EECS 351-1:Introduction to Computer Graphics
Syllabus: Winter(Jan-Mar) 2016

Updated: 1/8/2016 3:08 AM

Course Description

First (and the only prerequisite) in a 3-course set surveying methods and theory of computer graphics.

EECS 351-1: Introduction to Computer Graphics:	3D Basics, WebGL, GLSL
EECS351- 2: Intermediate Computer Graphics:	Particles, Soft Things, & Ray Tracing
EECS395/495: Computational Photography Seminar:	4D Ray space, HDR, Coded Apertures, etc.

Prerequisites

EECS 214 (was 311) Data Structures or equivalent (or instructor's OK; please ask if unsure)

Goals:

CS 351-1 is an introductory but in-depth course on **computer graphics principles** for engineers and scientists. We don't teach you how to *use* graphics packages (Blender, Maya, Renderman, SketchUp, Unity, or even "three.js") because you can just read the manuals for that. Instead, we will teach you *what's inside* them, *how* it's computed, and *why*; you learn enough to write *new* graphics programs, ones with features not available in any commercial product. We use the most universally available graphical API spec (OpenGL / WebGL) for easy, uniform access to the graphics acceleration hardware in almost all computerized devices (from supercomputers to cell phones), and the GLSL shading language for programmable shading by parallel programming (Microsoft-proprietary DirectX & HLSL are similar).

CS351-1 also touches briefly on a wide variety of topics covered in greater depth in the later courses, including perception, shape modeling, animation, physical light transport, and depiction. When you finish, you'll understand both the (now deprecated) WpenGL 'rendering pipeline' and the much more flexible shading language pipeline of GLSL. You will know how to write interactive 3D programs in WebGL (and can learn machine-native OpenGL, GLSL or DirectX, HLSL easily), and will understand the computations underlying 3D graphics packages and current game engines.

The best way to learn in this course is to write short graphics programs to test ideas from class meetings and the assigned readings. You will then be ready to write each of three large 'Project' programs that determine most of your grade, and ready to demonstrate your program to all other students in class:

- **Project A – Moving Shapes:** assemble sets of colorful 3D shapes (per-vertex colors) and 3D transforms for jointed objects that move smoothly and respond to mouse and keyboard.
- **Project B –3D Views & Shading:** Extend transforms to steered, moving 3D perspective cameras, to movable 3D lights, to quaternion rotations, and to build a diffuse- lit, animated 3D world.
- **Project C – Better Lights & Materials**: in-depth GLSL: write vertex & fragment shaders for more realistic surfaces: Phong lighting model, Gouraud and Phong shading, perturbed normals.
- **3 Take-Home Section Tests (Test A, TestB, Test C):** after each final program due date, you have 3 days to complete a take-home test on the project's concepts and underlying principles.
- **Participation Points:** Simple tasks, submitted on CANVAS, to ensure you don't get too far behind.

Course Grading Method:

No midterms, no final exam

Programming Projects:	3 x 24%	(Caution! Late Penalties!)
Take-home Section Tests:	3 x 7%	
Participation Points	7 x 1%	

Class Meetings:

Section 20: MWF 10:00AM-10:50AM, Lecture Room LR-5, Technological Institute (2145 Sheridan Rd.)
Section 21: MWF 2:00PM- 2:50PM, Lecture Room LR-2, Technological Institute (2145 Sheridan Rd.)

Instructor:

Jack Tumblin, Northwestern Univ. EECS, plus a team of TAs and Peer Mentors

Location: Room 3-341(ofc) or 3-231(Lab) Ford Engineering Design Center (Just south of Tech)

Contact: For any questions on course materials or programs, just ask during class; all will benefit. I'm always available after class too; If you're not, post questions on 'Discussion Board'. For private matters? j-tumblin@northwestern.edu Office: (847) 467-2129

Office hours: by appointment – send e-mail. >160 students! Better to talk with me right after class…

TAs & Graders:

TBA

Location:

Contact:

Office hours: (3 hrs/week per TA/Grader)

Textbook (Required):

"**WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL**"
By Kouichi Matsuda and Rodger Lea. (First edition) Addison-Wesley, © 2013 Pearson Education, Inc.
(Paperback or Kindle E-book – either is OK)
Also Required: Several miscellaneous PDFs to be posted on CMS/Canvas throughout the quarter.

