Course Information

- INSTRUCTOR: Professor Jing Hua (jinghua@cs.wayne.edu),
- CREDITS: 3
- PREREQUISITES:
  - C/C++, Linear Algebra
- LECTURES: Tuesday & Thursday, 11:30AM - 12:45PM, 314 State Hall
- OFFICE HOURS: Tuesday, 10:30AM - 11:30PM (5057 Woodward Suite 14109.1)

Grading Scheme

- Class attendance and performance (5%)
- Assignments
  - Assignment 0 (0%)
  - Assignment 1 (20%)
  - Assignment 2 (20%)
- Proposal, Course Project and Final Report (55%)

Textbooks

- RECOMMENDED TEXTBOOK:
  Edward Angel, Dave Shreiner

  The Visualization Toolkit, Prentice Hall PTR
  OpenGL programming guide

Goal

- A comprehensive overview of computer graphics
- The graphics and visualization pipeline
- State-of-art techniques in computer graphics related fields (shape analysis)
- Future research and work in computer graphics and vision

Collaboration Policy

Goal

- A comprehensive overview of computer graphics
- The graphics and visualization pipeline
- State-of-art techniques in computer graphics related fields (shape analysis)
- Future research and work in computer graphics and vision
Content

• Introduction
  – Overview, definition
  – Various application examples and areas
  – Graphics history
  – Graphics software and hardware systems
  – Graphics programming
  – User-computer interface

• Geometry and Mathematics
  – Curves, and surfaces
  – Solid and volumetric models
  – 3D geometric transformation
  – Data structures

• Geometric Modeling
  – Curves
  – Surfaces
  – Shape Modeling and Matching
  – Shape parameterization
  – Solids

• Visualization
  – Volume rendering
  – VTK
  – Applications

• Rendering
  – Object hierarchies
  – Ray tracing
  – Object and image order rendering
  – Rendering pipeline
  – Color perception and color models
  – Basic optics
  – Visibility

• Image-based techniques
  – Sampling
  – Filtering
  – Anti-aliasing
  – Image analysis and manipulation
Content (cont.)

- Advanced Topics
  - Animation
  - Transparency and shadows
  - Texture mapping
  - Image-based rendering and modeling
  - Advanced modeling techniques
  - Case studies
  - Software packages
  - ..........

What is Computer Graphics?

- Process of generating images using computers
- This is called rendering
- A rendering algorithm converts a geometric model into a picture
- This process is called scan conversion or rasterization
- How does visualization fit in here?

Content (cont.)

- Learn something you are interested through the course project
  - Interesting topic
  - Hand-on experience
  - Special instruction

Computer Graphics

- Computer graphics consists of:
  1. Modeling (representations)
  2. Rendering (display)
  3. Interaction (user interfaces)
  4. Animation (combination of 1-3)
- Usually “computer graphics” refers to rendering

A Classical Classification

Overview

Image Processing

2D Images

Computer Graphics

3D Models

Computer Vision
Surface Graphics

- Surface representations are good for objects that have **homogeneous** material distributions and/or are not **translucent** or **transparent**
- Such representations are good only if object boundaries are important
- Examples: furniture, mechanical objects, plant life
- Applications: video games, virtual reality, computer-aided design

Surface Graphics – Pros and Cons

- Good: explicit distinction between inside and outside makes rendering calculations easy and efficient
- Good: hardware implementations are inexpensive
- Good: can use tricks like texture mapping to improve realism
- Bad: an approximation of reality
- Bad: does not let us peer into and through objects

Volume Graphics

- Surface graphics doesn’t work so well for clouds, fog, gas, water, smoke and other **amorphous phenomena**
  “amorphous” = “without shape”
- Surface graphics won’t help us if we want to explore objects with very complex internal structures
- Volume graphics provides a solution to these shortcomings of surface graphics
- Volume graphics includes **volume representations** and **volume rendering** algorithms to display such representations

Applications

- Video games, virtual reality, computer-aided design

Computer Animation

- Computer Animation

Computer Aided Design (CAD)

- Computer Aided Design (CAD)
Digital Art

Visualization Pipeline

Graphics Pipeline

Pipelines

Rendering Pipeline

Lights, Cameras and Objects

- How are we able to see things in the real world?
- What’s the process that occurs?
- I’ll get you started:
  1. Open eyes
  2. Photons from light source strike object
  3. Bounce off object and enter eye
  4. Brain interprets image you see
Lights, Cameras and Objects

- Rays of light emitted by light source
- Some light strikes object we are viewing
  - Some light absorbed
  - Rest is reflected
  - Some reflected light enters our eyes

Surface Rendering

- We have considered interaction between light rays and object boundaries
- This is called surface rendering and is part of surface graphics
- Computations take place on boundaries of objects
- Surface graphics employs surface rendering to generate images of surface representations

Lights, Cameras and Objects

- How do we simulate light transport in a computer?
- Several ways
  - Ray-tracing is one
  - Start at eye and trace rays the scene
  - If ray strikes object, bounces, hits light source → we see something at that pixel
  - Most computer applications don’t use it. Why?
  - With many objects very computationally expensive

Mathematical Surfaces

- Equation of a sphere:
- How thick is the surface?
- Are there objects in real world thickness zero?

