and do not require an additional expenditure.

2. Current Practice

The availability of packages and high-level APIs facilitates the inclusion of more advanced concepts in introductory courses [10]. Packages can provide motivation [11] and allow students to experiment with more advanced algorithms without the necessity of implementing all of them [12][13]. For students entering the work force, the use of packages and high-level APIs more accurately mirrors what they will do on the job [14], because newly graduated practitioners will rarely be expected to create new algorithms [15], but will develop systems from a set of high-level tools.

Junior	4
Upper Level	10
Senior	2
Upper Level/Grad	3
Senior/Grad	1

Table 2: Level of Course

All of these influences affect curriculum. At SIGGRAPH 98, several computer graphics educators met to compare syllabi, and as a result of the discussion that ensued, decided to solicit syllabi from educators at a variety of institutions across the United States. Scott Grissom, Lew Hitchner, Bill Jones, Susan Reiser and I collected syllabi from 23 educators [16]. Of the 23 collected, two were strictly for graduate students and one was primarily an image-processing course. The following, originally reported in *Computer Graphics* [17], is a summary of the remaining 20 syllabi.

2.1. Level

As can be seen in table 2, instructors universally believe that an introductory computer graphics course requires a certain degree of maturity. No courses were targeted for students below the junior year, and several were crosslisted as graduate-level courses.

2.2. Previous Experience

One of the factors that places an introductory computer graphics course at an upper level is the extensive amount of background required for the course. Table 3 lists the immediate prerequisites for the surveyed courses. Most courses had more than one prerequisite. All of them required at least a year of programming courses or "programming fluency". In addition, almost every course required some background in mathematics, but the range in requirements is very wide, ranging from analytic geometry to multivariate calculus (Calculus III). However, the most prevalent math prerequisite was Linear Algebra.

Algorithms and Data Structures	12
CS2	4
Programming fluency	2
Software Engineering	2
Linear Algebra	11
Calculus I	2
Calculus II	1
Calculus III	1
Discrete Math	1
Analytic Geometry	1

Table 3: Immediate Prerequisites

2.3. Textbook, Software

A few textbooks featured prominently, namely Angel's *Interactive Computer Graphics: A top-down approach with OpenGL,* Hearn and Baker's *Computer Graphics,* and the classic Foley, van Dam, Feiner and Hughes' *Computer Graphics: Principles and Practice.* Two instructors did not use a text of any kind but created their own notes for classroom use.

There was also quite a bit of consensus on software selection, as Table 5 suggests. Quite a few instructors mentioned that the choice of computer and operating system was not important as long as students had access to OpenGL. Several instructors mentioned that they used OpenInventor or GLUT in addition to OpenGL. Two instructors failed to answer this question, which is why the total number of responses is less than 20.

2.4. Topics

The topics listed in Table 6 appear in the contributed syllabi. The range and number (38) of topics attest to the rich variety of backgrounds and interests of computer graphics instructors and include such diverse subjects as time-critical applications, animations, fractals, virtual reality and scientific visualisation. In general, instructors tend to present these more exotic topics near the end of the course.

OpenGL	12
C/C++	2
custom	2
VRML/Renderman/Java	1
Pascal	1

Table 5: Software used

Angel	8
Hearn and Baker	8
Foley, van Dam, Feiner, Hughes	2
notes	2

Table 4: Textbooks used

Although there are a large number of topics, five are mentioned in 70% or more of the courses, and three of these draw heavily on mathematics. The most prevalent, mentioned in 95% of the syllabi, is viewing transformations. (The thrust of the one course not mentioning this topic is building a ray tracer, which can obviate the need for a matrix representation of viewing transformations.) Another topic, 3D transformations, relies on the same matrix operations as does the viewing operations. Three quarters of the courses cover lighting or illumination models, which also requires a substantial math background.

Two additional topics are present in 70% or more of the syllabi. Most courses have an opening discussion covering graphics hardware and basic terminology, and they also include some discussion of the principles and implementation of interactive techniques.

The topic of object representation occurs in over half of the courses and includes a range of items, from representing points, line segments and polygons to representing surfaces of revolution. Although instructors mention curves and smooth surfaces fairly often, many add the proviso "as time permits." Rasterisation topics range from Bresenham's line- and circle-drawing algorithms to polygon fills. Another topic occurring fairly frequently is data structures that support graphics. This encompasses everything from display lists and polygon meshes to scene graphs, scene hierarchies and octrees.

