CMSI 371-01

COMPUTER GRAPHICS

http://myweb.lmu.edu/dondi/spring2015/cmsi371

Spring 2015—Doolan 222
TR 9:25–10:40am, 3 semester hours
Office Hours: TR 4:15–5:30pm, W 2–4pm, R 1:30–3pm, 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 WebGL*, 7th Edition, Addison Wesley, 2015
- 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 are assigned for outlier combinations between the above thresholds. Qualitative considerations (e.g., degree of difficulty, effort, class participation, time constraints, overall attitude) may improve proficiency measures. To resolve close calls, a quantitative calculation with 4, 3, 2, and 1 standing in for **+**, **|**, **/**, and **–** respectively will be used. You will receive feedback and proficiency updates after every assignment.

Term Portfolio

Your accumulated assignments for the semester comprise the *term portfolio*—the final, definitive artifact that demonstrates the proficiencies you have reached for each course outcome. It is how you show whether you have, indeed, accomplished the objectives of this course.

Final proficiencies for outcomes in Objectives 1–3 are based on their statistical mode, modulated by qualitative considerations such as degree of difficulty, cumulative nature of the material, or demonstrated progress by the student. Due to their best-practice nature, final proficiencies for outcomes in Objective 4 are based on their statistical *mean*, using the numeric mapping given previously. Incomplete portfolios are evaluated on a case-to-case basis.

2D Graphics

- A 2D parameterized sprite function library
- A tweened, 2D animated scene
- Simple image processing filters

Your work here affects your proficiencies for outcomes *1a*, *2a*, *2c*, *3a–3c*, and *4a–4f*.

3D Graphics

- A 3D object library with polygon meshes, object composition, and instance transformations
- Vertex and fragment shaders implementing commonly-used computer graphics algorithms and techniques
- An interactive 3D scene based on these libraries

Your work here affects your proficiencies for outcomes *1b*, *1c*, *2a*, *2b*, *2d*, *3a*, *3d*, *3e*, and *4a–4f*.

Resubmitting Work for Re-evaluation: Once Within Two Weeks of Feedback for the First $n - 2$ Assignments

Standards-based grading focuses ideally on achieving proficiency, not accumulating scores. Thus, all but the last two assignments may be resubmitted for re-evaluation *once within two weeks of receiving feedback* on it. I have no way of automatically knowing when you're finished with something, so *please notify me by email* when a submission is ready.

You must still submit all assignments by their respective deadlines—late work detracts from outcome *4f*. An assignment's number is its due date in *mmdd* format.

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.

Workload Expectations

In line with LMU's *Credit Hour Policy*, the workload expectation for this course is that for every one (1) hour of classroom instruction (50 scheduled minutes), you will complete at least two (2) hours of out-of-class work each week. This is a 3-unit course with 3 hours of instruction per week, so you are expected to complete $3 \times 2 = 6$ hours of weekly work outside of class.

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 16. The withdrawal or credit/no-credit deadline is March 20.

Academic Honesty

Academic dishonesty will be treated as an extremely serious matter, with serious consequences that can range from receiving no credit to expulsion. It is never permissible to turn in work that has been copied from another student or copied from a source (including the Internet) without properly acknowledging the source. It is your responsibility to make sure that your work meets the standard of academic honesty set forth in the *LMU Honor Code and Process*.

Special Accommodations

Students with special needs who require reasonable modifications or special assistance in this course should promptly direct their request to the Disability Support Services (DSS) Office. Any student who currently has a documented disability (ADHD, autism spectrum, learning, physical, or psychiatric) needing academic accommodations should contact DSS (Daum 224, x84216) as early in the semester as possible. All requests and 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, 3b)*
January 16	*Last day to add or drop a class without a grade of W*
February	Graphics and memory *(1a)*; 2D graphics primitives *(1a, 3a, 3c)*; 2D and 3D graphics with JavaScript and WebGL *(1b, 1c, 2a, 2b, 2c)*; introduction to programmable shaders *(3d)*
February 18–20	*Spring break; no class*
March	Object modeling *(1b, 1c)*; transforms *(2a, 3b, 3d)*; viewing and projection *(2b, 3b, 3d)*
March 20	*Withdraw/credit/no-credit deadline*
March 30–April 3	*Easter break; no class*
April	Lighting and shading *(2c, 3d)*; clipping and hidden surface removal *(2d)*; individual scene reviews *(1a–3d)*
May 8	*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.

Tentative Nature of the Syllabus

If necessary, this syllabus and its contents are subject to revision; students are responsible for any changes or modifications distributed in class or posted to the course web site.

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	*Be familiar with established algorithms such as clipping and hidden surface removal (HSR).*	

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

3a	*Develop a library of 2D and 3D objects.*	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; the outcomes in objective 3 look at your ability to implement them concretely.
3b	*Animate scenes in 2D and 3D.*	
3c	*Perform bit-level color manipulation.*	
3d	*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	*Use coding best practices, demonstrating principles such as DRY, proper separation of concerns, correct scoping of variables and functions, etc.*	This is the basis of good software design. It makes software easier to maintain, improve, and extend.
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 spacing & indentation, proper naming, presenting code in a manner that is consistent with its structure, documentation & comments when appropriate, 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 is a qualitative call between an A– and 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	*Be familiar with established algorithms such as clipping and hidden surface removal (HSR).*	|

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

3a	*Develop a library of 2D and 3D objects.*	+
3b	*Animate scenes in 2D and 3D.*	+
3c	*Perform bit-level color manipulation.*	+
3d	*Render a 3D scene using programmable shaders.*	|

4 Follow academic and technical best practices throughout the course.

4a	*Write syntactically correct, functional code.*	+
4b	*Use coding best practices, demonstrating principles such as DRY, proper separation of concerns, correct scoping of variables and functions, etc.*	+
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.*	/

This student reached advanced proficiency in 12 out of the 17 outcomes. Appropriate proficiency was reached in 4 out of 17, but proficiency was not reached in one outcome (*4f*).

The student might have been in the running for an A had work been submitted on time. Instead, the student is a close call between an A– and B+. Qualitative factors such as class participation will determine the final grade.

The student should have avoided the / for a guaranteed A-level grade. More +'s would have ensured the A rather than A–. More |'s rather than +'s would have taken the student to a B-level grade; more /'s would have taken the student to a C-level grade.

Totals

+	12
|	4
/	1
–	0
O	0