

22C:251 Advanced Computer Graphics (Realistic Image Synthesis)

Syllabus for Spring 2007

Class Time: TTH 9:30–10:45 am, 110 MacLean Hall.

Class Webpage: <http://www.cs.uiowa.edu/~cwyman/classes/spring07-22C251/>

Instructor: Chris Wyman

E-mail: cwyman@cs.uiowa.edu (*Preferred contact method*)

Office: 101J MacLean Hall

Office Phone: (319) 353-2549 (*Please e-mail if possible*)

Office Hours: *To be determined...*

Prerequisite: Grade of C- or better in 22C:251. A working knowledge of calculus and linear algebra. Proficiency in C/C++, or the ability to learn fast.

Course Objective:

This course builds off 22C:151 by investigating a specific area of computer graphics, that of realistic image synthesis. Realistic image synthesis includes many topics related to the creation of photorealistic computer graphics, including physics, statistics, material science, data structures, software engineering, and parallel programming.

This semester, we will focus on ray-based techniques for image synthesis, including ray and path tracing. The objectives of this course include: a better understanding of the physics of light, numerical sampling, material reflectance, efficient programming techniques, and the creation of a solid ray tracing infrastructure suitable for use as a research platform.

This course will require significant hands-on programming and self-motivation. Furthermore, it is highly recommended that students think about the expandability of their software design before coding. All assignments and projects will use *the same code base*, so bad software design early will significantly increase coding time later in the semester.

Course Requirements:

Course grades will be based on assignments and one final semester-long project, as follows:

10 % – *Class Participation*

40 % – *Homework*

50 % – *Final Project*

Because this will be a software engineering course, interactive ray tracing course, and realistic image synthesis course wrapped up in one, we will be covering a lot of material in class. Class attendance is thus highly encouraged to avoid missing vital information (which may only be covered once). The *Class Participation* portion of your grade will be based on class attendance, participation in class discussion, and occasional mini-assignments to get hands on experience with simple concepts in your ray tracer.

Homeworks and your final project may be performed either individually or in groups of two. Groups will be expected to do more work than individuals (though not necessarily twice as much) and document contributions from each individual.

To get full credit on assignments and your final project, I expect:

- Your code will be readable. This means I want lots of comments (see my ray tracing framework for an example), good code indentation, and a consistent coding style. This is typically not something you can do quickly at the last moment, so please get in the habit of doing this from the onset.
- You will provide me a link to a web page exhibiting examples from your assignments as well as final project updates.

- Your code will compile on a Linux machine (either the lab machines in 301 MLH or my 8-core ray tracing machine `dpmlh045.divms.uiowa.edu`, which you may ask me about to get access). This means if you develop under Windows, you must maintain *both* the VS.NET project file and the Makefile that come with the ray tracing framework.

Grades will be based on a plus/minus system, where pluses and minuses will only be given in borderline cases.

Final Project:

The final project will consist of writing a ray tracer with one of two goals (student/group's choice):

- To create a photorealistic renderer that generates images comparable to a photo. (*This will require photographing a simple scene and modelling it in your software. Each student in a group must model a different scene.*) If you choose to do this project, start thinking about where you can get access to simple objects for your scene, a camera, a tripod, a scanner (if needed), and a simple lighting environment in which to photograph your objects.
- To create a ray tracer designed for speed. This ray tracer will need to render even relatively complex scenes at multiple frames per second on a single CPU. However, you must also parallelize your software to run simultaneously on multiple processors for a further speedup.

Groups may choose to do “both” projects, with one member focusing on each of the two directions (realism and speed), but sharing a common code base. Please discuss this with me briefly if you like this idea.

Textbooks:

- **Required:** *Advanced Global Illumination (2nd Edition)*, P. Dutre, K. Bala, and P. Bekaert. AK Peters, 2006. ISBN: 1-56881-307-4.
- **Required:** *Realistic Ray Tracing (2nd Edition)*, P. Shirley and R. Morley. AK Peters, 2003. ISBN: 1-56881-198-5.
- **Alternate:** *Physically Based Rendering*, M. Pharr and G. Humphreys. Morgan Kaufmann, 2004. ISBN: 0-12-553180-X.

The idea behind the textbook choices is to give you information on both the implementation of a ray tracer as well as information about advanced global illumination techniques. Either the combination of the two required texts or the single alternative text will give you both. In the case of the required texts, the *Advanced Global Illumination* book will be most useful for the course, and if you feel confident you understand ray tracing, *Realistic Ray Tracing* may be unnecessary.

You may, alternatively, replace the pair of books with the single text *Physically Based Rendering*. I feel, however, that *Advanced Global Illumination* serves as a better reference for advanced rendering techniques, and that *Realistic Ray Tracing* describes a simple, well-designed ray tracing infrastructure that you can mold to your own programming style. Using the code from the *Physically Based Rendering* text requires a significantly larger time investment in reading code and understanding their programming style and design decisions, and the text focuses heavily on documenting the code, so finding discussion of the rendering theory can sometimes be difficult (however, the text is excellent).

Late Assignments:

In order to pass this course, all assignments *must* be turned in. However, late assignments will receive a grade of *zero* unless the situation is discussed with me before the due date.

Academic Honesty:

As this course is an advanced course aiming to prepare students for research, the use of other people's work is allowable in a number of situations. However, I still expect students to maintain academic honesty.

The use of non-original code is allowable if:

1. It does not defeat the purpose of the assignment (to learn a particular topic).
2. The student discusses it with the instructor in advance.
3. All non-original code is clearly cited and available to the instructor in its original form.
4. The student understands the code and can explain the functionality used to the instructor. (This means you can describe the algorithms used, parameter values, function calls, etc.)
5. It expands the functionality beyond the scope of the assignment/project.

For clarification on what constitutes academic dishonesty, contact me or consult the printed policy in the *Schedule of Courses*, the *CLAS Bulliten*, or online at http://www.clas.uiowa.edu/faculty/teaching/classroom_p&p/acad_fraud_etc.shtml. Clarification must occur *before* you turn in questionable work.

Further Considerations:

I need to hear from any student with a disability that requires modification to seating or other class requirements. Please talk with me as soon as possible during office hours, so that appropriate arrangements can be made in a timely fashion. For more information on the procedures refer to: http://www.clas.uiowa.edu/faculty/teaching/classroom_p&p/disabilities.shtml

Note: As a course offered by the College of Liberal Arts and Sciences, course policies are governed by the CLAS.

Complaints:

If you have complaints, please feel free to discuss them directly with me during office hours or via e-mail. If you have problems with the TA, please attempt to resolve them with her first before contacting me. If you do not feel I have appropriately dealt with your complaint, you should consult the Computer Science DEO/Chair, Professor Jim Cremer, 14D MacLean Hall, (319) 335-1713, cremer@cs.uiowa.edu. If still unresolved, complaints must be submitted in writing to (for undergrads) Helena Dettmer, the CLAS Associate Dean for Academic Programs, or (for grads) to Eric Wurster, Graduate College Associate Dean for Academic Affairs. Further information about this policy is available at: http://www.clas.uiowa.edu/students/academic_handbook/ix.shtml#4.