COURSE DESCRIPTION

Department and Course Number	CS450	**Course Coordinator**	Eun-Young (Elaine) Kang
Course Title	Computer Graphics	**Total Credits**	4

Current Catalog Description:

An advanced course in computer graphics with emphasis on rendering. Polygonal meshes, Bezier curves and surfaces, line and back face removal, shading, lighting, and texture algorithms.

Textbook:

F.S. Hill Jr., *Computer Graphics using Open GL, 2nd Ed.*, Prentice Hall, 2001.

References:

- Angel, Edward, *Interactive Computer Graphics – A Top-Down Approach Using OpenGL, 3rd Ed.,* Addison Wesley, 2003.
- Hearn, D and Baker, M. *Computer Graphics with OpenGL, 3rd Ed.*, Prentice Hall, 2004
- J. Foley, et al, *Computer Graphics-Principles and Practice, 2nd Ed.*, Addison Wesley, 1996.

Course Goals:

At the end of the course, students are able to understand

- The complete 3D graphics rendering pipeline.
- How to achieve visual realism through shading, texture mapping, and shadow.
- Hidden surface removal.
- Understand aliasing problems and anti-aliasing methods.
- Non-polygonal mesh objects such as Bezier curves, B-Spline, and NURB.
- Advanced OpenGL capabilities.

These course goals contribute to the success of **Student Learning Outcomes 1.a, 2, 3, 5, and 6**.

Prerequisites by Topic:

- Good programming skills in Java or C++
- Basic knowledge of data structures
- Basic knowledge of linear algebra such as matrix and vector space
- Basic knowledge of 2D rasterization process
- Basic knowledge of coordinate conversion, camera coordinate and view changes
- Basic knowledge of 3D graphics rendering pipeline
- Basic programming skill in OpenGL

Major Topics Covered in the Course:

- Review of linear algebra
- Review of three dimensional viewing and graphics rendering pipeline
    1. Camera coordinate
    2. Coordinate conversion
    3. Projection of 3D objects
- Local shading models (Flat, Gouraud, and Phong)
- Hidden surface removal (Backface removal, Z-buffer, Painters algorithm, BSP tree)
- Shadows
- Texture mapping (Mip-mapping, Environment mapping, and Procedural texture)
- Curve (Bezier curve, B-spline and NURB)
- OpenGL with advanced features such as shading and texture maps

Laboratory Projects (specify number of weeks on each):

Through out the quarter, students are required to work on homework or a project.

- Week 1-2: Review with graphics rendering pipleline and OpenGL.
- Week 3-4: Implement flat, Gouraud, and Phong shading models using a polygonal mesh input.
- Week 4-7: Implement texture mapping using a polygonal mesh input.
- Week 7-10: Implement an user interactive 3D renderer with shading and texture mapping features using OpenGL.

Estimate Curriculum Category Content (Quarter Hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		0.5	Data Structures		0.5
Software Design		1.0	Prog. Languages		2.0
Comp. Arch.					

Oral and Written Communications:

Written documentation of software built in labs and homework assignments.

Social and Ethical Issues:

No significant component.

Theoretical Content:

Vectors, Matrices, Bezier Curves.

Problem Analysis:

Students are required to understand the fundamental processes of the 3D graphics rendering pipeline. They are also required to understand the advanced features of rendering such as shading and texture mapping and to know how to define the underlying geometric and mathematical representations of the objects involved. They must then translate the representations into algorithms and program codes.

Solution Design:

Solution design adapts a progressive approach. The processes of the rendering pipeline are decomposed into independent steps with a well defined API that connects each to the next. Students complete one step, extend the codes in the next step, and produce integrated codes at the end. In addition, solution design involves understanding how to use OpenGL for such essential data structures as vectors, matrices, and matrix stacks. By making use of the well-established data representation of OpenGL, students learn how to develop efficient programs, and they develop the ability to make a smooth transition from a renderer program written in a conventional language to a renderer implemented using OpenGL.