CMSI 371-01

COMPUTER GRAPHICS

http://myweb.lmu.edu/dondi/spring2012/cmsi371

Spring 2012 — Pereira 207
TR 3:00–4:15pm, 3 semester hours
Office Hours: TR 11am–12pm/4:30–5:30pm, or by appointment

John David N. Dionisio, PhD
e-mail: *dondi@lmu.edu,* AIM: *dondi2LMU*
Doolan 106; (310) 338-5782

Objectives and Outcomes

This course explores the computer science subfield of *computer graphics* — the study and development of algorithms for synthesizing, manipulating, and displaying visual information. Long after the course concludes, my hope is that you will:

1. **Know how visual information is modeled and represented digitally.**
2. **Understand how visual information is computed and manipulated in 2D and 3D.**
3. **Be able to use and develop computer graphics APIs in both 2D and 3D.**

In addition to the course-specific content, you are also expected to:

4. **Follow academic and technical best practices throughout the course.**

Prerequisites/Prior Background

Mastery of a programming language such as JavaScript, Java, or C; expert knowledge of data structure and algorithm design; some familiarity with object-oriented programming, computer hardware, and operating systems.

Materials and Texts

- Edward Angel and Dave Shreiner, *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, 6th Edition, Addison Wesley, 2012
- Dave Shreiner, Bill Licea-Kane, and Graham Sellers (The Khronos OpenGL ARB Working Group), *OpenGL Programming Guide*, 8th Edition, Addison Wesley, 2012 — *a.k.a. "the Red Book"*
- Randi J. Rost, Bill Licea-Kane, et al., *OpenGL Shading Language*, 3rd Edition, Addison Wesley, 2009 — *a.k.a. "the Orange Book"*
- Assorted handouts, articles, and sample code to be distributed throughout the semester

The following text is of general use, with Chapter 9 pertaining specifically to graphics on the Web:

- John David Dionisio and Ray Toal, *Programming with JavaScript: Algorithms and Applications for Desktop and Mobile Browsers*, Jones & Bartlett Learning, 2013 (!)

Additional information is also available on the web; do not hesitate to look for further sources of information regarding the concepts, techniques, tools, and paradigms that we will discuss.

Course Work and Grading

This course uses standards-based grading: your proficiency in each course objective is directly evaluated according to the outcomes shown on page 4 of this syllabus. Proficiency is measured according to the following key:

+	Advanced proficiency
|	Appropriate proficiency
/	Approaching appropriate proficiency
–	Needs practice and support
O	No basis for evaluation

Your submitted work is used to evaluate these outcomes (see below). Letter grades are then assigned as follows:

	+	|	/	–
A	**many**		*none*	*none*
B		**many**	*none*	*none*
C			**some**	*none*
D				**some**
F				**many**

A–, B+, B–, C+, and C– grades may be assigned based on "close calls" along the proficiency measure thresholds and qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course. You may inquire at any time about the proficiency measures that I currently have on record for you.

Homework

Homework consists of questions, exercises, and programming assignments to be given throughout the semester. Homework is one mechanism for demonstrating the proficiencies expected of the course. You will be given feedback on these proficiencies, and *may resubmit your homework throughout the semester* in order to improve upon them.

With great flexibility comes great accountability. First off, *you must submit your homework on time*. The assignment due date is encoded in the homework number. Note that one of the outcomes in this class is the *ability to meet deadlines* (*4f*). Late work will detract from this outcome (duh).

Quizzes and Tests

Some outcomes are best demonstrated by answering questions or doing exercises in class. These resemble traditional quizzes and tests, but, like homework, they are evaluated according to standards and do not produce a numerical score. *They are typically spontaneous and unannounced.*

Questions may include content-oriented elements as well as forward-looking, applicative portions (i.e., "use this knowledge to resolve this situation"). Tests are open-paper-everything; no sharing. "Open computer" might be allowed depending on the circumstances. You may neither solicit nor give help while a test is in progress. Late and/or missed tests are handled on a case-to-case basis; in all instances, talk to me.

Term Portfolio

Your accumulated homework and tests for the semester comprise the *term portfolio* — the final, definitive artifact that demonstrates the proficiencies you have reached. The term portfolio provides you with an opportunity to finish and polish the work done throughout the semester; it is how you show that you learned from your mistakes or improved on already established knowledge.

Throughout the semester, you are encouraged to improve your work based on received feedback, and show it to me for re-evaluation. Improvements in proficiency are recorded and give you a good idea of how your term portfolio will fare long before its final version is submitted.

The final version of your term portfolio is due on May 4. Late portfolios detract from outcome *4f*.

