The Course at a Glance

Prerequisites

CSC212, MTH215 and MTH243
I know that the URI catalog lists CSC305 as a prerequisite, but CSC212 is the real CS prerequisite for this course (and should soon be the official one as well). It might come to you as a shock that linear algebra is used a lot in computer graphics. It would be a good idea for you to refresh your (fond, I am certain) memories of matrices in the next few days. MTH243 material will be needed when we start talking about parametric models of curves and surfaces, and kinematics (animation).

Objectives

This course will concentrate exclusively on the geometric and kinematic aspects of computer graphics. At the end of this course, you should be capable of

- Constructing a geometric model of a simple scene;
- Implementing the objects of this scene, using OpenGL, glut, and glu primitives;
- Controlling, interactively or automatically, the displacements of objects, including the camera(s) that capture the scene;
• Building a simple graphical user interface for your application, using the glut library.

Other aspects of computer graphics are covered in the following courses

• ART304 and ART306: 3D Graphics design (using Maya 6)
• CSC492M-592R: scene graphs, scene rendering, GUI,
• CSC492: introduction to computational geometry

Equipment and References

Course textbook

This semester, I have not picked an official textbook. The main reason for this is that I am really lousy at following a textbook. I am going to post pdf versions of my notes, but you might still want to purchase one of the following fine books:

OpenGL references

The standard references are the "red book" and the "blue book". There is also a "green book" that is dedicated to OpenGL for Unix/X11, but I can’t see why anyone would submit themselves voluntarily to something as ugly as X11/Motif programming (use a cross-platform framework instead!).


These editions cover version 1.4 of OpenGL. Earlier versions of books (opengl v. 1.2) are available (see the “Links” page on the course’s web site)

My recommendation for an OpenGL reference is the following handy little book:


This book covers all the OpenGL we are going to need in this course and is small enough that you can carry it with you.

Mathematics for graphics

If you feel that you need a refresher on linear algebra or calculus, I have some links to nice online books on the course’s web site. A few good books that cover the mathematics commonly used in computer graphics and game development are


The following book covers all the material of the previous two books, but in a less compact form, plus a lot of more advance material on animation and rendering:


**Graphic API**

In your future careers as computer scientists and programmers, you will often have to learn a new API (Application Programming Interface) and use it in your projects. This course offers us a good opportunity to develop this important skill.

We will be using OpenGL + glut. This API developed by SGI is the most standard cross-platform API used both for professional 3D editing applications (Maya, SoftImage) and computer games (Quake II & III, Descent 3, etc.). On some Unix platforms, what is found is not OpenGL *per se* but MESA, which is an open-source project (encouraged/supported by SGI) that offers source-level compatibility with OpenGL.

Virtually all the OpenGL tutorial and code you are going to find out there is in C, with a huge and ugly “display” function doing all the work. I am going to insist that you use proper object-oriented implementations for all the assignments, with a nice hierarchy of graphic classes taking care of their own drawing. once you get the hang of it, you will see that it makes your life a lot easier.

**Programming environment**

All projects will be implemented in C++. If you only know Java, dont worry. We will help you transition to C++. C++ in not superior to Java or more grown-up. It is just that there a number of low-level things in OpenGL are better done in C++, and that I want you to understand. There are Java implementations of wrapper classes around OpenGL. One such library is GL4Java (now seemingly dead) which we used in the past in this class. Another, still active project is jogl. Otherwise, Java3D is a really powerful and well-designed graphic engine...

You can use any IDE or tool you desire to program your assignments, but what you hand in must compile and run with Eclipse 3.3. I will provide a header file that solve most path issues so that the same code can compile on any platform.

**Grading**

**Examinations**

One midterm examination will be administered around the end of October, at the usual class hours. All notes and printed documentation will be allowed during this examination.

Short quizzes will be held occasionally as well.
Assignments

We will have a new programming assignment nearly each week. For each programming assignment, you will be asked to provide complete documented (in Javadoc/Doxygen style) source code and a report. On “slow” weeks I may also hand out homework assignments that will deal with the theoretical aspects of the material covered in this course.

The Mix

The following coefficients will be used to compute the final grade:

- Homework Assignments (labs and ): 15%
- Programming Assignments: 40%
- Midterm 20%
- Quizzes 5%
- Final Project 25%

Rules of Conduct

Late Submissions

All assignments should be turned in on the day and time they are due. If an assignment is not turned in on time, a 10% penalty will be applied for each late day (a fraction of a day will count as a whole day). Assignments more than one week late will not be graded.

If the assignment asks for multiple deliverables (code, project, report, JavaDoc-style documentation) the date at which the last item was handed in will be used to determine the late penalty.

If you submit an assignment late, don’t just upload it to the server. You must also send an email to the TA (with Cc. to me) to inform him that the assignment is now complete and ready to be graded. You cannot just expect people to watch after you.

Cheating

Unless explicitly stated otherwise, all work should be done individually. Any evidence of cheating may result in expulsion from the class with a failing grade and will be brought to the attention of the Dean for disciplinary action.

You are strongly encouraged to discuss the assignments with other students, and try to figure them out together, but when comes the time of writing a report or developing code, you are expected to do it by yourself (or with other students from your group when the assignment explicitly mentions groups of 2 to 3 students).

Copying software from a book or from a web site without identifying the original author(s) of the work is plagiarism, which is a form of cheating.
Laboratory use

Any abuse of computer or software equipment will be brought to the attention of the appropriate authority for disciplinary action. Software piracy (the use, appropriation, or storage of illegal copies of software) is a form of abuse of the equipment and will be treated accordingly.

Attendance

Class attendance is not mandatory, although I strongly suggest that you make efforts to attend all classes. We will spend a significant part of the time in classes discussing about examples of programs, possible errors, bugs, etc. Don’t spend too much time taking notes during classes; rather make efforts to understand on-line what is going on. It will only get easier as we advance in the semester.
Lab attendance is mandatory and will factor in the evaluation.

Tentative Class Schedule

Week 1  
**Computer Graphics: Introduction**
- Classes of graphical systems
- OpenGL
- glut

Weeks 2-4  
**Basic 2D and 3D Geometry**
- Points and vectors
- Reference frames
- Transformations between reference frames
- Quaternions

Week 5  
**Simple Geometric Problems**
- Distance to a line, to a plane
- Intersection problems
- Collision detection

Week 6  
**Simple Animation**
- Path vs. trajectory
- Point kinematics
- Kinematics of solid objects
- Animation with glut

Week 7  
**Virtual Cameras**
- Orthographic camera
- Pinhole perspective camera
Week 8  
**Simple Geometric Models of Objects**  
- Discrete vs. continuous  
- Parametric vs. implicit models  
- Polygonal meshes  
- Normal vectors

Weeks 9-10  
**Better Object Models**  
- Hierarchical models  
- Splines, NURBS  
- Texture maps  
- Bump maps and displacement maps

Weeks 11-12  
**Animation, Part II**  
- Key framing  
- Forward kinematics of articulated objects  
- Inverse kinematics of articulated objects

Week 13  
**Misc. Problems**  
- Collision detection  
- Optimization