**Description:** We focus on the theory and practice behind the three-dimensional representation and generation of geometry and other objects in the computer, and rendering of those objects within a computer graphics context. Topics include rays, curves, surfaces, volumes, rendering, shading, images, animation, simulation, and application to various disciplines and industries. We implement many of these concepts in C++ code to manipulate and visualize them. An understanding of their nature and technology is essential to anyone who will be making use of computer graphics in their academic or professional career.

**Outcomes:** This course will provide a thorough grounding in the theory and practice of 3D computer graphics. It is designed to prepare students to:

- understand the pipeline of graphics operations;
- write their own software for 3D graphics objects and rendering; and
- undertake creative work and research involving 3D computer graphics.

Students participate in lectures, read and discuss textbooks and online material, take written exames, and complete a series of exercises on the computer that involve programming and making use of graphics libraries.

**Topical Outline** Specific topics are flexible depending on the needs of the students as the course progresses. The topics covered will be a subset of this list:

- The pillars of 3D computer graphics
  - modeling geometry
  - lighting and shading geometry
  - viewing 3D worlds
  - sampling 3D worlds
- Points, Vectors and Vector Algebra
  - what is a vector?
  - geometric interpretation of a vector
  - additive operations
  - dot product
  - cross product
  - solving geometric problems with vector algebra
- Storage and display of Images
  - pixmaps
  - greyscale and RGB color
  - color systems and color spaces
  - ACES and OpenColorIO
  - color displays and output devices
  - LUTS (http://www.redsharknews.com/post/item/2966-the-beginners-guide-to-luts)
  - image file formats and libraries
  - conversion between formats
  - OpenEXR and OpenImageIO
  - OpenGL routines
- Geometry I
- lines
- planes
- spheres

**Rendering I**
- ray casting
- ray intersection with geometry
- a raycasting renderer for planes and spheres

**Shading I**
- lighting
- ambient, diffuse and specular reflection
- Phong shading model

**Geometry II**
- polygons
- polygonal surfaces and data structures
- implicit and parametric surface representations
- blobbies
- quadrics

**Rendering II**
- global vs. local illumination
- rendering equation
- recursive raytracing as a solution
- accelerated raycasting

**Matrices and matrix algebra**
- matrix representation
- matrix-vector multiplication
- the 2D and 3D affine transforms

**Coordinate systems and camera models**
- coordinate system transformations
- representations for rotations
- camera models

**Rigging and Animating Geometry**
- rigging concepts
- forward kinematics
- inverse kinematics
- weighting models
- keyframe animation
- procedural animation
- pose space deformation
- delta-mush

**Rendering III**
- screen space vs. object space renderers
- projection systems, orthographic and perspective
- homogeneous coordinates
- scan conversion (rasterization)

**Geometry III**
- scene graphs and hierarchical modeling
- Open Scene Graph
- OBJ file format
- alembic file format

**Shading II**
- bidirectional reflectance distribution function
– procedural shading
– texture maps and antialiasing
– Open Shading Language

• Geometry IV
  – polynomial representation of parametric curves
  – piecewise cubic curves
  – biparametric surfaces
  – subdivision surfaces

• Rendering IV
  – ambient occlusion
  – image based lighting
  – path tracing
  – photon mapping

• Particles
  – particle clustering
  – perlin noise
  – attributes
  – sprites

• Simulation I
  – Newton’s Laws
  – discrete time stepping
  – ballistic motion
  – damped/driven harmonic oscillator
  – rigid bodies

• Simulation II
  – fluids – gases
  – fluids – liquids
  – goo
  – oceans
  – cloth

• Volumes
  – density
  – ray marching
  – rendering equation: radiative transfer
  – level sets
  – OpenVDB

Prerequisites: CPSC 212 and MTHSC 311, or DPA 401, or permission of instructor.

Assignments: Most homework assignments involve programming in C++ and require the use of the open source libraries OpenImageIO, OpenGL, GLUT, and others. Work may be done on any computer supporting C++ and the necessary libraries. However, before turning in an assignment, the program must be compiled and tested under the School of Computing’s Ubuntu Linux environment, and both a working compile script (Makefile, CMakeLists.txt, etc.) and README must be provided. It is acceptable for students to access a suitable School of Computing Linux computer via a browser directed to [https://virtual.computing.clemson.edu](https://virtual.computing.clemson.edu). In order to turn in programming assignments, all students will need to use their computer science account and the handin system: [https://handin.cs.clemson.edu](https://handin.cs.clemson.edu). All students enrolled in CS courses should automatically be assigned CS accounts. You will need to login early in the semester to change your password, or the account may be expired. If you have problems logging in, send an email to ithelp@clemson.edu from your Clemson email account, or stop by on the first floor McAdams with a picture ID. More information here: [http://www.cs.clemson.edu/help/](http://www.cs.clemson.edu/help/). Each assignment has a due date, but no points are deducted for turning in an assignment after the due date. Assignment grades will be posted in CANVAS. Students in CPSC 6050 will have more in-depth project assignments and written exams than students in CPSC 4050.
**Grading:** The grade is a combination of class participation, written exams, and project assignments. There are 5 projects in this class. Each is worth 10 points. There are also 10 points possible for class participation (which means you *really should* participate actively in class, and *attend* class, whether in-person or online). There are 2 written exams during the semester (10 pts each), and a final exam (20 points). There is a total of 100 possible points. The grade is relative to the percentage of 100 points achieved. Grades will be posted in CANVAS. The due date for each assignment will be strongly enforced.


**Additional Reading:** Many more documents recommended in class.

**Communications:** Communications between the students and instructor/TA will be via the following mechanisms:

1. **Zoom:** this will be the video conference platform for office hours. Although office hours will be in-person, students will have the option of attending office hours via Zoom. Just prior to the start of the office hours, a zoom invitation will be generated and posted on Canvas.
2. **Ensemble:** Lectures will be recorded and posted to ensemble.clemson.edu at the URL listed at the top of the syllabus.
3. **Canvas:** Announcements for the class will be posted to canvas. Canvas will also be the location where assignment grades are posted.
4. **Email:** This mechanism is available as a means of having one-on-one conversations if needed. In unforeseen circumstances, announcements to the class may be posted via email.
5. **Office visit:** In addition to regular office hours, we can schedule an in-person office visit. Office visits, including office hours, can be attended via Zoom video conference.

The primary tool for communications from instructor to students will be Canvas (announcements, grades).

**POLICIES**


**Late Instructor Policy** If the instructor or a lab instructor is late to class or labs, then students should wait at least 15 minutes and check the course announcements before leaving.