EECS 351-1:Introduction to Computer Graphics
Syllabus: Winter 2017 (Jan-Mar)

Updated: 1/8/2018 4:15 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 Shape, View & Light 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 won't study big content-creation packages (Blender, Maya, Renderman, SketchUp, Unity, or even "three.js") or how to use them. Instead, you will learn *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) OpenGL '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 switch to machine-native OpenGL, GLSL or DirectX, HLSL easily), and will understand the computations underlying all 3D graphics packages, visual special effects, and current game engines.

Topics:

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 cameras, to movable 3D lights, to quaternion rotations for objects, and to build a diffuse- lit, animated 3D world.
- **Project C – Better Lights & Materials**: in-depth GLSL: vertex & fragment shaders for realistic surfaces: Gouraud and Phong lighting & shading model, textures, perturbed normals, etc.
- **3 Take-Home Section Tests (Test A, Test B, 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 L-361, Technological Institute (2145 Sheridan Rd.)

Instructor: Jack Tumblin, Northwestern Univ. EECS, plus a few Peer Mentors

Location: Room 3-205 (office) 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. >70 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 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 make original moving pictures on-screen" by applying the new graphical tools and methods we learned in the past 3 weeks.

- Try to do the assigned reading early. Review the lecture-notes posted, too! The written take-home tests rely heavily on the reading assignments, lecture notes, and even the starter code.
- **Combine reading with programming.** To learn graphics principles and WebGL well, I encourage you to **write plenty of small graphics programs** incrementally to test what you learned from the reading.
- **Build projects incrementally from starter code.** Make a small improvement, make sure it works, save all with the next higher-numbered version, building a long series of better programs that work. Expect surprises – WebGL and GLSL development & debugging can seem impossible if not done incrementally.
- **This is *NOT* an on-line course!** I will post everything I can on CANVAS, but please don't miss our class meetings, as then you won't miss any in-class 'participation' activities (and their 7 points), and you'll get answers to questions, discussions and extra help for tricky parts of the programming projects.

Project Organization:

Canvas→Assignments supplies ***all*** the project information you will need. For each Project, you will find an

- '**Assignment Sheet' (a PDF file)** with all project details and expectations, and a
- **'Grading Sheet' (a PDF file)** that holds the Project's entire and exact grading rubric (lists points earned for each implemented feature), also used for Demo Day.

We grade each of the three Projects (A, B, C) in two steps.
First, everyone attends a (mandatory) Demo Day, demonstrates their program for everyone, exchanges advice and marks scores on each other's 'grading sheets'. Demo Day is often fun and inspiring, as everyone learns and many realize important ways to further improve their projects. Class attendance is *vital* on Demo Day! We then give everyone a few days after Demo Day to make final revisions/improvements to their Project.
Second, everyone submits their Finalized Project (by Canvas) for grading.

- To earn **100 points**, complete all Project requirements flawlessly, and attend Demo Day well-prepared.
- 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 emergency-room 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(!).

Programming Projects are the majority of your grade and should take the majority of your attention, so it's never a good idea to do them in a desperate last-minute scramble. Give yourself time to explore and experiment to make something visually interesting. Please **do** start early, and please **do** explore the web and please **do** share hints and ideas on Discussion Board to solve problems and improve results. But, of course,

don't share code with other students, and **don't** plagiarize or copy anyone else's code. Let's not repeat last year's plagiarism tragedies: if you can find others' old code on GitHub, then so can I, and 'MOSS' will flag it.

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: put 'project' directory on the 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. **(!! I do not accept .rar files !!)**

d) Include your projects' written report as a PDF file within this ZIP file (see 'Assignment Sheet')
   Your 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 use any of the 'starter code' that I supply you on CANVAS; you can keep or modify any of it as you wish without citing its source.

