Computer Graphics 1

Chapter 7 (June 17th, 2010, 2-4pm):
Shading and rendering
The 3D rendering pipeline (our version for this class)
Local Illumination: Shading

• Local illumination:
  – light calculations are done locally without the global scene
  – no cast shadows (since those would be from other objects, hence global)
  – object shadows are OK, only depend on the surface normal

• Loop over all polygons
• For each polygon:
  – determine the pixels it occupies on the screen and their color
  – draw using e.g., Z-buffer algorithm to get occlusion right

• Each polygon only considered once
• Each pixel considered multiple times
Flat shading

- Determine one surface normal for each triangle
- Compute the color for this triangle
  - using e.g., the phong illumination model
  - using the normal, camera and light positions
- Draw the entire triangle in this color

- neighboring triangles will have different shades
- visual „crease“ between triangles

- cheapest and fastest form of shading
- can be a wanted effect, e.g. with primitives
Gouraud shading

• Determine normals for all mesh vertices
  – i.e., triangle now has 3 normals

• Compute colors at all vertices
  – using e.g., the phong illumination model
  – using the 3 normals, camera and light positions

• Interpolate between these colors along the edges
  – neighboring triangles will have smooth transitions
  – if normals at a vertex are the same for all triangles using it

• simplest form of smooth shading
  – specular highlights only if they fall on a vertex by chance
Phong Shading

• Determine normals for all mesh vertices
• Interpolate between these normals along the edges
• Compute colors at all vertices
  – using e.g., the phong illumination model
  – using the interpolated normal, camera and light positions

• neighboring triangles will have smooth transitions
  – if normals at a vertex are the same for all triangles using it

• has widely substituted gouraud shading
  – specular highlights in arbitrary positions
  – have to compute phong illumination model for every pixel
Scan-line algorithms

- Sort all polygons by their top Y position in the image
- Scan through the image line by line
  - load polygons which start there into memory
  - discard polygons that have ended on the scan line before
- Draw all polygons currently loaded
  - sort from left to right by intersecting edges with scan line
  - using appropriate depth calculation (e.g., Z-Buffer)
  - using appropriate lighting (e.g., phong illumination model)
- Only a small number of polygons considered in each line
  - good for optimizing memory access and caching

- Each polygon only considered once
- Each pixel considered only once
Global illumination: Ray tracing

• Global illumination:
  – light calculations are done globally considering the entire scene
  – i.e. cast shadows are OK if properly calculated
  – object shadows are OK anyway

• Ray casting:
  – from the eye, cast a ray through every screen pixel
  – find the first polygon it intersects with
  – determine its color at intersection and use for the pixel
  – also solves occlusion (makes Z-Buffer unnecessary)

• Ray tracing: recursive ray casting
  – from intersection, follow reflected and refracted beams
  – up to a maximum recursion depth

http://pclab.arch.ntua.gr/03postgra/mladenstamenico/ (probably not original)
Brainstorming: what makes Ray Tracing hard?
Recent development: Real Time Ray Tracing

- Various optimizations presented over the last few years
- Real time ray tracing has become feasible
- Follow http://openrt.de/ (images from there)
Global illumination: Radiosity

- Simulation of energy flow in scene
- Can show „color bleeding“
  - blueish and reddish sides of boxes
- Naturally deals with area light sources
- Creates soft shadows
- Only uses diffuse reflection
  - does not produce specular highlights

http://www.webreference.com/3d/lesson46/
Radiosity algorithm

• Divide all surfaces into small patches
• For each patch determine its initial energy
• Loop until close to energy equilibrium
  – loop over all patches
    • determine energy exchange with every other patch
• „radiosity solution“: energy for all patches
• Recompute if ____________________________ changes


http://pclab.arch.ntua.gr/03postgra/mladenstamenico/ (probably not original)
Non-Photorealistic Rendering (NPR)

• Create graphics that look like drawings or paintings

• One method: stroke-based NPR
  – instead of grey shades, determine a stroke density and pattern
  – imitates pencil drawings or etchings (Kupferstich)

• Other methods: using image manipulation on rendered images
  – can in principle often be done in photoshop

• Active field of research
  – http://graphics.uni-konstanz.de/forschung/npr/watercolor/
  – many others

http://www.cs.ucdavis.edu/~ma/SIGGRAPH02/course23/
http://www.katrinlang.de/npr/