A development which may not be apparent from the list of topics is the balance between 2D and 3D topics. In earlier times, 2D topics dominated, and a few 3D elements rounded out the course [2, 3, 7]. In more recent course offerings, 2D topics may be mentioned briefly in the opening two weeks, but most of the lectures discuss 3D topics [10, 11, 13, 14, 15].

Results of the syllabus study and discussions at SIGCSE 99 [18] have produced some insights that may provide a framework for developing courses. These include:

viewing transformations/ camera	19	math foundations/review	5
hardware/terminology	15	2D transformations	4
lighting models	15	3D rendering	4
3D transformations	14	fractals	4
user interaction	14	co-ordinate systems	3
object representation	11	photorealistic methods	3
shading models	11	antialiasing	2
color/color models	10	hidden-line removal	2
curves	10	pipeline details	2
hidden-surface removal	10	virtual reality	2
rasterisation	9	VRML	2
implementation particulars	7	2D ray tracing	1
supporting data structures/models	7	computational geometry	1
texture mapping	7	global illumination	1
clipping	6	interactive internet applications	1
ray tracing	6	projections	1
surfaces	6	scientific visualisation	1
2D drawing	5	shadows	1
animation	5	time-critical applications	1

Table 6: Course Topics

- Computer graphics is inherently 3D and courses should be also.
- The fundamental subject of a computer graphics course is geometry and how it is expressed in computational terms. Thus, geometry is a major part of the introductory course. Geometry is expressed in terms appropriate to the field, such as co-ordinate systems, transformations, and surface normals. The basic shape is the triangle. The mathematics of curved surfaces is typically treated in a more advanced course.
- Besides geometry, computer graphics is about light and surfaces, and about developing algorithms to simulate their interplay. Courses need to include material about light and surface properties and discuss the distinction between the ways various algorithms visually present light and surfaces.
- Computer graphics is intrinsically visual, and even the most technically oriented graphics practitioner must be aware of the visual effects of algorithms. Unlike other areas of computer science, algorithms must be considered not only for time and memory usage, but also for their visual effect.
- Computer graphics has matured to a state in which there are high-level APIs and packages that support all the fundamental concepts needed for early work. Courses should be built upon this kind of high-level approach.
- Computer graphics should be interactive. Courses should include interactive projects and cover event-driven programming.

3. Future Possibilities

Both the literature and the surveyed syllabi indicated that there is a rich variety of contexts in which computer science faculty teach introductory graphics. These include:

- an elective for computer science majors. Most likely students will not pursue graphics as a career. [6, 7, 15]
- a preparation for graphics practitioners. Students will be seeking a job in the graphics field upon graduation. [5, 7, 15, 18]
- a preparation for additional coursework in computer graphics. [7, 10, 12, 15, 18]
- a service course for other majors. [6, 18, 19, 20]

Due to the large variance of institutional resources, student preparation and student goals, attempts to create a single curriculum to meet all needs will be counterproductive. Instead an approach similar to the one chosen by the ACM/IEEE-CS Task Force in developing Curriculum 2001 [21] may prove more effective. Their principles include identifying a relatively small set of core concepts and skills to be required of all students and taking into account the

specific problems and limitations facing computer science programs at typical institutions.

In addition to identifying topics, educators welcome the continuing process of identifying and organising relevant material (visualisations, demonstrations, teaching software and presentation tips) for inclusion in a repository such as www.education.siggraph.org. Perhaps an inventory of available materials will lead to an identification of areas of greatest need and could serve as a focus for future efforts.

4. Conclusion

The history of Computer Graphics is short, but full of excitement due in part to the blazing pace of technology innovations and the ever-expanding arenas of application. A special challenge to graphics educators is to embrace the process of re-examination and re-evaluation in the quest of serving our students in the best way we possibly can.

References

[1] ACM/IEEE-CS Joint Curriculum Task Force. *Computing Curricula 1991.* Association for Computing Machinery, February, 1991.

