CPSC 314
Computer Graphics

Wolfgang Heidrich

People

Instructor:
• Wolfgang Heidrich

TA(s):
• Christopher Batty
• Abhijeet Ghosh
Course Organization

**Components:**
- Lectures
- Homework problems
- Labs
- Programming assignments (3+1)
- Quizzes (2)
- Final

Course Organization

**Grades and Grading**
- Programming assignments: 35% (10% each)
  - 5% for assignment 0
- Quizzes: 25%
  - (10% for first quiz)
- Final: 40%

**Homework problems**
- NOT graded
- BUT: essential preparation for quizzes/final
- Solutions discussed in lab sessions
Course Organization

**Programming assignments:**
- C++, Windows or Linux
- OpenGL graphics library / GLUT for user interface

**Collaboration policy:**
- No collaboration on programming assignments
- Reference all external resources

Course Organization

**Up-to-date information:**
- [http://www.ugrad.cs.ubc.ca/~cs314](http://www.ugrad.cs.ubc.ca/~cs314)
- WebCT (follow link from course home page)
  - Bulletin board
  - Reporting of grades
Books

**Textbook:**
  - *We are not going to follow this text very closely*

**Other Books:**
  - *This one is online: see course page*

Learning OpenGL

**This is a graphics course using OpenGL**
- Not a course *on* OpenGL

**Learning API mostly on your own**
- Only minimal lecture coverage
  - *Basics, some of the tricky bits*
- Also: ask in the labs
- OpenGL Red Book
- many tutorial sites on the web
  - *nehe.gamedev.net*
What is Computer Graphics?

Create or manipulate images with computer
- this course: algorithms for image generation

What is CG used for?

Graphical user interfaces
- Modeling systems
- Applications

Simulation & visualization
What is CG used for?

**Movies**
- Animation
- Special effects

**Computer games**
What is CG used for?

**Images**

- Design
- Advertising
- Art

Real or CG?

http://www.autodesk.com/eng/etc/fakeorfoto/quiz.html
Real or CG?

2

Real or CG?

3
Real or CG?

What This Course Is About

Topics covered

• Fundamental algorithms of computer graphics
• Interactive graphics:
  – The rendering pipeline
    ▪ Abstract model for the functioning of graphics hardware and interactive graphics systems
  – Color spaces and reflection models
  – Shadow algorithms
• Ray-tracing
• (Global illumination)
What This Course is NOT About

Topics NOT covered:
• Artistic and design issues
• Usage of commercial software packages
• Applications (i.e. game design)

Topics covered with little detail:
• Animation, Geometric Modeling
  – These have separate undergrad classes
  – CPSC 424 (Geometric Modeling) next year

Syllabus

Overview

The Rendering Pipeline (1)
• Geometry transformations, linear, affine, and perspective transformations
• Lighting/illumination
• Clipping of lines and polygons
• Vertex arrays, triangle strips, display lists
Syllabus

The Rendering Pipeline (2)
- Scan conversion of lines and polygons
- Shading and interpolation
- Texture mapping

The Rendering Pipeline (3)
- Modern hardware features
- Vertex shaders / register combiners etc.

Syllabus

Color and reflection
- Color spaces and tristimulus theory
- Physical reflection models

Shadow Algorithms
- Shadow volumes and shadow maps

Ray-tracing

(Global illumination)
- Only if there is time
The Rendering Pipeline – An Overview

Wolfgang Heidrich

3D Graphics

Modeling:
• Representing object properties
  – Geometry: polygons, smooth surfaces etc.
  – Materials: reflection models etc.

