Description
An introduction to the fundamental principles of computer graphics using OpenGL. Among the topics covered are: graphics hardware, 2-D graphics (including line and circle drawing, transformations, windows, viewports, and clipping), 3-D perspective graphics, back-face removal, one or more hidden-surface algorithms, and simple light models.

Prerequisites
Completion of CSCI 230 (Data Structures and Algorithms), MATH 220 (Calculus II), and MATH 307 (Discrete Structures II).

Instructor
Anthony Leclerc, Ph.D.
J. C. Long Building Room 208
Office Phone: 953-5963
E-mail: leclerc@cs.cofc.edu

Office Hours
10:00 a.m. – 12:00 p.m. MWF
2:00 p.m. – 3:15 p.m. R
other times by appointment

Classroom
LONG 221

Required Texts

Grading Procedure and Scale

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90-100</td>
<td>A</td>
</tr>
<tr>
<td>87-89</td>
<td>B+</td>
</tr>
<tr>
<td>80-86</td>
<td>B</td>
</tr>
<tr>
<td>77-79</td>
<td>C+</td>
</tr>
<tr>
<td>70-76</td>
<td>C</td>
</tr>
<tr>
<td>65-69</td>
<td>D</td>
</tr>
<tr>
<td>0-64</td>
<td>F</td>
</tr>
</tbody>
</table>
Course Policies

- **Noisy portable electronic device policy:** Before entering class, turn off all cellphones, pagers, and other electronic devices that may interrupt the class with noise.

- **Attendance policy:** You are expected to attend every class. Attendance is critical to your success in this course. While most information will be available either on-line or in the textbook, some information may only be presented during class discussion. You are responsible for all information and announcements (including test times, assignment due dates, etc.) given in class.

- **Submission policy:** An assignment must be submitted electronically using the submit executable program available on stono and its clients.

- **Lateness policy:** An assignment must be submitted at or prior to the beginning of class on the due date scheduled by the instructor. The submit executable will reject any submissions made after the due date and time.

- **No compilation policy:** No credit will be give for a programming assignment which does not compile. It is your responsibility to verify that your program compiles on stono. If you develop your program on another system, port it to stono and re-compile (and run) it there (before you submit it) to verify that everything works.

- **Makeup policy:** No makeup tests will be given. If a student presents a written excuse from the Undergraduate Dean’s office for a missed test, then the following test (possibly the final exam) score will count additionally for this missed exam. A score of zero will be recorded for any other missed test.

- **Plagiarism policy:** On all assignments, you are expected to do your own work! It is easy to “surf” the web (or some repository of past 480 solutions) and find a solution (or “near” solution), modify it, and then submit it as your own work. Or, easier still, a student may choose to let someone else (possibly a fellow classmate or a student who has taken this course before) do all or much of the work. In this course, such activity is considered academic dishonesty, and, if caught, you will be charged with such according to the CoC student handbook. This is a serious offense and will affect the remainder of your academic career!

A “gray zone” with regard to plagiarism has to do with lesser forms of collaboration on assignments. Is it OK to discuss the assignment with someone else? If the “someone else” is your instructor, then yes, it’s OK.

If you are discussing the assignment with anyone else, then it depends on what you mean by “discuss”. If you are just talking about the assignment and not taking notes, typing, writing on the board, or recording the discussion in any way other than “in your head” then I feel this is OK (for this course). You walk away from the discussion with “nothing in hand” and continue to design and develop your own solution with (hopefully) a better understanding of the problem.

On the other hand, if your “discussion” leads to code or an outline or sketch for the solution (pseudo-code, flowchart, or the like) being draw-up (by hand, typed, or electronically manipulated or transferred) then you have crossed the line into academic dishonesty. I can’t say it strongly enough: Do your own work! If in doubt, ask me.

- **E-mail policy:** The best way to contact me “off hours” is via e-mail. Please expect a reasonable 1-day turnaround time for any e-mail inquiries (2-days if sent just prior to or on the weekend). Also, often times students e-mail me questions concerning “bugs” in their program. It is important to note that it can be very difficult for me to help you debug a program using e-mail: turnaround
times are too slow and interaction possibilities are limited. For this reason I will make an attempt to help debug your program via e-mail, but don’t be disappointed or surprised if we eventually need to meet face-to-face to figure things out.

- **Special Needs** Any student who, because of a disability, may require special arrangements in order to meet course requirements should contact me as soon as possible to make necessary arrangements. The instructor will require a Professor Notification Letter (PNL).

**Topics**

1. Graphics Systems and Models
2. Graphics Programming
3. Input and Interaction
4. Geometric Objects and Transformations
   4.0. Spaces and Matrices (Appendices B and C)
   **Plus all this chapter**
5. Viewing
6. Lighting and Shading (**2/3 of semester** at end of this topic)
7. From Vertices to Fragments
   7.2. Four Major Tasks
8. Discrete Techniques
   8.7. Texture Mapping
   8.11. Compositing
   8.12. Multirendering and the Accumulation Buffer
   8.13. Sampling and Aliasing
9. Programmable Shaders
   9.1. Programmable Pipelines
   9.2. Shading Languages
   9.4. The OpenGL Shading Language
   9.6. Linking Shaders with OpenGL programs
   9.7. Moving vertices
   9.8. Vertex Lighting with Shaders
   9.9. Fragment Shaders
   9.10. Per-Vertex Versus Per-Fragment Lighting
   9.13. Bump Mapping
11. Procedural Methods
   11.8. Recursive Methods and Fractals
13. Advanced Rendering
   13.2. Ray Tracing