

Curriculum for a Course on Scientific Visualization

Martin Rotard, Daniel Weiskopf, and Thomas Ertl

Visualization and Interactive Systems Group, University of Stuttgart, Germany
{rotard, weiskopf, ertl}@vis.uni-stuttgart.de

Abstract

In this paper we present a curriculum for an advanced graduate course on scientific visualization. Based on our experiences with ten years of teaching this field, we have come to a stable list of topics that could also serve as a basis for similar courses at other universities. Our goal is to provide students with concepts and a firm mathematical foundation, as well as technical aspects of algorithms. Practical skills in programming visualization algorithms, using commercial visualization tools, and applying methodologies and techniques to new problems are taught in accompanying exercises.

Keywords: Curriculum, scientific visualization, computer graphics

1. Introduction

Scientific visualization deals with all aspects that are connected with the visual representation of data sets from scientific experiments or simulations to achieve a deeper understanding or a simpler representation of complex phenomena. Therefore, visualization tools have become an important element in many applications in engineering, science, and medical imaging, and scientific visualization has evolved to a mature research field. The growing number of participants at the annual IEEE Visualization Conference, which was established in 1990, is an indication for the increasing relevance of scientific visualization.

We see a gap between the need for an increasing number of experts in scientific visualization on the one hand, and a lack of an officially adopted curriculum by professional computer science organizations at the other hand. A related problem is that a comprehensive textbook is still missing. Nevertheless, scientific visualization is taught at many schools throughout the world, and these courses apparently have a large overlap in topics. The goal of this paper is to present a curriculum for scientific visualization that has a very long tradition (with respect to computer graphics) and, as we think, has a potential to be relevant for many more years to come. We report on ten years of teaching experience with our course on scientific visualization that is designed for graduate students with a major in computer science or a related major. The main contribution of this paper is a well-established curriculum that is suitable for a one-semester

course. We describe the structure of the course by an exhaustive list of keywords and topics so that others can adopt our curriculum. We also report on our experiences with the course (Section 3).

2. Curriculum

In this section, we describe the structure and curriculum of the course that is designed for graduate students in computer science or a related major. The proposed length is one semester with three hours of lecture and one hour of exercises per week. Prerequisites are a working knowledge in undergraduate mathematics (in particular, matrix and vector math, calculus, and numerical methods for ordinary differential equations), basic programming skills in languages like C, C++, or Java, and an introductory course on computer graphics. It is assumed that students are familiar with basics of 3D computer graphics (rendering pipeline, transformations via homogeneous coordinates, Phong model) and with OpenGL [OWN*99]. At the University of Stuttgart, an introductory course on “Graphical-Interactive Systems” lays a basis in computer graphics and human-computer interaction. Alternatively, a typical introductory course on computer graphics, as it is offered at many other universities, could serve as an equally adequate basis.

The goal of our course is to teach all relevant concepts and techniques of state-of-the-art scientific visualization. In addition to a firm theoretical foundation, we emphasize the

| | |
|--|---|
| <ul style="list-style-type: none"> ● Introduction <ul style="list-style-type: none"> – Definitions and goals – History – Examples ● Visualization pipeline <ul style="list-style-type: none"> – Data sources – Pipeline – Sources of error ● Data representation <ul style="list-style-type: none"> – Domain and data values – Grids (unstructured, structured) – Scattered data – Classification schemes ● Filtering and data reconstruction <ul style="list-style-type: none"> – Voronoi diagrams and Delaunay triangulation – Differentiation and interpolation on grids – Interpolation without grids – Fourier transform (sampling theorem, reconstruction, frequency filtering) ● Traditional mapping techniques <ul style="list-style-type: none"> – Diagram techniques – Function plots and height fields – Isolines – Color coding – Glyphs and icons ● Indirect volume visualization <ul style="list-style-type: none"> – Marching cubes (original MC algorithm, asymptotic decider, dividing cubes, discretized MC) – Marching tetrahedra – Acceleration techniques (octree-based isosurface extraction, range query) | <ul style="list-style-type: none"> ● Direct volume rendering <ul style="list-style-type: none"> – Classification (transfer functions, segmentation) – Volumetric shading – Slicing – Ray casting (image-space approach, acceleration techniques) – Texture-based volume rendering (stack of 2D textures, viewport-aligned slicing with 3D textures) – Shear-warp (implementation by Lacroute) – Splatting – Projection of tetrahedral grids (Shirley-Tuchman) – Optical model for volume rendering (equation of transfer for light, volume rendering integral, front-to-back and back-to-front compositing, maximum intensity projection) ● Vector field visualization <ul style="list-style-type: none"> – Basic math of vector fields (vector calculus, Hodge-Helmholtz decomposition) – Arrows and glyphs – Particle tracing (characteristic lines, geometric mapping methods based on particle tracing, numerical integration) – Particle tracing on grids (P-space, C-space) – Line integral convolution (LIC, Fast LIC, OLIC) – Topology-based visualization (critical points, classification, separatrices) – 3D vector fields (geometric particle tracing methods, texture-based techniques) ● Tensor field visualization <ul style="list-style-type: none"> – Diffusion tensors (sources of data from medical imaging, eigenvector decomposition, classification of isotropy / anisotropy) – Hue balls and lit tensors – Hyperstreamlines and tensorlines |
|--|---|