Extra Credit

In terms of standards-based grading, "extra credit" takes on a different meaning: it indicates work that, if successfully performed, would indicate advanced proficiency (+). Extra credit tasks may be assigned for either homework or the term portfolio. Accomplish them successfully to rack up those +'s. You do not need to perform extra credit work to show advanced proficiency; it merely demonstrates such proficiency more readily.

Version Control

Version control is an indispensable part of today's computer science landscape in industry, the academe, and the open source community. We use version control heavily in this course: make sure that you get the hang of it.

Attendance

Attendance at all sessions is expected, but not absolutely required. If you must miss class, it is your responsibility to keep up with the course work. Note that the last day to add or drop a class without a grade of W is January 13. The withdrawal or credit/no-credit status deadline is March 16.

University Policy on Academic Honesty

Loyola Marymount University expects high standards of honesty and integrity from all members of its community. All students are expected to follow the LMU Honor Code and Process, as stated in the *LMU Undergraduate Bulletin*.

Americans with Disabilities Act

Students with special needs as addressed by the Americans with Disabilities Act who need reasonable modifications, special assistance, or accommodations in this course should promptly direct their request to the Disability Support Services (DSS) Office. Any student who currently has a documented disability (physical, learning, or psychological) needing academic accommodations should contact DSS (Daum Hall, Room 224, x84535) as early in the semester as possible. All discussions will remain confidential. Please visit *http://www.lmu.edu/dss* for additional information.

Topics and Important Dates

Correlated outcomes are shown for each topic. Specifics may change as the course progresses. University dates (italicized) are less likely to change.

January	2D graphics with JavaScript and the `canvas` element *(1c, 2b)*; introduction to traditional animation (c/o Prof. Tom Klein) *(2a)*
January 13	*Last day to add or drop a class without a grade of W*
February	Graphics and memory *(1a)*; 2D graphics primitives *(1b, 3a, 3b)*; 2D and 3D graphics with JavaScript and WebGL *(1c, 2c)*; introduction to programmable shaders *(1c, 2c)*
February 27 to March 2	*Spring break; no class*
March	Object modeling *(1d, 1e)*; transforms *(1b, 2d, 3c)*; viewing and projection *(2e, 3d)*; clipping *(2f)*; hidden surface removal *(2g)*
March 16	*Withdraw/credit/no-credit deadline*
April	Lighting and shading *(2h)*; portfolio workshops *(3e)*
April 4 to April 6	*Easter break; no class*
May 4	*Term portfolios due*

You can view my class calendar in any iCalendar-savvy client such as Google Calendar or Apple iCal by subscribing to:

webcal://www.me.com/ca/sharesubscribe/1.9392690/M2CD-5-1-5B14D70A-E341-4026-A665-D391D97E01B8.ics

(I know, it's ugly; the link is also available on my web site for convenience.)

If necessary, this syllabus and its contents are subject to revision. Students are responsible for any changes or modifications announced in class.

Course Outcomes

1	**Know how visual information is modeled and represented digitally.**
1a	*Understand how computer graphics is represented at the bit level.*
1b	*Understand how computer graphics is represented geometrically.*
1c	*Understand how computer graphics is represented within a graphics library.*
1d	*Know what constructive solid geometry is and how it is implemented.*
1e	*Know what polygon meshes are and how they are implemented.*
2	**Understand how visual information is computed and manipulated in 2D and 3D.**
2a	*Be familiar with animation concepts and fundamentals.*
2b	*Know the parts of a computer graphics system.*
2c	*Know the phases and flow of a computer graphics pipeline.*
2d	*Understand graphics entities and transformations, and how they are modeled using points, vectors, and matrices.*
2e	*Understand how viewing and projection matrices are derived and used.*
2f	*Be familiar with the concept of clipping and clipping algorithms.*
2g	*Be familiar with the concept of hidden surface removal (HSR) and HSR algorithms.*
2h	*Understand how light and color are modeled and approximated.*
3	**Be able to use and develop computer graphics APIs in both 2D and 3D.**
3a	*Implement 2D graphics primitives such as line segments, circles, and polygon fills.*
3b	*Perform bit-level color manipulation.*
3c	*Implement a vector and matrix library.*
3d	*Implement transforms, projections, and camera functions.*
3e	*Implement a 3D graphics pipeline using programmable shaders.*
4	**Follow academic and technical best practices throughout the course.**
4a	*Write syntactically correct, functional code.*
4b	*Provide clear, appropriate inline documentation (i.e., comments).*
4c	*Write code that is properly indented and spaced for human readability.*
4d	*Use available resources and documentation to find any required technical or developer information.*
4e	*Commit to version control early and often, accompanying committed changes with informative messages.*
4f	*Meet all designated deadlines.*