Models and Architectures

Matthew Evett
Dept. Computer Science
Eastern Michigan Univ.

Objectives

• Learn the basic design of a graphics system
• Introduce pipeline architecture
• Examine software components for an interactive graphics system

Image Formation Revisited

• Can we mimic the synthetic camera model to design graphics hardware/software?

• Application Programmer Interface (API)
  - Need only specify
    • Objects
    • Materials
    • Viewer
    • Lights
• But how is the API implemented?

Physical Approaches

• Ray tracing: follow rays of light from center of projection until they are absorbed by objects or go off to infinity
  - Can handle global effects
    • Multiple reflections
    • Translucent objects
  - Slow
  - Need whole data base

• Radiosity: Energy based approach
  - Very slow

Practical Approach

• Process objects one at a time in the order they are generated by the application
  - Can consider only local lighting

• Pipeline architecture

  application
  program

  • All steps can be implemented in hardware on the graphics card

The Programmer’s Interface

• Programmer sees the graphics system through an 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 (1D object)
  - Line segments (2D objects)
  - Polygons (3D objects)
  - Some curves and surfaces
    - Quadrics
    - Parametric polynomial
- All are defined through locations in space (vertices)

Example (OpenGL)

```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
  - Orientation
- Lens (focal length)
- Film size
- Orientation of film plane
  - (relative to camera dir.)

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

Following the Pipeline: Transformations

- Much of the work in the pipeline is in converting object representations from one coordinate system to another
  - World coordinates
  - Camera coordinates
  - Screen coordinates
- Every change of coordinates is equivalent to a matrix transformation
Clipping

- Just as a real camera cannot "see" the whole world, the virtual camera can only see part of the world space.
  - Objects that are not within this volume are said to be clipped out of the scene.

Illustrating Clipping

- Try running the program, `projection.exe` that came in the TUTORS directory of the book's CD.
  - Manipulate the zNear and zFar parameters to see how the clipping region affects what is displayed.

Projection

- Must carry out the process that combines the 3D viewer with the 3D objects to produce the 2D image.
  - Perspective projections: all projectors meet at the center of projection.
  - Parallel projection: projectors are parallel, center of projection is replaced by a direction of projection.

Rasterization

- If an object is visible in the image, the appropriate pixels in the frame buffer must be assigned colors.
  - Vertices assembled into objects.
  - Effects of lights and materials must be determined.
  - Polygons filled with interior colors/shades.
  - Must have also determine which objects are in front (hidden surface removal).