Rendering:
• Generation of images from models
  – Interactive rendering
  – Ray-tracing

Animation:
• Making geometric models move and deform
Rendering

**Goal:**
- Transform computer models into images
- May or may not be photo-realistic

**Interactive rendering:**
- Fast, but until recently low quality
- Roughly follows a fixed patterns of operations
  - Rendering Pipeline

**Offline rendering:**
- Ray-tracing
- Global illumination

---

Rendering

**Tasks that need to be performed (in no particular order):**
- Project all 3D geometry onto the image plane
  - Geometric transformations
- Determine which primitives or parts of primitives are visible
  - Hidden surface removal
- Determine which pixels a geometric primitive covers
  - Scan conversion
- Compute the color of every visible surface point
  - Lighting, shading, texture mapping
The Rendering Pipeline

What is it? All of this:

- Abstract model for sequence of operations to transform a geometric model into a digital image
- An abstraction of the way graphics hardware works
- The underlying model for application programming interfaces (APIs) that allow the programming of graphics hardware
  - OpenGL
  - Direct 3D

Actual implementations of the rendering pipeline will vary in the details
The Rendering Pipeline

Geometry Database

Geometry database:
- Application-specific data structure for holding geometric information
- Depends on specific needs of application
  - Independent triangles, connectivity information etc.

The Rendering Pipeline

Model/View Transformation

Modeling transformation:
- Map all geometric objects from a local coordinate system into a world coordinate system

Viewing transformation:
- Map all geometry from world coordinates into camera coordinates

© Wolfgang Heidrich
The Rendering Pipeline

**Lighting:**
- Compute the brightness of every point based on its material properties (e.g. Lambertian diffuse) and the light position(s)

---

The Rendering Pipeline

**Perspective transformation**
- Projecting the geometry onto the image plane
- Projective transformations and model/view transformations can all be expressed with 4x4 matrix operations
**The Rendering Pipeline**

**Clipping**

- Removal of parts of the geometry that fall outside the visible screen or window region
- May require *re-tessellation* of geometry

**The Rendering Pipeline**

**Scan Conversion**
**Scan conversion**

- Turning 2D drawing primitives (lines, polygons etc.) into individual pixels (*discretizing/sampling*)
- Interpolation of colors across the geometric primitive
- This yields a *fragment* (pixel data associated with a particular location in the final image and color values, depth, and some additional information)
Texture mapping

- “gluing images onto geometry”
- The color of every fragment is altered by looking up a new color value from an image
**The Rendering Pipeline**

**Depth Test**

*Depth test:*
- Removes parts of the geometry that are hidden behind other geometry
- Test is performed on every individual fragment
  - *we will also discuss other approaches later*

---

**The Rendering Pipeline**

**Blending**

- Geometry Database
  - Model/View Transform
  - Lighting
  - Perspective Transform
  - Clipping
  - Scan Conversion
  - Texturing
  - Depth Test
  - Blending
**The Rendering Pipeline**

**Blending:**
- Fragments are written to pixels in the final image.
- Rather than simply replacing the previous color value, the new and the old value can be combined with some arithmetic operations (blending).
- The video memory on the graphics board that holds the resulting image and is used to display it is called the framebuffer.

---

**The Rendering Pipeline**

1. Geometry Database
3. Lighting
4. Perspective Transform.
5. Clipping
6. Scan Conversion
7. Texturing
8. Depth Test
9. Blending
10. Framebuffer

© Wolfgang Heidrich
Discussion

**Advantages of a pipeline structure**

- Logical separation of the different components, modularity
- Easy to parallelize:
  - *Earlier stages can already work on new data while later stages still work with previous data*
  - *Similar to pipelining in modern CPUs*
  - *But much more aggressive parallelization possible (special purpose hardware!)*
  - *Important for hardware implementations!*
- Only local knowledge of the scene is necessary

**Disadvantages:**

- Limited flexibility
- Some algorithms would require different ordering of pipeline stages
  - *Hard to achieve while still preserving compatibility*
- Only local knowledge of scene is available
  - *Shadows*
  - *Global illumination*
Coming Up...

**Wednesday, May 10:**
- More details on the rendering pipeline

**Friday, May 12:**
- Geometric transformations