Figure 1: Curriculum for a course on Scientific Visualization.

need of practical skills and therefore equip students with the ability to solve visualization tasks by programming in C++ and OpenGL, and by using IRIS Explorer [Num01]. We have the objective that our course should lead to a level of education that allows students to start conducting self-responsible research for a M.S. or Ph.D. thesis in scientific visualization.

Our course focuses on nine main topics that are briefly outlined in what follows. Figure 1 shows a detailed list of all topics and subtopics. Our curriculum is related to similar courses at other universities, as shortly summarized in [Dom03].

Our curriculum starts with a definition of scientific visualization, its goals, and a historical overview. The following

chapter introduces the visualization pipeline, which consists of filtering, mapping, and rendering steps. We use the visualization pipeline to identify and assign different visualization tasks. The third chapter deals with data representations and grid types. The dimensionalities of the domain and the data values are crucial to classify the large body of mapping techniques that are discussed later on in the course. The next chapter contains mathematical techniques and algorithms for the filtering step of the visualization pipeline, especially interpolation methods. The remaining—and largest—part of the course focuses on the mapping step of the visualization pipeline. We begin with traditional approaches (diagrams, color coding, or icons). The following topic is the visualization of 3D scalar fields, which is split into indirect volume vi-

sualization by means of geometrically extracted isosurfaces and direct volume rendering. Following the classification of mapping techniques according to the dimensionality of data values, the following chapter contains approaches for vector fields. Finally, the course ends with topics of tensor field visualization.

Throughout the course, typical visualization examples are included give a motivation and links to real-world applications, e.g., the visualization of finite-element simulations from the automotive industry, medical CT or MRI data, and 3D vector fields from computational fluid dynamics.

3. Experiences and Teaching Practice

We have been teaching this course since 1994, starting with lectures given at the University of Erlangen-Nuremberg. Since 1999 this course has been regularly offered at the University of Stuttgart. Today, the course takes place each spring semester and comprises three hours of lecture and one hour of accompanying exercises for a time span of thirteen weeks. A course is typically attended by 15–30 students. Most students have their major in computer science, software engineering, or information technology, while some students take this course as a selective subject in a related major (e.g., mathematics or engineering). We have experience in teaching the course in German and in English. Since 2001 we have exclusively been teaching in English to offer the course as a part of the international M.S. program in Information Technology.

3.1. Lecture

The lecture is mainly based on a presentation of Powerpoint slides with a video projector. We have prepared some 450 slides for the whole course. Where possible and adequate, additional teaching material is employed. We make frequent use of interactive Java applets to illustrate complex concepts. To avoid the time-consuming development of applets, we use existing programs available on the web, for example by the University of Tübingen [HS02, HS03]. Electronic films with animations of typical visualizations are included in the presentations to motivate and illustrate visualization algorithms by corresponding applications. In addition to modern multimedia contents, we still rely on traditional chalk-on-blackboard techniques to develop most mathematical ideas and derivations. This combination of different teaching styles is supported by the technical environment of our lecture hall, which allows us to simultaneously use the blackboard and a slide projection. We have good experience with using different presentation styles because they can be adapted to different types of contents to be taught.