Other Good Resources (Recommended, NOT required)

"**Mathematics for 3D Game Programming and Computer Graphics**" **(3rd Edition or later)**
By Eric Lengyel Cengage Learning, ©2012 Course Technology
(Paperback or Kindle E-Book – either is OK). Also used for EECS 351-2 "Intermediate Graphics" course.

"**Real-Time Rendering**" (survey of techniques; little/no implementation help) by Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. 1045 pages, from A.K. Peters Ltd., ISBN 978-1-56881-424-7, 2008. Fascinating blog/updates/discussion here: http://www.realtimerendering.com/

OpenGL.org website; full of goodies and the latest news, msg boards: http://www.opengl.org/
FAQs, OpenGL online function-reference pages, GLSL function reference pages, and more:

"OpenGL Shading Language" (**3rd Edition or later:** the "Orange Book") encyclopaedic, authoritative guide to programmable shading in OpenGL and WebGL using GLSL; complete architectural guide.

"**OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3" (8th Edition or later,** the "Red Book"**)**, By Dave Shreiner, Graham Sellers, John M. Kessenich, Bill M. Licea-Kane. Addison-Wesley, © 2013 Pearson Education, Inc. (Paperback or Kindle E-Book)

Course Organization:

This course covers a large volume of fairly easy-to-grasp material, most of it posted on CANVAS, organized into 3 major sections, each with its own major project and take-home test, each with reading assignments each week, and each with a few 'participation activities' (graded only as done/not-done) to help you get started:

Project A & Section Test A-- Moving Shapes;	Projects: 3 x 24% of grade
Project B & Section Test B-- 3D Views & Shading;	Section Test: 3 x 7% of grade
Project C & Section Test C-- Better Lights &Materials.	'Participation': 7 x 1% of grade

Your assigned goal for all three graded Projects (A,B,C) is "to generate remarkable pictures on-screen" by applying the new graphical tools and methods we learned in the past 3 weeks. Each week we will talk about our projects, identify common problems and discuss good ideas that will help you solve them.

Please do the assigned reading before class, so we can discuss what you did or did not understand. While my class lecture notes should help, you will need to complete the reading and work hard on programming to do well on the tests and projects. To ensure you have done and understood the reading, you will complete a **written take-home test** for each of the 3 sections of the course. In the weeks before each project, I will ask you to post on CANVAS some **programs we began, discussed or completed in class**; each time you complete the task you **earn a 'participation' point** (up to 7 points in your final goal).

I do my best to put all useful course materials on CANVAS, but CANVAS alone is never enough. **Please note: This is *NOT* an on-line course!** Don't miss and don't skip our class meetings, or "you're gonna have a bad time"; http://knowyourmeme.com/memes/super-cool-ski-instructor, because you will miss some 'participation' activities, and may miss vital information that augments the book to help you with your project.

Reading isn't enough! To learn graphics and WebGL well, you must **write numerous graphics programs incrementally**. It's non-trivial, but the WebGL book helps: try their examples, make them work, and experiment with them to make a series of progressively improved versions of your own to learn subtleties and gain fluency. Try to build up the program towards something you find personally interesting. Create many small programs as you build your large "Project" programs in a methodical way. Test each small step and save each version, each progressively improved with bits of code from your many smaller, already-tested programs.

Project Organization:

Programming projects are the majority of your grade, not tests: I want these assignments to be challenging and to inspire your creativity, and not a desperate last-minute scramble. Please **do** start early, and please **do** explore the web and please **do** talk with other students with hints to solve problems or improve results. But please **don't** share code with other students, and **don't** plagiarize or copy.

Always refer to Canvas→Assignments for all Project information. For each Project, you will find an '**Assignment Sheet**' **(a PDF file)** with all project details, and a **'Grading Sheet' (PDF file)** that holds the Project's entire grading rubric (points earned for each implemented feature), also used for Demo Day.
We grade each of the 3 Projects (A,B,C) in two steps.

First, everyone attends a (mandatory) Demo Day, shows each other their work, exchanges advice and marks scores on each other's 'grading sheets'. As everyone learns and many realize important ways to further improve their projects, we give everyone a few days to make final revisions/improvements to their Project. Second, everyone submits their Finalized Project (by Canvas) for grading. Demo Day is often fun and inspiring, but it doesn't help anyone when a student arrives unprepared with an unfinished project, or simply does not attend the Demo Day class. These grade penalties help ensure that all students arrive well-prepared:

- To earn **100 points,** complete all Project requirements flawlessly, and attend Demo Day.
- To earn **more points**, add 'extra-credit' features to your project (described on 'Grading Sheet').

