CMSI 371-01

COMPUTER GRAPHICS

http://myweb.lmu.edu/dondi/spring2013/cmsi371

Spring 2013 — Pereira 207
TR 3:00–4:15pm, 3 semester hours
Office Hours: TR 10am–12pm, or by appointment

John David N. Dionisio, PhD
email: *dondi@lmu.edu*
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 be able to:

1. **Represent, model, and create visual information digitally.**
2. **Manipulate and display visual information in 2D and 3D.**
3. **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 and the Khronos OpenGL ARB Working Group, *OpenGL Programming Guide*, 7th Edition, Addison Wesley, 2009—*a.k.a. "the Red Book"* (if you spot the 8th edition, even better)
- 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

In addition, do not hesitate to look for further 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**	**few**	*none*
C			**some**	**few**
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. Late homework detracts from outcome *4f* (*Meet all designated deadlines*).

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 during a quiz or test. Late or missed tests are handled case-to-case; 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 for each course outcome. The term portfolio provides you with an opportunity to 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 may 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 10. Late portfolios detract from outcome *4f*.

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. The last day to add or drop a class without a grade of W is January 18. The withdrawal or credit/no-credit deadline is March 22.

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 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 *(1a, 2a)*; introduction to animation *(1c, 3a)*
January 18	*Last day to add or drop a class without a grade of W*
February	Graphics and memory *(1a)*; 2D graphics primitives *(3b, 3c)*; 2D and 3D graphics with JavaScript and WebGL *(1b, 1c, 2a, 2c)*; introduction to programmable shaders *(3e)*
March	Object modeling *(1b, 1c)*; transforms *(2a, 3b, 3d)*; viewing and projection *(2b, 3d)*; clipping and hidden surface removal *(2d)*
March 4–8	*Spring break; no class*
March 22	*Withdraw/credit/no-credit deadline*
March 27–29	*Easter break; no class*
April	Lighting and shading *(2c, 3e)*; portfolio workshops *(1a–3e)*
May 10	*Term portfolios due*

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course web site (it's too long to provide in writing).

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 Represent, model, and create visual information digitally.

1a	*…in terms of pixels and geometric primitives.*	With a few exceptions, these outcomes will be demonstrated within a single, cumulative "scene" program throughout the semester. More than in prior courses, assignments will incrementally build on previous ones. It will thus be more important than usual that you keep up and keep current.
1b	*…in terms of polygon meshes: vertices, edges, and faces.*	
1c	*…as a composition of multiple discrete objects (scenes).*	

2 Manipulate and display visual information in 2D and 3D.

2a	*Apply transforms to 2D and 3D objects.*	In the same way that the study of general-purpose data structures starts with the structures themselves, then goes into algorithms and operations on those structures, so goes the study of computer graphics. Learning objective 1 looks at structure; learning objective 2 looks at algorithms and computations.
2b	*Project 3D objects onto a 2D viewport.*	
2c	*Perform color and light computations.*	
2d	*Perform clipping and hidden surface removal (HSR).*	

3 Use and develop computer graphics APIs in both 2D and 3D.

3a	*Animate scenes in 2D and 3D.*	There is some overlap between these outcomes and the ones for learning objective 2. This is by design—the outcomes in objective 2 focus on understanding these computations conceptually, including doing them manually; the outcomes in objective 3 look at your ability to implement them in code.
3b	*Implement 2D graphics primitives such as line segments, circles, and polygon fills.*	
3c	*Perform bit-level color manipulation.*	
3d	*Develop a library of geometric primitives, operations, and matrix transformations.*	
3e	*Render a 3D scene using programmable shaders.*	

4 Follow academic and technical best practices throughout the course.

4a	*Write syntactically correct, functional code.*	Code has to compile. Code has to work. No errors, no bugs. Use unit tests as much as possible.
4b	*Demonstrate proper separation of concerns.*	This is the basis of good software design. It makes software easier to maintain, improve, and extend. Proper separation of concerns includes but is not limited to correct scoping of variables & functions and zero duplication of code.
4c	*Write code that is easily understood by programmers other than yourself.*	This outcome involves all aspects of code readability and clarity for human beings, including but not limited to documentation & comments, spacing & indentation, proper naming, and adherence to conventions or standards.
4d	*Use available resources and documentation to find required information.*	The need to look things up never goes away. Remember also that the course instructor counts as an "available resource," so this outcome includes asking questions and using office hours.
4e	*Use version control effectively.*	In addition to simply using version control correctly, effective use also involves appropriate commit frequency and descriptive commit messages.
4f	*Meet all designated deadlines.*	

Sample Standards Achievement Report

Based on these proficiencies, the student will get a B.

1	**Represent, model, and create visual information digitally.**	
1a	*…in terms of pixels and geometric primitives.*	+
1b	*…in terms of polygon meshes: vertices, edges, and faces.*	+
1c	*…as a composition of multiple discrete objects (scenes).*	+
2	**Manipulate and display visual information in 2D and 3D.**	
2a	*Apply transforms to 2D and 3D objects.*	+
2b	*Project 3D objects onto a 2D viewport.*	+
2c	*Perform color and light computations.*	|
2d	*Perform clipping and hidden surface removal (HSR).*	|
3	**Use and develop computer graphics APIs in both 2D and 3D.**	
3a	*Animate scenes in 2D and 3D.*	+
3b	*Implement 2D graphics primitives such as line segments, circles, and polygon fills.*	+
3c	*Perform bit-level color manipulation.*	+
3d	*Develop a library of geometric primitives, operations, and matrix transformations.*	/
3e	*Render a 3D scene using programmable shaders.*	|
4	**Follow academic and technical best practices throughout the course.**	
4a	*Write syntactically correct, functional code.*	+
4b	*Demonstrate proper separation of concerns.*	+
4c	*Write code that is easily understood by programmers other than yourself.*	+
4d	*Use available resources and documentation to find required information.*	+
4e	*Use version control effectively.*	|
4f	*Meet all designated deadlines.*	/