CSCD18: Computer Graphics

Professor: Leonid Sigal
lsigal@utsc.utoronto.ca
ls@cs.toronto.edu
Office: Room SW626
Office hours: Monday 12:00-1:00pm (or send email for an appointment)

Teaching Assistant: Alexander Wong
wongam@cs.toronto.edu
Office hours: TBA as needed

Lectures: Monday at 10:00-11:00am in BV-361
Wednesday at 10:00-11:00am in BV-363
Tutorials: Monday at 4:00-5:00pm in SW-221
Website: http://www.cs.toronto.edu/~ls/teaching.html

Course Description:

This course introduces the basic concepts and algorithms of computer graphics. It covers the basic methods needed to model and render 2D and 3D objects, including much of the following: graphics displays, rastering, parametric representations, curves and surfaces, geometrical optics, affine and perspective transformations, visibility, illumination and reflectance models, radiometry, energy transfer models, parametric representations, curves and surfaces, texture mapping, ray tracing, graphics toolkits, animation systems. The course also introduces the students to standard graphics packages like OpenGL and GLUT.

Course Prerequisites :

MATB24H and MATB42H and [CSCB09H or proficiency in C] and CSCB63H and [CSCC36H or CSCC50H] and [CGPA 3.0 or enrolment in a CSC subject POSt]

The student is expected to review material in order to be comfortable with elementary linear algebra, elementary geometry, and vector calculus (including partial differentiation).

Textbooks:

Currently, there is no textbook that reflects all the material covered in this class. Textbook material will be supplimented by in-class lectures and online notes (lecture slides and course notes). Lecture slides and online notes are required. Textbook sections listed next to each lecture also constitute required reading.

Required:
Fundamentals of Computer Graphics, Second Edition, by Peter Shirley, Michael Ashikhmin, Michael Gleicher, Stephen Marschner, Erik Reinhard, Kelvin Sung, William Thompson, Peter Willemsen.

Recommended:
OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 2.1 (6th Edition), by OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis.
Older version is available on-line at http://fly.cc.fer.hr/ unreal/theredbook/.

OpenGL(R) Reference Manual: The Official Reference Document to OpenGL, Version 1.2 (3rd Edition), by Dave Shreiner.

Older version is available on-line at `http://www.glprogramming.com/blue/index.html`.

Reference:

There are several other excellent textbooks on computer graphics that can serve as references for the course. Mainly,

- Alan Watt. 3D Computer Graphics, Third Edition, Addison Wesley, 1999.
- F. S. Hill, S. M. Kelley. Computer Graphics Using OpenGL, Third Edition, Prentice Hall, 2001.
- J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. Computer Graphics: Principles and Practice, 2nd Edition, in C , Addison Wesley, 1990.
- D. Hearn and M. P. Baker. Computer Graphics, Third Edition, Addison Wesley, 2003.

Finally, we will have a set of online notes to complement the textbook and lectures. These online notes will be an excellent reference for the material in the course, further readings, related openGL code, and further exercises. The online notes and lecture slides will be available from the course website as they are developed.

Background Material:

It is expected that students will take it upon themselves to ensure they have a reasonable background for this course. Roughly, the background material we assume is as follows (readings refer to the course text by Shirley):

- Introduction: (Chapters 1)
- Basic Geometry and Linear Algebra: (Sections 2.1–2.4, Chapter 5)
- Rudimentary Calculus in Multiple Variables (i.e., partial differentiation)

Course Material (readings):

Below are the readings from the course textbook that include the material covered in the course.

- 2D Curves and 3D Surfaces:
  (Sections 2.5–2.11)
- Raster Algorithms:
  (Sections 3.1, 3.5, 3.6)
- 2D and 3D Transformations:
  (Chapter 6)
- Camera Modelling (Geometrical Optics):
  (Chapter 7)
- Elementary Radiometry, Reflectance and Shading:
  (Sections 19.1, 9.1–9.2, 24.1–24.2)
- Backface, Hidden Surface Removal and BSP Trees:
  (Chapter 8, Section 12.1, 12.4)
- Texture Mapping:
  (Sections 11.1–11.4)
- Ray Tracing:
  (Chapter 10, Chapter 14, Sections 24.1–24.2)
- Parametric Curves and Surfaces:
  (Chapter 15)

- Animation:
  (Sections 16.1–16.5)

Note that the emphasis given to different topics material in the lectures will not be identical to the course textbook. So these readings will, in places, cover matieral in more or less detail than required for the course. The online notes that augment the textbook will be available from the course website.

Grading:

The assignments will be marked by the TA; the midterm will be marked by the professor and the TA; the final examinations will be marked by the professor. As a general rule, small matters of marking (apparent addition errors, questions about evaluation criteria, etc.) should be taken first to the TA (via email). More significant issues, or unresolved matters, are appropriate to take to the professor. The mark weighting is:

- Assignments – 50%
- Midterm – 15%
- Final – 35%

Important note: Students must obtain a mark of at least 35% on the final examination in order to pass the course.

