

INSTRUCTOR: Tom Naps

OFFICE: Halsey 229, phone 424-1388

EMAIL: naps@uwosh.edu

OFFICE HOURS: MTuWThF, 11:30-12:30.

TESTS: • Test 1: Monday, September 29

- Test 2: Monday, October 27
- Final Exam: Wednesday, December 10 & Friday, December 12

REFERENCES: • Daily class handouts - 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

- Optional but good references: (1) *OpenGL Programming Guide*, a.k.a. the “Red Book” by Woo, Neider, David, and Schreiner, (2) *Interactive computer graphics : a top-down approach with OpenGL* by Edward Angel, and (3) *Computer Graphics Using OpenGL* by F.S. Hill and S.K. Kelley
- Lots of programs that will allow you to do ”what-if” experimentation (I use these myself to make up devilish test questions)
- Course Web page at <http://csf11.acs.uwosh.edu/moodle>. The first time you login to this page, use the enrollment key *cs371* to enroll in this course.

Topic Coverage

1. Graphics systems in general – the pipeline and the “algorithmic rendering ladder”
2. Two-dimensional graphics
3. Math necessities beyond the course prerequisites
4. Transformations in two- and three-dimensions
5. Modeling 3-D shapes with polygon meshes
6. 3-D Viewing with the synthetic camera
7. Lighting and Shading
8. Adding realism with ray tracing, photon mapping, and radiosity
9. Approximating/interpolating curves and surfaces
10. Raster Algorithms – What’s Below OpenGL? – discussions of these topics scattered throughout the course

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
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. Trace the Cohen-Sutherland two-dimensional clipping algorithm on points in world-coordinate space
5. Trace Bresenham’s line-drawing algorithm in two-dimensional space
6. Define and discuss the role of double buffering in real-time animations
7. 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
8. Define and perform the perspective and orthographic projections on points and scenes in three-dimensional space
9. Compare scenes rendered by perspective and orthographic projections
10. Plan and design scenes animated by an underlying hierarchical model
11. Identify the role of the model-view transformation and its matrix representation in rendering hierarchical models
12. 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
13. Trace the depth-buffer (Z-buffer) algorithm as it is used to determine hidden points and surfaces in a rendered scene
14. Define and compare the variety of transformations used in texture mapping to identify a point in texture space with a point in world coordinate space
15. To discuss the mathematics underlying two- and three-dimensional interpolating curves and surfaces (for example, Bezier curves and surfaces)
16. Discuss the roles played by color, lighting, and material parameters in the progression of increasingly sophisticated shading models - flat, smooth, Gouraud, Phong, ray-tracing, radiosity, and photon-mapping
17. 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
18. 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

19. Using a graphics library such as OpenGL, implement three-dimensional animations rendered in real-time using an appropriate lighting model
20. Using an appropriate scene description language such as POVray, implement scenes that employ lighting and shading algorithms that cannot be rendered in real-time

Course Grading Policies

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

Factor	Weight
6-8 Assignments	45% in total
Class participation and preparation	10%
3 exams:	
Exam 1	12.5%
Exam 2	12.5%
Exam 3	20%

Unlike other courses I teach, in this course meeting the specifications on a programming assignment will only guarantee a 90% grade. To get 100% (or qualify for golly-gee-whiz points), 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:

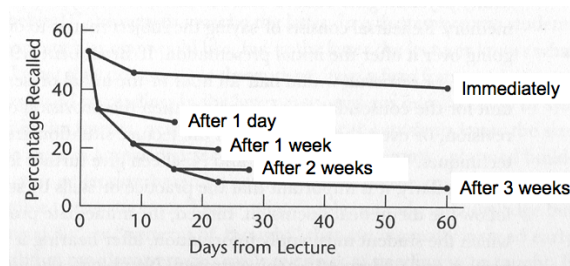
A ≥ 92	B ≥ 82	C ≥ 72	D ≥ 60
AB ≥ 89	BC ≥ 79	CD ≥ 69	F < 60

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. 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.

How much time will this course take? Figure about three hours outside of class for each hour in class. That heuristic makes being a full-time student pretty much equivalent to holding a full-time job, so this is really good preparation for the real world that awaits you after graduation.

How can I best prepare for the exams? We’ve known what the following graph illustrates since 1968:



Bassey M. (1968), in *Learning methods in tertiary education*

Consequently at the end of most of our class periods I will give you one or more exam review problems. The time to work on these review problems is immediately after the material is covered in 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 come my office anytime within three days after you have received the review problem in class.

Stuckness shouldn’t be avoided. It’s the psychic predecessor of all real understanding. (Robert Pirsig – Zen and the Art of Motorcycle Maintenance)

After those three days (not counting weekends), *because you have made the choice to not learn effectively*, you are on your own in terms of grappling with these review problems.

What do I have to hand in for these assignments and what if it’s late? Handing in an assignment requires that you submit:

1. An electronic copy of source files that contain your programming work via your “drop box” for this course on the campus Linux network.
2. A URL that I can use to run your program via “Webstart”

Each assignment will carry with it a due date. The electronic copy of your programming work and the Webstart URL are due "at the bewitching hour" on the due date. The program that is Web-started from the URL must be the program that is produced when I compile your source code. Late programming assignments 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 ...

Is there any way I can carelessly lose points in the course? Yes ...

- Be late in handing in your work on assignments.
- Don't participate in and prepare for the class.

What is this class participation/preparation stuff? How does it add up to 10% of my grade? ...

- Be sure to do those review problems before the next class meeting. If you do that and get them right, you get full credit for them. If you do that but get them wrong, you get half credit. If you don't do them, you get no credit.
- Exhibit your knowledge when called on to explain your correct answer to a review problem.
- Be sure to "vote" on the aesthetics of the programs that others submit.

Is there any way I can get some bonus points? ...

- You can get "golly-gee-whiz" points by being creative and going beyond the plain vanilla specifications for your programming assignments. This will enable you to fare well in the vote for the "best program" on each assignment.
- Do an outstanding job when called on to explain your correct answer to a review problem.

Can I get an 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.

If I miss a test, 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 work with others on assignments? No, not in the sense of two people working on the same program. However, it is acceptable to consult another student for help in debugging a program that you have authored yourself and that is not producing the result your expected. It is also acceptable to cut-and-paste code snippets from in-class demos or examples you find on the internet *provided* that you cite the sources of these code snippets in the introductory documentation block at the beginning of your program.

Can I do programming work on my own computer instead of using the Linux systems in the Halsey lab? Sure, you can download Java, Apache ant, and JOGL ("Java OpenGL") for your Windows, Linux, or Mac system. Links to appropriate download sites are available on the course Web page.