Syllabus – CS 371 – Computer Graphics – Fall 2019

1:50 - 3:20, MW

COURSE DESCRIPTION: An introduction to the mathematics, data structures, and algorithms used to create graphical output in the programmable pipeline. Topics include graphics hardware, shaders, transformations in two and three dimensions, three-dimensional viewing, modeling three-dimensional shapes with polygon meshes, hierarchical modeling of three-dimensional objects, lighting and shading techniques, raster algorithms. Prerequisite: Computer Science 262 and Mathematics 171 or Math 206 each with a grade of C or better.

INSTRUCTOR: Tom Naps

EMAIL: naps@uwosh.edu or thomas.naps@gmail.com

OFFICE HOURS: MW 12:30 - 1:30 (Halsey 214) and as needed in Halsey 266 after class ends

REFERENCES:

- Daily class handouts available by 4:00am MW on D2L. They comprise a high-level, but not detailed, outline of what we will cover during class. Get a copy of them before class, organize them, take notes on and about them. Handouts that are not liberally saturated with your own explanatory notes will likely prove useless when you need them most
- Lots of demonstration programs that will allow you to do "what-if" experimentation (I use these myself to make up devilish test questions)
- Review problems posted on D2L at the end of each class. Complete them before the start of following class meeting and be ready to discuss the answers you have submitted.
- An optional but highly recommended reference: *Interactive computer graphics : a top-down approach with WebGL*, 7th edition, by Edward Angel and Dave Shreiner. Much of what we do will be based upon this book. However, mathematically we will be less rigorous than the approach taken in the book.
- Materials from an online course offered by Edward Angel that is based on the above book.
- Portions of a free online graphics book (`http://math.hws.edu/graphicsbook`) authored by David Eck of Hobart and William Smith College. In particular, the most relevant sections are:
  - Chapter 6 – Introd. to WebGL
  - Chapter 7 – 3D Graphics with WebGL
  - Appendix A, Section 3 – JavaScript

Topic Coverage

1. Graphics hardware, specifically the somewhat obsolete fixed function pipeline (OpenGL) and the programmable pipeline (WebGL and GLSL) that has replaced it.
2. Two-dimensional graphics
3. Math necessities beyond the course prerequisites
4. Transformations in two- and three-dimensions
5. 3-D Viewing with the synthetic camera
6. Modeling 3-D shapes with polygon meshes
   - Meshes obtained from various data collections
   - Meshes obtained from "pure" mathematical surfaces, that is, surfaces with coordinates defined strictly from (parametric) functions.
   - Approximating/interpolating curves and surfaces
7. Hierarchical modeling of 3-D objects
8. Lighting and Shading – the algorithmic rendering ladder
   - The Phong reflection model and its variations, the Lambertian and Phong-Blinn models
   - Gouraud shading
   - Phong shading
   - Texture mapping
   - Bump mapping
   - Reflection mapping
   - Ray tracing
   - Photon mapping
   - Radiosity
9. Raster Algorithms – as time allows

Learning Outcomes

Given our coverage of these topics, you will be expected to . . .

1. Identify and define the purpose of each component in the graphics pipeline that transforms a vertex in world coordinates to a pixel location with a particular color
2. To perform in manual fashion the transformation carried out by the graphics pipeline on points in two-dimensional and three-dimensional world coordinate space

3. Discuss the relationship between the aspect ratio of a scene and the viewport in which it is rendered
4. Define and discuss the role of double buffering in real-time animations
5. Apply linear affine transformations such as scaling, translation, and rotation to points in two- and three-dimensional space and analyze the effects of such transformations on the points in a rendered scene
6. Define and compare the perspective and orthographic projections on points and scenes in three-dimensional space
7. Plan and design scenes animated by an underlying hierarchical model
8. Identify the role of the model-view transformation and its matrix representation in rendering hierarchical models
9. Define the roles of the eye point, look point, and up vector parameters in the synthetic camera's view of a three-dimensional scene and to perform the computations necessary to illustrate how these parameters affect the model-view transformation matrix
10. Trace the depth-buffer (Z-buffer) algorithm as it is used to determine hidden points and surfaces in a rendered scene
11. Define and compare the variety of transformations used in texture, bump, and reflection mapping that associate a coordinate on a model with a color or normal vector determined by the corresponding map.
12. To discuss the mathematics underlying two- and three-dimensional interpolating curves and surfaces (for example, Bezier curves and surfaces)
13. Discuss the roles played by color, lighting, and material parameters in the progression of increasingly sophisticated shading models such as flat, smooth, Gouraud, Phong, ray-tracing, radiosity, and photon-mapping
14. Using visual clues, differentiate between scenes rendered by a variety of shading models such as flat, smooth, Gouraud, Phong, ray-tracing, radiosity, and photon-mapping
15. Analyze the relationship between computational rendering algorithms for increasingly sophisticated shading models - flat, smooth, Gouraud, Phong, ray-tracing, radiosity, and photon-mapping - and the time required to render the scene using that algorithm
16. Using a graphics library such as WebGL in conjunction with the GLSL shading language, implement three-dimensional animations rendered in real-time using an appropriate lighting model built into the programmable pipeline.