Surface Ray-Tracing

- Can you think of objects or phenomena for which this approach to rendering will fail?
- When is a surface representation not good enough?
- Would a surface representation suffice to represent the internal structure of the human body?

Surface Graphics
**Volumetric Representations**

- A volumetric data-set is a 3D regular grid, or 3D raster, of numbers that we map to a gray scale or gray level.
- Where else have you heard the term raster?
- An 8-bit volume could represent 256 values \([0, 255]\).
- Typically volumes are at least \(200^3\) in size, usually larger.
- How much storage is needed for an 8-bit, \(256^3\) volume?

**Volume Rendering**

- In volume rendering, imaginary rays are passed through a 3D object that has been discretized (e.g., via CT or MRI).
- As these viewing rays travel through the data, they take into account of the intensity or density of each datum, and each ray keeps an accumulated value.

**Volume Graphics**

- Volumetric objects have interiors that are important to the rendering process (what does that mean?)
- Interior affects final image.
- Imagine that our rays now don’t merely bounce off objects, but now can penetrate and pass through.
- This is known as **volumetric ray-casting** and works in a similar manner to surface ray-tracing.

**Volume Rendering**

- As the rays leave the data, they comprise a sheet of accumulated values.
- These values represent the volumetric data projected onto a two-dimensional image (the screen).
- Special mapping functions convert the grayscale values from the CT/MRI into color.

**Volumetric Ray-Tracing**

- **Semi-transparent rendering**
Volume Graphics

- Good: maintains a representation that is close to the underlying fully-3D object (but discrete)
- Good: can achieve a level of realism (and "hyper-realism") that is unmatched by surface graphics
- Good: allows easy and natural exploration of volumetric datasets
- Bad: extremely computationally expensive!
- Bad: hardware acceleration is very costly ($3000+ vs $200+ for surface rendering)

Surface Graphics vs. Volume Graphics

- Suppose we wish to animate a cartoon character on the screen
- Should we use surface rendering or volume rendering?
- Suppose we want to visualize the inside of a person’s body?
- Now what should approach we use? Why?
- Could we use the other approach as well? How?
- We could visualize body as a collection of surfaces

Hardware
Trackball, Joystick, Touch Pad

3D Laser Range Scanner

Haptics Device (PhanTom 1.0)

3D Camera

3D Laser Range Scanner

Digital 3D Reconstruction
Structure Light 3D Scanner

Real-time Capture

Full View Capture

High-resolution Capture

High-quality 3D Synthesis

OpenGL

- Most widely used 3D graphics Application Program Interface (API).
- Truly open, independent of system platforms.
- Reliable, easy to use and well-documented.
- Default language is C/C++.
OpenGL

- The GL library is the core OpenGL system:
  - modeling, viewing, lighting, clipping
- The GLU library (GL Utility) simplifies common tasks:
  - creation of common objects (e.g. spheres, quadrics)
  - specification of standard views (e.g. perspective, orthographic)
- The GLUT library (GL Utility Toolkit) provides the interface with the window system.
  - window management, menus, mouse interaction

• To create a red polygon with 4 vertices:
  
  ```
  glColor3f(1.0, 0.0, 0.0);
  glBegin(GL_POLYGON);
  glVertex3f(0.0, 0.0, 3.0);
  glVertex3f(1.0, 0.0, 3.0);
  glVertex3f(1.0, 1.0, 3.0);
  glVertex3f(0.0, 1.0, 3.0);
  glEnd();
  ```

  - `glBegin` defines a geometric primitive:
    - GL_POINTS, GL_LINES, GL_LINE_LOOP, GL_TRIANGLES, GL_QUADS, GL_POLYGON...
  - All vertices are 3D and defined using `glVertex`

VTK

- VTK is a C++ class library for developing visualization applications
- www.vtk.org → Manual 4.2 → Class Hierarchy
- Every non-trivial VTK program must contain the following seven elements:
  1. vtkRenderWindow – the window on screen
  2. vtkRenderer – C++ object for drawing shapes
  3. vtkLight – light to illuminate scene
  4. vtkCamera – camera

FLTK

- Fast Light Tool Kit (FLTK)
- www.fltk.org
- C++ oriented
  - A set of UI classes such as Window, box, etc.
- Can mix use with GLUT
- FLUID: fast light UI Designer
  - Fast creation of GUI
  - Automatically writes parts of GUI code from a graphical spec
  - Good for elaborate interfaces

Comments on Programming

- OpenGL, VTK, plus Glui
  - Simple, easy to program, limitations
- OpenGL, VTK, plus FLTK
  - Cross platform, more powerful
- OpenGL, VTK, plus Visual C++
  - Super!
  - Only run under windows system