Models and Architectures
- Basic design of a graphics system
- Graphics pipeline architecture
- Please review these basics after class

Image Formation
- Mimic the synthetic camera model to design graphics hardware and software
- Need to specify
  - Objects
  - Materials
  - Viewer
  - Lights
- How to implement

Advanced Physical Approaches
- Ray tracing: follow rays of light from center of projection until they either are absorbed by objects or go off to infinity
  - Can handle global effects
    - Multiple reflections
    - Translucent objects
  - Slow
    - Must have whole data available at all times
- Radiosity: Energy based approach
  - Very slow

General Approach
- Process objects one at a time in the order they are generated by the application
  - Consider only local lighting
- Pipeline architecture
  - All steps can be implemented in hardware on the graphics card
**Vertex Processing**

- Much of the work in the pipeline is in converting object representations from one coordinate system to another
  - Object coordinates
  - Camera (eye) coordinates
  - Screen coordinates
- Every change of coordinates is equivalent to a matrix transformation
- Vertex processor also computes vertex colors

**Projection**

- *Projection*: from 3D objects to 2D image
  - Perspective projections: all projectors meet at the center of projection
  - Parallel (orthogonal) projection: projectors are parallel, center of projection is replaced by a direction of projection

**Primitive Assembly**

Vertices must be collected into geometric objects before clipping and rasterization can take place
- Line segments
- Polygons
- Curves and surfaces

**Clipping**

Just as a real camera cannot “see” the whole world, the virtual camera can only see part of the world or object space
- Objects that are not within this view volume are said to be *clipped* out of the scene
Rasterization

- If an object is not clipped out, the appropriate pixels in the frame buffer must be assigned colors
- Rasterizer produces a set of fragments for each object
- Fragments are “potential pixels”
  - Have a location in frame buffer
  - Color and depth attributes
- Vertex attributes are interpolated over objects by the rasterizer

Fragment Processing

- Fragments are processed to determine the color of the corresponding pixel in the frame buffer
- Colors can be determined by texture mapping or interpolation of vertex colors
- Fragments may be blocked by other fragments closer to the camera
  - Hidden-surface removal

The Programmer’s Interface

- Programmer sees the graphics system through a software interface: the Application Programmer Interface (API)

API Contents

- Functions that specify what we need to form an image
  - Objects
  - Viewer
  - Light Source(s)
  - Materials
- Other information
  - Input from devices such as mouse and keyboard
  - Capabilities of system
Object Specification

- Most APIs support a limited set of primitives including:
  - Points (0D object)
  - Line segments (1D objects)
  - Polygons (2D objects)
  - Some curves and surfaces
    - Quadrics
    - Parametric polynomials
- All are defined through locations in space or vertices

Example

```c
glBegin(GL_POLYGON)
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(0.0, 1.0, 0.0);
glVertex3f(0.0, 0.0, 1.0);
glEnd();
```

Camera Specification

- Six degrees of freedom
  - Position of center of lens (COP)
  - Orientation
- Lens – focal length
- Film size (h, w)
- Orientation of film plane

Lights and Materials

- Types of lights
  - Point sources vs distributed sources
  - Spot lights
  - Near and far sources
  - Color properties
- Material properties
  - Absorption: color properties
  - Scattering
    - Diffuse
    - Specular