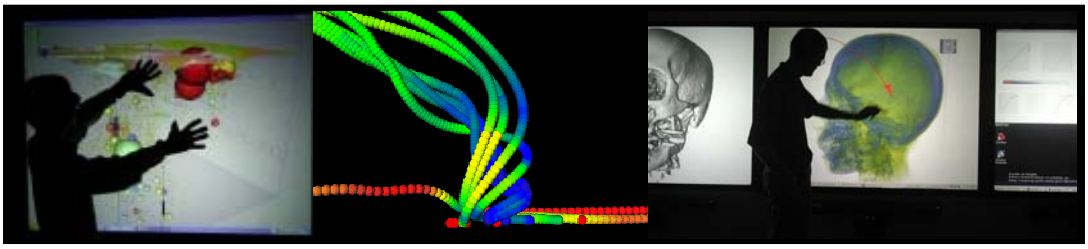
Scientific Visualization
SE207

Course Syllabus



Course Overview

This course introduces visualization techniques for various types of measured and computer-simulated data. Typical applications for these visualization techniques include the study of static and dynamic surface or volume models found in structural, mechanical, aerospace and biomedical engineering, earth system science and medicine.

Course Objectives

Visualization transforms large quantities of raw data into graphical representations that exploit the outstanding visual processing capability of the human brain to detect patterns and draw inferences. The primary objective of this course is to learn about the fundamental principles of scientific visualization. A comprehensive introduction to the theory and its application towards solving hands-on research problems will be provided. You will learn to design and implement a variety of 2D and 3D visualization tools for the analysis of scalar and vector field data, considering steady and time-varying cases.

Concepts learned in SE 207, such as meshing, iso-contouring, iso-surfacing and volume rendering will be applied to a range of data sets pervasively encountered in the engineering, biomedical and earth-science domains, with some of the data coming directly from different UCSD research groups. You are encouraged to study topics in the fields of image processing, visual perception, mathematics, and parallel computing in order to derive more efficient visualizations strategies. Topics covered in this course include:

I. Review of essential visualization algorithms

II. Scalar field visualization
- A. Multiple, arbitrary slicing surfaces
- B. Transparency paradigms
- C. Multiple, transparent isosurfaces
- D. Internal-based contouring

III. Volume visualization
- A. Review of Levoy's algorithm
- B. Sabella's algorithm
- C. Splatting

IV. Vector field visualization
- A. Ray casting vector fields
- B. Topological visualization
- C. Features in vector fields

V. Data compression and reduction
   A. Pyramid hierarchies for structured data
   B. Approximation techniques for hierarchies of fields
   C. Data reduction methods

VI. Animation for visualization
   A. Approximation methods for animation
   B. Animation of field data
   C. Animation of image space data

VII. Tesselation methods
   A. Voronoi diagrams
   B. Delannay triangulations

Prerequisites

In this course you will primarily use C/C++ and OpenGL to study the theoretical foundation and techniques of scientific visualization. You should have a strong working knowledge in mathematics (linear algebra, trigonometry, etc.), experience with a programming language and a basic understanding of algorithms and data structures.

Course Organization

First Day of Class:	04/03/2007	Section ID:	588242
Lectures:	T/TH 14:00-15:20	Location:	CENTR 202
Lecture:	3 hours	Laboratory:	3 hours
Credit Units:	4		
Homepage:	http://vis.ucsd.edu/courses/se207		

Instructor Information:

Prof. Falko Kuester — Email: fkuester@ucsd.edu
2302 Atkinson Hall (Callit2) — Homepage: http://vis.ucsd.edu/~fkuester
Office Hours T/TH TBA

Course Outcomes

You will design algorithms for the analysis and visualization of scientific and engineering data sets. In particular, algorithms and techniques for the analysis of mechanical engineering, aerospace engineering, ocean and atmosphere modeling, and medical imaging data will be studied and developed.

Resources

All course material will be provided on a the course web site including lecture notes, laboratory notes, useful links, recommended references, time schedule, contact information for faculty and TAs, guidelines for projects, coding standards and more (http://vis.ucsd.edu/courses/se207).