[2] M. R. Ohlson. The Role and Position of Graphics in Computer Science Education. In *Proceedings of the Seventeenth SIGCSE Technical Symposium on Computer Science Education*, pages 232-237, 1986.

[3] J. D. McGregor. An Introductory Course in Graphics. In *Papers of the Seventeenth SIGCSE Technical Symposium on Computer Science Education,* pages 222-224, 1986.

[4] J. E. Rager. Graphics Packages for Teaching Graphics. In *Papers of the Seventeenth SIGCSE Technical Symposium on Computer Science Education,* pages 225-231, 1986.

[5] M. M. Larrondo-Petrie, J. Bresenham, C. Laxer, J. Lansdown, G. S. Owen, Approached to Teaching Introductory Computer Graphics. *SIGGRAPH 94 Computer Graphics Proceedings, Annual Conference Series.* pages 479-480, 1994.

[6] J. R. Brown, R. P. Burton, S. Cunningham, and M. Ohlson. Varieties of Computer Graphics Courses in Computer Science. In *Papers of the Nineteenth SIGCSE Technical Symposium on Computer Science Education.* page 313, 1988.

[7] M. Palazzi, W. Carlson, R. Lusas, M. Schweppe, and M. Yanilmaz, Preparing for the Future. *Computer Graphics.* Vol. 23 No. 5, pages 295-332, December 1989.

[8] Wolfe, 3D Freebies: A Guide to High Quality 3D Software Available via the Internet. *Computer Graphics* Vol. 32 No. 2 pages 30-33, May 1998.

[9] OpenGL Architecture Review Board. *OpenGL Programming Guide.* Addison-Wesley, 1993.

[10] G. S. Owen. Teaching Introductory and Advanced Computer Graphics Using Micro-Computers. In *Papers of the Twentieth SIGCSE Technical Symposium on Computer Science Education.* pages 283-287, 1989.

[11] K. Karpouzis and S. Kollias. The Rendering Pipeline in the Classroom: A Diversified Approach. In *Proceedings of the 6th annual conference on the teaching of computing (ITiCSE '98).* pages 139-142, 1998.

[12] G. S. Owen. Teaching Computer Graphics Using RenderMan. In *Papers of the Twenty-Third SIGCSE Technical Symposium on Computer Science Education.* pages 304-308, 1992.

[13] R. Klein, F. Hanisch, and W. Strasser. Web-Based Teaching of Computer Graphics: Concepts and Realisation of an Interactive Online Course. *SIGGRAPH 98 Conference Abstracts and Applications.* pages 88-93, 1998.

[14] L. H. Tichenor. Inexpensive Advanced Graphics Applications for the CS Majors Graphics Class. In *Papers of the Twenty-Sixth SIGCSE Technical Symposium on Computer Science Education.* pages 191-194, 1995.

[15] S. Grissom, J. Bresenham, B. Kubitz, G. S. Owen, D. Schweitzer. Approaches to Teaching Computer Graphics. In *Papers of the Twenty-Sixth SIGCSE Technical Symposium on Computer Science Education.* pages 382-383, 1995.

[16] OpenGL: Agent of Change or Sign of the Times? *Computer Graphics,* pages 29-31, November 1998.

[17] R. Wolfe. A Syllabus Survey: Examining the State of Current Practice in Introductory Computer Graphics Courses. *Computer Graphics.* Vol. 33 No. 1 pages 32-33. February 1999.

[18] L. Hitchner, S. Cunningham, S. Grissom, and R. Wolfe. Computer Graphics: The Introductory Course Grows Up. In *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science*

Education. pages 341–342, 1999.

[19] R. Geitz. Algorithms and Images: Computer Graphics as an Introduction to Science. In *Papers of the Twenty-Second SIGCSE Technical Symposium on Computer Science Education.* pages 82-86, 1991.

[20] D. House and D. Levine. The Art and Science of Computer Graphics: A Very Depth-First Approach to the Non-majors Course. In *Papers of the Twenty-Fifth SIGCSE Technical Symposium on Computer Science Education.* pages 334-338, 1994.

[21] E. Roberts, R. LeBlanc, R. Shackelford, and P. J. Denning. Curriculum 2001: Interim Report from the ACM/IEEE-CS Task Force. *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education.* pages 343–344, 1999.