Computer Graphics

spring, 2013

Computer Graphics

- We’re interested in...
  - 3D computer graphics
  - Real-time rendering
  - Interactive applications

3D Modeling Tools

- What do they do?
  - input: Your imagination
  - output: a 3d scene
- Simplest tool -- provides UI to put a triangle in the space
- Lots of ways to model various shapes + Instant visual feedback
- Blender, SketchUp, Maya, 3ds Max, ...

3D Rendering

- Digital camera analogy
  - What are the differences?
- Inputs - scene description
  - (Geometric) objects with material (wood, plastic, gold,...)
  - Camera (location+orientation, lens,...)
  - Lights (location+orientation, types: sun, flashlight, fluorescent lamp,...)
- Output
  - 2D bitmap image

3D Scene

- Usually represented as a scene graph
- Hierarchical structure (logical & spatial)
- File formats: pov, 3ds, blend, dae (COLLADA), max, ...

Geometric Objects

- Properties?
  - Shapes, material, texture, ...
- How to represent their shapes?
  - (Smooth) surfaces, polygonal/triangular mesh, implicit representation, ...
- Case study: POV-Ray, Blender, OpenGL, ...
- Polygonal/triangular mesh
  - Requires special data structure
  - file formats: off, obj, x, ...

Camera

- Properties?
  - Types - “projection”
  - Position & orientation
- Case study: POV-Ray, Blender, OpenGL, ...

Lights

- Properties?
  - Types, color, ...
  - Position & orientation
- Case study: POV-Ray, Blender, OpenGL, ...
- Shading model
  - How to compute the shading?
  - OpenGL: Gouraud shading in fixed pipeline, but flexible now

Rendering APIs

- High-level (scenegraph-based)
  - APIs to build a scene graph
  - Usually renders using low-level APIs internally
  - OpenSceneGraph, OpenSG, Java 3D, Open Inventor, ...
- Low-level
  - APIs to render a scene
  - OpenGL, DirectX
  - Hardware-accelerated

Rendering Algorithms

- How to render a scene? Can we just “simulate” what happens to digital cameras?
- Major algorithms
  - Ray-casting, Ray-tracing (specular, transmissive, etc.), Radiosity (diffusive), Rasterization
- Minor algorithms
  - Scanline rendering, Micropolygon pipeline, Tile rendering

Ray-casting

- Arthur Appel (1968)
- Image-order rendering algorithm
- Non-recursive --> No reflection, refraction, etc.
- Easily handles visibility, shadows & implicit objects
- Easily parallelizable
- May miss small objects
- Case study: Wolfenstein 3D, Volume ray-casting

Ray-tracing

- Turner Whitted (1980)
- Recursive
  - Can simulate lots of effects -- reflection, refraction, scattering, etc.
  - When to stop?
  - Too slow
- Suitable for rendering transparent/shiny objects
- Similar pros/cons with ray-casting
- Usually coupled with other rendering algorithms

Radiosity

- Goral et al. (1984)
- A “shading” algorithm
- Renders diffuse indirect lighting
- Viewpoint-independent --> Can be pre-computed
- Computed using FEM (Finite Element Method)

Scanline Rendering

- Wylie et al. (1967)
- A VSD (visible surface determination) algorithm
- Used in Nintendo DS

Rasterization

- Object-based rendering
- Direct illumination (local shading) only
  --> limitations?
- Easily parallelizable
- Real-time rendering -- OpenGL, DirectX, etc.

Rasterization (cont'd)

- Input: points, lines, triangles
- Transformations: all affine
- Projections: orthographic/perspective
- Issues
  - Primitives outside view volume
  - Visibility
  - Parallelization
- Graphics pipeline