Tentative Assignment Dates and Grading:

Assignment	Date Out	Written Due Date	Programming Due Date	Grade
#1	Wed Sept 17	Wed Oct 1	Fri Oct 10	15%
#2	Wed Oct 8	Wed Oct 22	Fri Oct 31	15%
#3	Wed Oct 29	Wed Nov 12	Fri Nov 21	10%
#3	Wed Nov 19	N/A	Fri Dec 2	10%

Assignment policies:

- Assignments 1, 2, and 3 will have a written component and programming component (assignment 4 will only have a programming component).
- Written assignments are due in the D18 drop box before class on the due date specified, which will always fall on a Wednsday. Written assignments must be written clearly (or typed). If the TA can't read the solution, points will be deducted accordingly.
- Programming assignments must be submitted electronically and will be due by 11:59pm on the due date specified, which will fall on a Friday (except for assignment 4).
- Assignment 4 will be due on the last day of classes (Dec. 1).

Programming Assignment Submission: Follow these steps when submitting your programming assignment:

- If you are not already working on `mathlab`, ftp all of the files making up your program to a separate directory on your `mathlab` account.
- Test your program. If you have developed your code on another platform, make sure it works as you expect on `mathlab` when compiled and executed.
- Submit all of your files using the command

  ```
  submit -N assignmentname cscd18f filename
  ```

  where assignmentname is A1 for assignment #1, A2 for assignment #2, etc., and filename is the name of the file to be submitted. To be safe, use a separate submit command for each file to be submitted. Wildcards are permitted, but be careful to specify the correct files if you use them.

Submit only the files we've asked for. In particular, since the marker will recompile your program, do not submit your executable files. Further documentation on the submit command may be found by typing 'man submit' on `mathlab`.

Late policy:

- For each day late, including weekends, 15 percent of the total possible points will be deducted (a day ends at the due time). For example, if a written assignment is due on Wednesday, no late penalty will be assessed if it is turned in by 9:59am on Wednesday. If it is turned in after 9:59am on Wednesday but before 9:59am on Thursday there will be a 15 percent penalty. Since the drop box is not going to be checked during the weekend (i.e. after 9:59am on Friday), it is your responsibility to submit the late written assignment by e-mail to TA before the designated time during the weekend. Otherwise, any assignment submitted to the drop box during the weekend will automatically loose th maximum of 75 percent of the total possible points.
- **No work will be accepted more then 5 days late**.
- Beyond this, no extensions will be granted on homework assignments, except in extreme cases (e.g. medical reasons). Please plan ahead.

System Details:

The language used for the assignments in the course is C++. This is the standard language for most industrial and research computer graphics programming. You are expected to be a competent programmer in C or C++ in this course. In addition to C++, you will be using OpenGL, a library of graphics classes and primitives. The first tutorial will review both C++ and OpenGL.

You will be using your `mathlab` account at UTSC for all of your program development. A C++ compiler and the OpenGL library are available on this platform. You may use other platforms for development (e.g. Visual Studio), however your final submissions must work on the Unix
t mathlab environment. You are on your own if you use other platforms, although we may provide some hints on cross-platform development with Windows and Unix if we have time and the relevant expertise.

Scholarly Conduct:

Plagiarism is a serious academic offence; the work submitted must be your own. If you have exchanged ideas with a fellow student and thus have answers which might be falsely construed as being plagiarised, you should state this.

- Tests and exams in this course must be strictly individual work.
- Assignments will be strictly individual work (except for Assignment 4, where groups may be allowed).
- **For programming components of an assignment:** Collaboration on a programming component by individuals (whether or not they are taking the class) is encouraged at the level of ideas. Feel free to ask each other questions, brainstorm on algorithms, or work together at a blackboard. Be careful, however, about copying the actual code for programming assignments or merely adapting others' code. This sort of collaboration at the level of artifacts is permitted if explicitly acknowledged, but this is usually self-defeating. Specifically, you will get zero points for any portion of an artifact that you did not transform from concept into substance by yourself. If you neglect to label, clearly and prominently, any code that isn't your own or that you adapted from someone else's code (from another student in the class, from a previous year's assignment, or from the web), that's academic dishonesty for the purpose of this course and will be treated accordingly.
- The principle behind the collaboration rule is simple: I want you to learn as much as possible. I don't care if you learn from me or from each other. The goal of artifacts (programming assignments) is simply to demonstrate what you have learned. So I'm happy to have you share ideas, but if you want your own points you have to internalize the ideas and them craft them into an artifact by yourself, without any direct assistance from anyone else, and without relying on any code taken from others (whether at this university or from the web).

- There are some circumstances under which you may want to collaborate with someone else on the programming component of an assignment. You and a friend, for example, might create independent parts of an assignment, in which case you would each get the points pertaining to your portion, and you'd have the satisfaction of seeing the whole thing work. Or you might get totally stuck and copy one subroutine from someone else, in which case you could still get the points for the rest of the assignment (and the satisfaction of seeing the whole thing work). But if you want all the points, you have to write everything yourself.

For purposes of this class, academic dishonesty is defined as:

- Any attempt to pass off work on a test that didn't come straight out of your own head.
- Any collaboration on artifacts in which the collaborating parties don't clearly and prominently explain exactly who did what, at turn-in time.
- Any activity that has the effect of significantly impairing the ability of another student to learn. Examples here might include destroying the work of others, interfering with their access to resources (eg. digital cameras), or deliberately providing them with misleading information.

Note: this policy has been choosen explicitly to be consistent with the course offered at St. George campus, taught by Kyros Kutulakos.