Advanced Computer Graphics
CSCI B581 - Fall 2014 - course section 18334

Instructor
Mitja Hmeljak
mitja.indiana.edu

Sections
Lecture Section 18334:
BH141 (Ballantine Hall),
MW 9:30AM - 10:45AM

Office Hours
TBA,
SoIC Undergraduate Annex 203

Lab
optional (recommended) weekly lab
Lab Section 31227:
LH030 (Lindley Hall),
F 4:00PM - 5:15PM
(or other time arranged on request)

Textbook:
Required: "Interactive Computer Graphics with WebGL, 7/E"
by Edward Angel and Dave Shreiner - 2014


Course Description:
• Introduction to graphics hardware and software.
• Two-dimensional graphics methods, transformations, and interactive methods.
• Three-dimensional graphics, transformations, viewing geometry, object modeling and interactive manipulation methods.
• Basic lighting and shading.
• Video and animation methods.
• Credit not given for both B581 and B481.

Course Prerequisites:
Required: CSCI-C 343, MATH-M 301 or M 303 or equivalent experience (data structures & mathematics for vectors and matrices)
Recommended: C212, JavaScript (or other C-like language) programming experience.

Planned special topics for Fall 2014:
WebGL is the 3D graphics API implemented right into the web browser, without the need of plugins.
WebGL is based on OpenGL ES 2.0, exposed through the HTML5 Canvas element as Document Object Model interfaces. OpenGL ES 2.0 and WebGL are very similar: both are Shader-based API using GLSL, with semantically similar constructs. WebGL specification is very close to the OpenGL ES 2.0 specification, with some concessions made for what developers expect out of memory-managed languages such as JavaScript.
Course Objectives:

Course Goals:
B581 is the main course in computer graphics for Computer Science graduate students. Graphics is a crucial part of many interfaces and applications. A basic knowledge of computer graphics (CG) is useful for any task that involves a computer. This course concentrates on:

- fundamental principles of displaying 2D information using computer graphics
- 3D programming for geometry modeling, transformation, viewing, lighting and shading
- interactive techniques for real-time computer graphics
- computer graphics for the web

For the Fall 2014 semester, we plan to learn WebGL http://www.khronos.org/webgl/ programming for web browsers.

Learning Objectives:
By the end of this semester, students should be able to:

- understand the fundamental stages and mathematical foundation of the 3D polygon projection rendering pipeline
- develop interactive 2D and 3D graphics applications using WebGL and OpenGL ES
- describe the relative advantages and disadvantages of low-, middle-, and high-level computer graphics
- identify state-of-the-art trends in graphics hardware, display devices, and rendering methods

Additional considerations
There will be approximately six coding assignments, and six written-question homework assignments. You will be writing JavaScript code to program the WebGL API. Modern web browsers and programmer’s text editors will be adequate for development. Other programming languages, APIs or OSes will not be considered for assignments submitted for grading.

Exams and Grades
- Final Exam: 25%
- Midterm: 15%
- Coding assignments and projects: 40%
- Readings, written assignments, in-class presentations: 15%
- Participation, active class presence and journal note-taking: 5%
Course Topics

Note: the following outline is a superset of what we'll actually have time to cover during the semester. It will be refined as the Fall 2014 semester start approaches. Study guides for mid-term and final exams will specify which material will be tested, such as amount of details and emphasis on specific topics.

Introduction:


Two-Dimensional Concepts:

2D geometry, vectors, vector products, normals, matrix transformations, and explicit coordinate transformations. Intersection calculations.

Scan conversion of lines. Fundamental implicit and parametric equations. Fundamental properties of scan conversion algorithms (integer arithmetic, local recursion equations), construction of the algorithms. Error measure concept and how you use it.

2D geometry. Know operations and transformations on 2D vectors. Affine transformations, their properties and inverse transformations. Translation, rotation, scaling and shearing in 2D. Explicit equations. Homogeneous coordinates. Matrix representations, the "fixed point rule" not just for displaced fixed point but also rotated coordinate axes. How to manipulate 2D objects in world coordinate system and screen coordinate system; going from "ideal" to world coordinate systems. Line-line and point-line geometry - how to find nearest point. Inside-outside tests. Signed area measurements.

Clipping. Basic motivation and definitions; why is it important? Equations needed to clip lines and polygons. Basic principles used in standard line and polygon clipping, including Cyrus-Beck and Cohen-Sutherland algorithms.

Curves and Splines. Basic properties and ideas. Implicit, explicit and parametric representations of lines and curves. Principles of constructing splines with higher derivatives from linear interpolations (actually linear interpolation is just the simplest spline). Qualitative ideas of matching derivatives on successive curve segments. Contrast Bezier, Catmull-Rom, B-spline cubic spline properties (no need to memorize explicit coefficients). Basis functions.

Three-Dimensional Concepts:

Math and Geometry for 3D Graphics. 3D vectors, homogeneous coordinates. 3D Transformations - Translation, Scaling, Rotation, Reflection, Shear. Composite Transformations.

3D shape representation. 3D surfaces - polygon, curved (Bezier, spline), surface generation methods.

Hidden Surface and Line Removal Methods.

3D Pipeline and Mid-level APIs. WebGL vs. OpenGL. Special Effects Buffers. API-specific operations. GPU-specific operations.

Light, Surfaces, Scenes and Animation. Color: how are specific colors constructed; additive and subtractive color models; metamers.

Modeling Light and Shading. Light sources and ambient light. Diffuse and specular reflection, refraction. Texture and surface patterns.


High-level APIs. Historical examples. Scene Graph Scripting.

**Advanced Topics:**