Unfortunately, we face the problem that there is not yet an established textbook that could be used for our lecture.

Copies of the slides are provided as one source of information for the students. We also supply additional reading material with more background information, such as book chapters and conference or journal papers. References are included on the slides to direct students to the corresponding reading material and, in particular, help them to find detailed information for self studying in preparation for the exam. For the last two years we have been working on an extended script that contains more descriptive text than the slides but is not yet a complete textbook. We plan to use this script during the upcoming spring 2004 semester to provide students with a self-contained learning environment.

3.2. Exercises

One exercise hour per week accompanies the lecture. Depending on the number of students, we have one or two exercise groups so that each group has not more 20 students. In this way, a personal relationship between tutors and students can be established and students can be encouraged to participate actively. Homework assignments are handed out each week; handed-in assignments are checked by the tutors and returned to the student just before the exercise hour. During the exercise hour, students present their solutions in front of the class. A discussion about the results is explicitly wanted and encouraged by the tutors. We allow students to solve and present the solutions of the assignments in groups.

Different types of assignments are employed to achieve different didactic goals. Mathematical and theoretical concepts play an important role in the course and are mainly practiced by traditional assignments with pen-on-paper solutions. Programming assignments allow students to improve their practical coding skills; we use C++ as programming language and OpenGL as graphics API. Because it would be too time-consuming to have students develop all relevant visualization algorithms from scratch within programming assignments, we also employ practical exercises with a higher-level commercial visualization tool—IRIS Explorer [Num01]. In this way, a large variety of typical visualization problems can be addressed and students gain familiarity with usefully combining filtering, mapping, and rendering modules within a visual programming environment for the visualization pipeline. To keep the homework assignments interesting to the students, different types of assignments are combined on each homework sheet.

3.3. Resources on Course Web Page

All relevant information is distributed to students on a course web page. This web page contains copies of the lecture slides, homework assignments, test data sets, example solutions, additional reading material, and up-to-date announcements.

3.4. Exams

Student assessment is based on an oral exam at the end of the course. The exam takes 30 minutes if a student is tested only for this course. As another option, students may take a combined 45 minutes oral exam for this course and another graduate course.

3.5. Evaluation

All courses at the University of Stuttgart are regularly evaluated via a university-wide evaluation system. In five questionnaires, students are asked for their opinion with respect to the contents of the lectures and the exercises, the teaching methods, the teaching environment, and their satisfaction with the course. There is also a qualitative part in which constructive critic can be given in plain text. While we try to constantly improve the course by incorporating students' feedback, the overall structure has reached a stable state. The evaluation shows that students are satisfied with the contents and the teaching methods used in the course.

4. Conclusions

We have proposed a curriculum for an advanced course on scientific visualization. Our goal is to teach concepts and a firm mathematical foundation upon which technical aspects of algorithms can be built. This approach has led to a very stable curriculum over the last ten years because only minor updates with respect to new mapping or rendering techniques had to be included (e.g., pre-integrated volume rendering or diffusion tensor visualization approaches). Most other parts describe a stable knowledge base of mathematical concepts and basics visualization methods. From this experience we believe that our curriculum will stay valid even in many more years to come and thus could serve as an origin for courses at other universities.

References

- [Dom03] DOMIK G. (Ed.): *Draft report on an ACM SIGGRAPH Curriculum for Visualization*. 2003. <http://www.uni-paderborn.de/cs/vis>.
- [HS02] HANISCH F., STRASSER W.: Highly interactive web-based courseware. In *Proc. of CGE02 - Eurographics/SIGGRAPH Workshop on Computer Graphics Education* (2002), pp. 31–35.
- [HS03] HANISCH F., STRASSER W.: Adaptability and interoperability in the field of highly interactive web-based courseware. *Computer & Graphics* 27, 4 (2003), 647–655.
- [Num01] NUMERICAL ALGORITHMS GROUP LTD: IRIS Explorer, 2001. http://www.nag.co.uk/Welcome_IEC.html.

[OWN*99] OPENGL ARCHITECTURE REVIEW BOARD, WOO M., NEIDER J., DAVIS T., SHREINER D.: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*, third ed. Addison-Wesley, Reading, MA, USA, 1999.