LATE PENALTIES:

- **You lose 20pts** if you do not attend Demo Day, unless you obtained a written, formal excuse in advance from Dr. Tumblin (instructor), or provide written emergency medical papers, etc.

- **You lose 10pts** if you come to Demo Day but show us a mostly-unfinished program.
  What you show the class must earn at least 50 of 100 points on the Project's 'Grading Sheet'.
- **You lose 10pts per day** if you turn in your Final Project after the deadline
  e.g. if you turn in your work 3 days late, we subtract 30 points from your score(!).

The final version of your project will fill multiple files. To turn in your work,

a) put the entire project (all files) into one carefully-named directory, and always in this format:
   FamilynamePersonalname_ProjA
   For example, my project C folder name would be: **TumblinJack_ProjC**

b) **Make sure your project file is complete.** You must include sub-directories (e.g. 'lib'), libraries (e.g. 'cuon-matrix.js') and all other files needed to run your program. HINT: move your 'project' directory to your desktop; does it still run? if not, you're missing some files!)

c) make a 'ZIP' file (or a "compressed folder" – do not use .rar or .tar compression!) of that project folder, using the same name. For example, my ZIP file name would be **TumblinJack_ProjC.zip**. Just like the 'starter code' I give you, I should be able to get your ZIP file, 'extract' its folder onto my desktop and see your results working flawlessly onscreen.

d) Include your projects' written report as a PDF file within this ZIP file (see 'Assignment sheet')
   The report filename must also match. My own report file would be: **TumblinJack_ProjC.pdf**.

e) Write and test your program to **use the Chrome Browser**. We use Chrome for grading your work.

Section Tests:

After the due date for each project (A,B,C) you will complete a 'take-home' test on its key concepts. I will post the test on CANVAS, and you will submit your answers there. The test is open-book/open-notes/open-internet, but you must complete it without any help from anyone else. The majority of each test will ask questions about material covered in class and the assigned reading. In a large class the test will consist mostly –and perhaps entirely – of multiple choice questions that require calculations and some programming.

Outside Sources & Plagiarism Rules:

Simple: *never* submit the uncredited work of others as your own.

You are welcome to begin with the book's example code and the 'starter code' I supply; you can keep or modify any of it as you wish without citing its source. I strongly encourage you to always start with a basic graphics program (hence 'starter code') that already works correctly, and incrementally improve it, testing and correcting at each step. Don't make massive untested changes all at once; you may never find all the flaws hidden in all the new code, and it's never easy to test it thoroughly. Instead, break down your big changes into small easy-to-test steps; if one small step ruins the program, search only your small set of most-recent changes and you will always find the fatal flaws.

I ***want*** you to explore -- learn from websites, tutorials and friends anywhere (e.g. StackOverflow, MDN, CodeAcademy, OpenGL.org, etc), and to apply what you learn in your Projects. Share what you find with other students, too -- list the URLs on CMS/Canvas discussion board, *etc.* and list in the comments the sources that helped you write your code. ALWAYS credit the works of others—**no plagiarism!**

Plagiarism rules for writing essays apply equally well to writing software. You would never cut-and-paste paragraphs or whole sentences written by others unless it's a clearly marked quote with cited sources. The same is true for whole functions, blocks and statements of code and scripts written by others. Never cut-and-paste others' code without crediting them, and never try to disguise it by rearrangement and renaming (TurnItIn won't be fooled). Instead, write your own code to learn the principles well and to earn your own grades: study their good code carefully, learn its best ideas, habits and methods, and then close the book or website; apply what you learned, and write your own code. Take their good ideas, but not their code: add a

gracious comment that recommends the inspiring source of those good ideas, and then write your own, better code in your own better style; stay compact, yet complete, create an easy-to-read, easy-to-understand style.

Also, please note that I apply the 'TurnItIn' anti-plagiarism software on all graded assignments: https://canvas.northwestern.edu/courses/1580/pages/turnitin-in-canvas

(If I find any plagiarism evidence (sigh), the University requires me to report it to the Dean of Students for investigation. It's a defeat for all involved: when they find misconduct they're very strict and very punitive).

SCHEDULE:

Introduction:

Week 01: Jan. 04, 06, **08** (**Fri Jan 08: last day to change course registrations** for Winter Quarter)

Reading: WebGL Guide: Chap 1,2

Topics: Administrivia. Chrome JavaScript console + basics; how WebGL fits into HTML-5 & Javascript. Geometry: Canonical View Volume (CVV). Math review: points, vectors, coord systems, dot, cross, matrix. Vertex/Vector/Matrix Math; homogeneous coords (x,y,z,w). cuon-matrix library.