Textbook

The following two books are recommended for this course. The textbook covers all major topics that will be covered in this course and will provide a valuable resource for further advanced study. The second book listed below is the official OpenGL programming guide, which covers most of the OpenGL related topics, required for the successful completion of the course projects.

G. M. Nielson, B. Shriver, Visualization in Scientific Computing, IEEE Computer Society Press

OpenGL Architecture Review Board, Dave Shreiner, Mason Woo , Jackie Neider, Tom Davis
OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2 (5th Edition)

Assignments, Exams and Grading

The final course grade will be based on projects and the final. The projects, as a major part of this course, will contribute 80% to the final grade. Assignments have to be turned in electronically (see details on course web page).

	Grade [%]
Project 1	20%
Project 2	20%
Project 3	20%
Project 4	20%
Final	20%
Total	(100%)

Each project will be graded based on completeness, correctness, your understanding of the algorithms and implementation (80%), and the style (20%). Style considerations include the user interface design, the written project documentation and coding standard compliance. All projects will be graded on a 100-point basis.

Category	[%]
Completeness & correctness	80%
User Interface	10%
Documentation	5%
Coding standard compliance	5%
Total	(100%)

Only programs that can be compiled and executed on the laboratory machines will be graded. Incomplete projects will only be graded if a detailed description of the available and missing functionality is provided.

Each class is different, and no absolute grading scheme can be defined in advance. However, the following guarantees will always be made

90%	80%	70%	60%	50%
A	**B**	**C**	**D**	**F**

A grade of ‘Incomplete’ is not given unless extreme circumstances are presented.

Late Policy

To receive full credit on a project, the project must be turned in on the official due data. Projects that are turned in late will be subject to a 10-point daily penalty. A weekend counts as one day.

Course Load

The projects require significant effort and in general substantial time commitment, specifically during the final weeks of the quarter. Keep this in mind when you plan your total course load. If you plan on taking other courses that require finishing a project by the end of the quarter, you should consider this in your course strategy.

Is this the right course for me?

SE207 does require C, C++ programming skills and a good background in mathematics. All of the assignments and projects require programming in combination with OpenGL as the graphics API of choice. OpenGL knowledge is desirable before taking this class. Most importantly, a certain addiction to programming and familiarity with UNIX/Linux or Windows environments are important. If you are not familiar with these topics, you should be prepared to spend a considerable amount of time on acquiring these skills that are crucial for the implementation of the individual group projects. Lectures and lab sessions present a substantial amount of technical information required for the understanding of the material and successful development of the projects. You should not only be interested in the fascinating aspects of scientific visualization, but also be willing to learn and apply the concepts. The goal of the course is to do scientific visualization and not just to learn about it. The completion of the course projects is a rewarding experience and a valuable step towards applying your cumulative theoretical knowledge in mathematics, physics and other engineering disciplines.

Academic Integrity

Academic integrity (honesty) is taken very seriously. It is your responsibility to be familiar with UCSD's academic honesty policies. Please take the time to read the current UCSD Senate Academic Honesty Policies. Please note that any form of academic dishonesty could result in a grade of “F” in the class in addition to disciplinary action from the department or university. Detailed information about the Academic Senate Policy on Academic Honesty can be accessed through the references section on the course web page.

However, we encourage cooperation within well-defined bounds. This is a very challenging and work intensive course that will require you to refine and improve your mathematics, computer language and programming skills. At times it is difficult to learn all of the necessary programming techniques and tricks, implementation philosophies and debugging strategies on your own. I encourage you to draw from different sources including your classmates to improve these skills and to learn about available tools.

At the same time all assignments (source code) that you turn in has to be your own. You are allowed to talk to other students, the TAs or the instructor, share ideas, strategies and design philosophies, but in the end **do your own work and write your own code**. This means, you may **not** use material/code that anyone other than yourself has written. Material/code explicitly forbidden includes code taken from the web, from books or from any source other than yourself.

The only exception to this rule is that you should feel free to use any of the routines that are provided by the instructor or TA.

Laboratory Projects

The programming projects for this class are chosen to enhance the lecture material in the course.