This class

- Recap of first lecture
- Overview of 3D Graphics
- Building Interactive Applications using OpenGL in Java using GL4Java
- Assignment #1
- Representation and Modeling of Objects
  - Polygonal, CSG, Volumetric, Implicit

Recap from last class

- Take away:
  - Basic organization and history of raster graphics hardware
    - Structure of graphics cards
    - Architecture of CRT displays
  - Basic terminology
  - Artifacts of CRT technology on quality of rendered image

3D Graphics: Overview

- Take away:
  - Global vs. local illumination
    - The major distinction between graphics algorithms
  - Global: unsolved, active research problem
    - 2 well known approaches: ray-tracing, radiosity
    - Slow, offline algorithms
  - Local: widely used, less realistic
    - OpenGL, other immediate packages
    - Typically relies on gross approximations
    - Fast, interactive algorithms

Global vs. Local Illumination

- Global
  - Model interactions between light and objects
  - Impossible in practice!
    - Still active research problem
  - Basic idea: "Follow the light"
- Local
  - Consider each object independently

Ray Tracing

- "Trace" rays around scene
  - Can't follow photos from light source to eye
  - Rather, trace from eye into scene
    - Models direct interactions
    - Complex interactions are approximated by "ambient" term
  - Basis for modern offline algorithms
    - Films such as Toy Story

Radiosity

- Attempt to model complex interactions using heat transfer equations
  - Basic techniques limited to perfect diffusers
  - No sharp highlights or reflections
- Realistic looking ambient interiors

Local Illumination

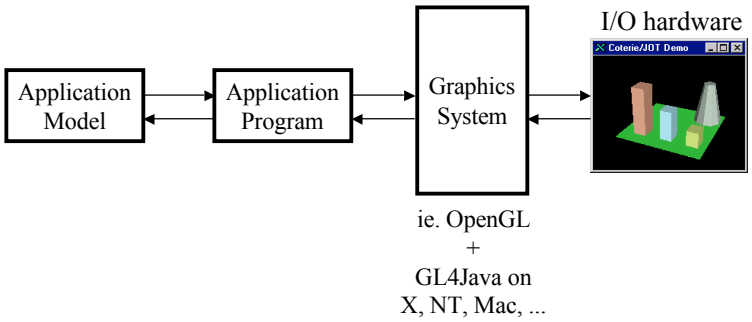
- Interaction of object and lights
  - Objects are completely defined
  - No object-object interaction
- Phong model
  - Gross simplification, no physical basis: Hack!
    - Bland, plastic look
  - Basis of modern local illumination hardware
- "What looks good and is fast?"

Local Illumination

- Add on algorithms for
  - Shadows
  - Complex surfaces
    - Texture mapping
    - Bump mapping
    - Environment mapping
- All are approximations

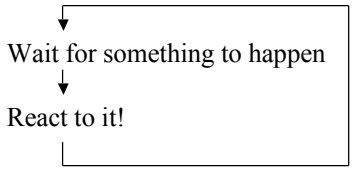
Building Interactive Applications using OpenGL in Java using GL4Java

- Conceptual Application Model



Application Control Flow

- Interactive programs are *event driven*



- What are *events*?
  - User input
  - Window system, application generated

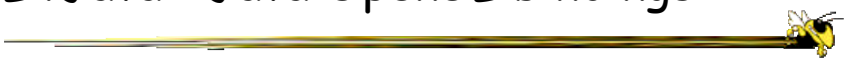
OpenGL

- 3D graphics library
  - Output only: render graphics
    - Only knows about "graphics contexts"
- Platform independent: No support for
  - Window creation
    - GLX, GLW, …
    - Hidden in GLUT (C/C++) or Java (GL4Java)
  - Input

OpenGL

- Immediate mode pipeline
  - Collection of state
    - Xforms, materials, lighting, textures, modes, …
  - Feed fragments in
    - Triangles, points, lines, quads, …
  - Operate on each ***immediately***
    - Xform, cull, shade, texture, rasterize, …
    - No memory of fragments

GL4Java: Java OpenGL bindings



- OS independent Java/OpenGL programs:
  - Multiple windows for OpenGL rendering.
  - Java AWT-style events.
  - Other features
    - Support for animation and timing.
    - Utility routines to generate various objects (built on the GLUT toolkit).
    - Support for bitmap and stroke fonts.
    - Support for texture loading.

Input

- Uses standard AWT event mechanisms

Assignment #1

- Hand out today, due Friday August 29th
- Purpose:
  - Learn how to write Java/OpenGL programs
  - Handle input, simple non-trivial output

Representation and Modeling

- Take away:
  - Different approaches are appropriate for different situations
    - Polygonal, CSG, Volumetric, Implicit
  - Rendering usually (but not always) requires a polygonal model
    - e.g., Volume rendering hardware exists

Polygonal Modeling

- Most common representation
  - Object == collection of flat polygonal faces
  - May be closed or not
- Simple to render
  - Basis for most modern hardware
- Hard to model complex objects
  - Hard to update, modify objects

Polygonal Objects

- Many possible data structures can be used
- Typically
  - Collection of vertices, V
  - Collection of edges (refer to V), E
  - Collection of faces (refer to E), F
- Other attributes associated with each
  - Color or material, normal, geometric properties, 1 or 2 sided, texture info, etc.

CSG (Constructive Solid Geometry)

- User representation
  - Define objects by combining primitive objects using set operations and linear xforms
    - Cones, spheres, cubes, …
    - Union, difference, intersection
- Compact and elegant when appropriate
  - CAD, machining

Volumetric Representations

- Divide space into regions
  - "Voxel" == smallest cubic element
- Label each as in or out of object
- Appropriate when data is in this form
  - Medical visualization (e.g., MRI data)
- Representations (octree, …) used to accelerate other algorithms
  - Organize polygonal scenes for ray tracing, …

Implicit Representation

- E.g., $x^2 + y^2 + z^2 = 1$
- Used in scientific processes
- Limited use for modeling
  - Few models can be represented this way
  - Hard to manipulate, render
- As before, useful for optimizing other algorithms
  - Bounding volumes for collision detection, …