I ***want*** you to explore -- learn from websites, tutorials and friends anywhere (e.g. StackOverflow, MDN, CodeAcademy, OpenGL.org, even GitHub (but DON'T COPY!)), to help make great Projects. Share what you find with other students, too -- list the URLs on CMS/Canvas discussion board, *etc.* as comments to give 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 copy others' code without crediting them, and never try to disguise it by rearrangement and renaming (MOSS won't be fooled). Instead, read their code, learn its best ideas and methods, then *delete the file, close the webpage*.
Copy nothing -- write your own, better code in your own better style, in your own words.
Then add a gracious comment to your code that credits the source of those good ideas, too.

Also, please note that I apply the 'MOSS' anti-plagiarism software on all graded assignments:
   https://theory.stanford.edu/~aiken/moss/

(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 the investigation concludes there was misconduct, then they're very strict and very punitive). Several students were badly shamed by this last year. !Please! No more!

EECS 351-1 SCHEDULE:

Introduction:

Week 01: Jan. 08, 10, **12** (**Fri Jan 12: 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. Canonical View Volume (CVV). Commonplace user interface in HTML/ JavaScript: tags, console, innerHTML; how to make callbacks for keyboard and mouse in HTML-5 canvas and browser window.

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. ~~15~~, 17, 19 **(Mon Jan 15: Martin Luther King Celebrations)**

Reading: WebGL Guide: Chap 3,4;

Topics: Math review: points, vectors, coord systems, dot, cross, matrix math, homogeneous coords (x,y,z,w). cuon-matrix library. Matrix duality: Scene Graphs, trees of transformations for jointed objects. Vertex Buffer Objects (VBOs); GLSL & Shader basics; how to set ‘attributes’ and ‘uniforms’ from JavaScript.

Week 03: Jan. 22, 24, 26,

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

Topics: Vertex colors & fragment shading. User controls, some basic triangle-strip objects, mesh organizing methods (winged-edge, half-edge, loops, hairs, cuts, shells. Extended user interface methods: trackball, pick methods.

Week 04: Jan. **29**, 30, Feb 02 (**Mon Jan 29: Demo Day, 24pts**), (**Thu Feb 01: 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? Shaders explained: what GLSL can (and can’t) do for you. Approximating overhead light in GLSL.

PROJECT B: 3D Views & Shading

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

Week 05: Feb. 05, 07, 09

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

Topics: Camera geometry→camera matrices: Orthographic, skewed, and projective: glFrustum, pseudo-depth and why this approx. works. Multiple ‘viewports’ & cameras in WebGL fitted to re-sizeable canvas.

Week 06: Feb. 12, 14, **16** (**Fri. Feb. 16: Drop Day**)

Reading: WebGL Guide: Chap 7.

Topics: How cameras and views expand SceneGraphs and the tree of transformations. Do-it-yourself ‘view’ matrix; we just “Push the World Out of Your Eyes”. How to build the ‘look-at’ matrix.

Week 07: Feb. **19**, 21, 23 (**Mon Feb 19: Demo Day: 24pts**), (**Thu Feb 22: 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. How to mount cameras on jointed, moving objects: SceneGraph result. Lighting basics: N·L*color; Surface Normal attribute.

PROJECT C: Better Lights & Materials

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

Week 08: Feb. 26, 28, Mar 02 (Spring Qtr. Registration Week)
Reading: WebGL Guide: Chap 8 pg. 310 to Chap 8 end. Skim/Review Chap 9 (scene graphs)
Topics: Computing surface normals for each vertex in VBO. Gouraud vs Phong shading: Phong vs. Blinn Phong lighting model (ambient, diffuse, specular, emissive); shader-writing, object-oriented organizing.

Week 09: Mar. 05, 07, 09
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. **12**, 14, 16 (**Mon Mar 12: Demo Day: 24pts**) (**Thu Mar 15: take-home test: 7pts**)
Reading: WebGL Guide: Chap 10 page 386-end.   (Mar 13-18: WCAS Reading Week)
Topics: Shader-switching; render-to-texture methods; framebuffer access; basic shadow methods.
(Help Session Friday Mar 16).
(Mar 19-23: Final Exam Week)