Course Grading Policies

Your grade for the course will be based on the following weighted factors:

Factor:	**Weight:**
6-8 Assignments	45% in total
Daily Class Participation	5%
Review problems	5%
Three Exams:	45%

To get the 45% contribution to your grade from the three exams, I will use the formula:

$$E = \frac{2}{9} \times E_{worst} + \frac{4}{9} \times E_{best} + \frac{1}{3} \times E_{other}$$

where E_{worst} is your worst exam score and E_{best} is your best exam score. Essentially this lessens the effect of a bad exam score by only counting that exam half of your best exam score.

Meeting the specifications on a programming assignment will only guarantee a 90% grade. To get 100% (or qualify for golly-gee-whiz points above 100%), you must provide add-on features that are clearly explained and enumerated in the opening documentation block that accompanies your program.

At the end of the term, your work in all of these areas will contribute to a numerical grade for the course based on a 100-point scale. Grade cutoff levels on this final scale are:

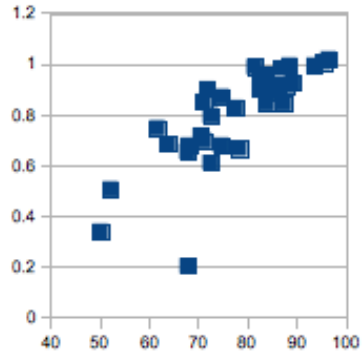
A $\geq$ 92	B $\geq$ 82	C $\geq$ 72	D $\geq$ 62
A- $\geq$ 90	B- $\geq$ 80	C- $\geq$ 70	D- $\geq$ 60
B+ $\geq$ 88	C+ $\geq$ 78	D+ $\geq$ 68	F $<$ 60

FAQ

Do I have to come to class? You are expected to arrive prepared to ALL the course sessions. Furthermore you are expected to participate in the classroom discussions and activities to the best of your abilities. This includes being ready to defend your answer to the review problems from the previous class (more on that later). It is difficult to envision a student missing and/or arriving unprepared to a number of the class sessions and still succeeding in the course.

What if I'm late in completing and submitting a programming assignment? It will be accepted but will be penalized at the rate of 10% of point value the first day late, *an additional* 20% the second, *an additional* 30% the third ... Additionally late work will not qualify for any golly-gee-whiz points.

What if I'm late in completing the daily review exercises? You will get zero credit for that set of review exercises. Although the review problems only count 5% of your grade, the following correlation from a previous course between review-problem-percentage (on a 0 to 1.0 vertical scale) and overall percent in the course (on a horizontal scale of 0 to 100) is indicative of their true importance.



We will always discuss the review problems assigned on a given day at the beginning of the next class. If you have participated in class the day the review problem was distributed, have made a good faith effort to work on the review problem, and are "stuck" on it, I will be more than happy to help you with it if you check with me after class or come to my office hours before the next class. If you wait longer than that to see me for help, you are on your own in terms of grappling with these review problems *because you have made the choice to not learn efficiently.* In particular, I will not entertain questions about past review problems as an exam approaches.

If I miss an exam, can I make it up? If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do BOTH of the following, which are then subject to my approval:

- Make arrangements prior to the scheduled exam (for last minute emergencies, telephone me at 424-1388 or leave a message at the computer science office, 424-2068). No after-the-fact notifications will be accepted . . . *AND*
- Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

Only one make-up exam will be given. It will be a rigorous comprehensive exam given at an arranged time during the last week of the semester.

Can I get a no-penalty extension on work that is due on a specified date? Only if you're ill enough to provide a signed note from the attending physician or have other reasons serious enough that the Dean of Students Office is willing to provide a written note justifying the extension.

Can I work with others on the homework assignments or the project? You have the choice to work on your own or with *one* partner, in effect working in *pair learning/programming* style. (See research by A. Cockburn and L. Williams indicating that, when done conscientiously, working in paired environments will slightly increase the time you put in and significantly improve the quality of the work.) You are not allowed to collaborate with anyone else for assignments. If you decide to work with a partner, here is the way that it must be done:

- *Both* you and your partner must "declare" via emails sent to me *three days* before an assignment is due that you will be working together. Once that declaration is made, it will remain in effect until you email me again that you no longer wish to work as partners.
- If you decide to work with a partner, only one of you will submit the assignment, and that is what I will grade. You will both receive the same grade.
- You may not share or even discuss your work with anybody but your partner unless you can live with a zero and the other potential academic sanctions of cheating (see the UWO Student Discipline Code, Chapter UWS 14).

If you decide to work on your own, then the third bullet point above also applies (minus the partner).

Required disclosure statement from financial aid office Students are advised to see the following URL for disclosures about essential consumer protection items required by the Students Right to Know Act of 1990: `https://uwosh.edu/financialaid/consumer-information/`.