

CS 5721: Computer Graphics (4)**Catalog Description:**

Mathematics for computer graphics, basic raster algorithms, 2D and 3D transformations and viewings. The graphics pipeline including visible surface determination, shading, transformations, and viewings. The graphic pipeline including visible surface determination, shading, ray-tracing, texture mapping, and clipping. Data structures: triangle meshes, scene graphs, ray-tracing, texture mapping, and clipping. Data structures: triangle meshes, scene graphs. Graphics applications using software systems such as Open GL.

Textbook: P. Shirley et al., *Fundamentals of Computer Graphics, 2nd Ed.*, AK Peters, 2005.

References: *The Open GL Programming Guide: The Official Guide to Learning OpenGL, Version 2, 5th Ed.*, Addison-Wesley, 2005.

Course Goals:

The course will introduce the fundamentals of computer graphics that are used to draw 2D and 3D computer rendered images. The topics presented will focus on the concepts necessary for 2D line and triangle rasterization, color representation, vector and matrix mathematics, 3D viewing, and hidden surface elimination. Additional topics may include image texturing, graphics hardware programming, ray tracing, and/or visual perception. Lab time will be used to relate the lecture topics with contemporary computer graphics hardware using OpenGL. Over the course of the semester, students will implement different portions of the 3D graphics pipeline and will have developed an understanding for how 2D and 3D graphics is currently programmed.

Prerequisites by Course & Topic

CS 2511: Software Development – use of software libraries, design of large-scale software

MATH 2326: Introduction to Linear Algebra and Mathematical Reasoning – vectors, transformations

Major Topics Covered in the Course

- 2D line and triangle rasterization
- Color representation
- Vector and matrix mathematics
- 3D viewing
- Hidden surface elimination
- Additional topics may include image texturing, graphics hardware programming, ray tracing, and/or visual perception.
- OpenGL

Class/Laboratory Schedule: Lecture: 3 hours per week, Laboratory: 1

Course Outcomes

1. Ability to design and implement 2D and 3D graphics programs.
 - a. Analyze the requirements for a graphics program.
 - b. Using the requirements, design a graphics program or modifications of such a program.
 - c. Implement the desired functionality using calls to graphics package routines.
 - d. In the 3D case, implement script-controlled and user-controlled animation.
 - e. Thoroughly test each implementation to show that it meets the requirements.
2. Familiarity with the software architecture of 2D and 3D graphics packages.
 - a. Understand the implementation issues for graphics output primitives.
 - b. Understand the implementation of input interaction for 2D and 3D graphics packages.
 - c. Understand the use of graphical resources such as color, fonts, and off-screen pixmaps.
 - d. Understand how a retained-mode graphics package with a hierarchical structure works.
3. Familiarity with mathematical transformations used in 2D and 3D graphics.
 - a. Understand the matrix representations of 2D and 3D transformations.
 - b. Understand the efficiency issues that arise when implementing such transformations.

- c. Understand how to implement more complex transformations that are provided by standard graphics packages.
- 4. Familiarity with mathematical transformations and the 3D viewing operation.
 - a. Understand how modeling transformations are used and implemented in a 3D hierarchical graphics system.
 - b. Understand how views are specified and how they determine the view orientation transformations.
 - c. Understand the different ways clipping can be incorporated into the 3D viewing operation.
 - d. Understand the output pipeline in a 3D graphics package.
- 5. Ability to design and implement basic 2D and 3D graphics algorithms.
 - a. Design basic algorithms, such as line generation, polygon filling, clipping, pick correlation, visible surface determination, and rendering.
 - b. Comparatively analyze such algorithms for time and space efficiency.
 - c. Implement 2D and 3D graphics algorithms.
 - d. Thoroughly test the implementations to show that they are correct and argue that they have the desired performance with respect to efficiency.

Relationship to Program Outcomes

A student must have completed introduction to linear algebra and software design and analysis before taking CS 5721. This course contributes to meeting the following program outcomes:

- 1. *Students understand the mathematics and statistics that underlie scientific applications.*
 Students learn about the theory of 2D and 3D transformations, their matrix representation, and the construction of more complex transformations from simpler ones via composition. Students also learn about the theory of projections and the specification of a 3D view. Outcomes 3 and 4 map to this program outcome.
- 2. *Students can design, develop, and analyze significant software systems.*
 Students gain additional experience in this area by designing and implementing complex graphics program based on graphics software libraries. This is done over a sequence of assignments, during which attention must be paid to maintenance issues. Course outcomes 1-5 map to this program outcome.
- 3. *Students understand the fundamentals of computer organization and architecture, data structures and related algorithms, and programming languages.* Course outcomes 1-5 map to this program outcome.
 Students use data structures and basic algorithms to design and implement complex graphics software.
- 4. *Students can apply computer science principles and practices to a variety of problems.*
 Students design and implement end user graphics application programs, interactive 3D programs, and rendering programs. Course outcomes 1-5 map to this program outcome.

Assessment Plan for Course:

This course is assessed every third year by the instructor and a course assessment document covering all of the course outcomes and their effect on the program outcomes is prepared.

Estimate CSAB Category Content

	CORE	ADVANCED		CORE	ADVANCED
Data Structures			Computer Organization and Architecture Concept of Programming Languages		
Algorithms		1			
Software Design		1			

Coordinator/Prepared by: P. Willemsen