PROJECT A: Moving Shapes

Assemble sets of 3D points, lines, and polygons into faceted shapes; apply 3D transforms to make colorful 3D moving objects that respond to mouse and keyboard inputs.

Week 02: Jan. 11, 13, 15

Reading: WebGL Guide: Chap 3,4;

Topics: Vertex Buffer Objects (VBOs); GLSL & Shader basics; how to set 'attributes' and 'uniforms' from JavaScript. Matrix duality: Scene Graphs, trees of transformations for jointed objects.

Week 03: Jan. ~~**18**~~, 20, 22 **(Mon Jan 18: Martin Luther King Celebrations)**,

Reading: WebGL Guide: Chap 5 (stop at pg. 160, "pasting an image onto a rectangle").

Topics: User controls, mesh organizing methods (winged-edge, half-edge, loops, hairs, cuts, shells. Vertex colors & fragment shading. Extended user interface methods.

Week 04: Jan. **25**, 27, 29 (**Mon Jan 25: Demo Day, 24pts**), (**Thu Jan 28: take-home test, 7pts**)

Reading: Quaternions: excerpt from "3D Mathematics…" Lengyel.

Topics: 2D and 3D rotational difficulties—how can we compute a 'trackball' user interface?

PROJECT B: 3D Views & Shading

Practical animated 3D cameras & basic Lighting; multiple re-sizeable windows.

Week 05: Feb. 01, 03, 05

Reading: WebGL Guide: Chap 6. Also review WebGL 1.0 specification (Khronos website).

Topics: Shaders explained: what GLSL can (and can't) do for you. Simple diffuse overhead light in GLSL. Intuition for the do-it-yourself 'view' matrix; we just "Push the World Out of Your Eyes".

Week 06: Feb. 08, 10, **12** (**Fri. Feb. 12: Drop Day**)

Reading: WebGL Guide: Chap 7:

Topics: How to build your own 'look-at' matrix: the clever orthonormal trick: $A^{-1}==A^{T}$. Camera geometry→camera matrices: Orthographic, skewed, and projective: glFrustum, pseudo-depth and why this approx. works. How cameras and views expand scene graphs and the tree of transformations.

Week 07: Feb.**15**, 17, 19 (**Mon Feb 15: Demo Day: 24pts**), (**Thu Feb 18: take-home test: 7pts**)

Reading: WebGL Guide: Chap 8 through page 310. (Spring Qtr. Course-Planning Week)

Topics: Graceful camera navigation: the 'point-on-cylinder' steering method. Multiple 'viewports' & cameras in WebGL fitted to re-sizeable canvas. How to mount cameras on jointed, moving objects. Lighting basics: How to transform and position lights in your Scene Graph.

PROJECT C: Better Lights & Materials

The Phong lighting model and beyond: textures, buffer tricks, and advanced WebGL shader writing.

Week 08: Feb. 22, 24, 26 (Spring Qtr. Registration Week)

Reading: WebGL Guide: Chap 8 pg. 310 to Chap 8 end. Skim/Review Chap 9 (scene graphs)

Topics: Phong lighting model (ambient, diffuse, specular, emissive). How to construct 'Surface normal vector' attributes for each vertex in your WebGL buffers.

Week 09: Feb. 29,Mar. 02, 04

Reading: WebGL Guide: Chap 5 pg 160-189; Skim/Review Chap 10 through page 385 (misc I/O)

Reading: Lengyel Excerpt: Lighting and Shading (Chap 7).

Topics: Texture map basics: diffuse-only; as part of Phong lighting; 'Bump maps'.

Week 10: Mar. 07, 09, 11 (**Mon Mar 07: Demo Day: 24pts**) (**Thu Mar 10: take-home test: 7pts**)

Reading: WebGL Guide: Chap 10 page 386-end. (Mar 08-11: WCAS Reading Week)

Topics: Shader-switching; render-to-texture methods; framebuffer access; basic shadow methods. (Help Session Friday Mar 11).

(Mar 14-18: Final Exam Week)

Further topics you can pursue on your own:

Advanced shader methods: reflection mapping; render-to-buffer tricks: mirrors. Why WebGL still doesn't quite match a photograph; advanced shaders. Radiometry Basics: BRDF and light transport.