Visible Surface Determination

CSC 292:
Computer Graphics

Randal C. Nelson

Basic Problem

- Determine which lines or surfaces of a model are visible and display them.
- Known as Visible-line or visible-surface determination; also hidden-line or hidden surface elimination.
- Takes lots of computation.
- Two basic approaches:
  - *Image precision*: Find closest object for each pixel and color pixel appropriately.
  - *Object precision*: Determine unobstructed pieces of each object and draw them.

Functions of Two Variables

- Simplified situation: $y = f(x, z)$
- Want to plot wire curves.
- For single set, scan planes of constant z in front to back order.
- Keep track of visible silhouette (two values for each projected x value on screen).
- Draw only potions of new curves that lie outside of the silhouette.
- To do wireframe, interlace drawing curves of constant z with constant x "spurs" that connect to next constant z curve.
- Can have aliasing problems.

Efficiency

- In many situations we need to perform intersection calculations; objects with objects, rays with objects etc.
- Important to make as efficient as possible.
- Various basic techiques can be exploited.
  - *Coherence*: (objects, faces, edges, scan-lines, areas, depths, frames)
  - *Transform* of perspective view to orthographic (normalized view volume)
  - *Bounding volumes* (boxes, spheres etc)
  - *Back-face culling*
  - *Spatial partitioning*
  - *Structure hierarchies*

Robert's Algorithm

1. Cull back faces.
2. Find polyhedra that might obscure remaining edges.
3. Eliminate most via extent testing.
4. Compute visual intersection (if any) for remaining polyhedra.

- Earliest example of hidden line removal.
- Works for convex polyhedra.
- Object precision algorithm.

Appel's Algorithm

1. Define *contour lines* as potential visibility boundaries (edges between front-facing and back facing polygons).
2. Find intersections between line segments with contour lines, and keep track of *quantitative invisibility*
3. Draw line only when invisibility becomes 0.

- Works for general sets of polygons.
- Object precision algorithm.

The z-buffer Algorithm

1. Keep an extra frame buffer the *z-buffer* containing the minimum depth seen so far at each pixel as well as color.
2. Scan convert all the objects in arbitrary order.
3. Write pixel to frame buffer only if depth is less than current value of z-buffer.
4. At end, display frame buffer.

- Not limited to polygons.
- Can make use of other visibility information if available.
- Image precision; can have aliasing problems.

List Priority Algorithms

Depth Sort Algorithm

1. Sort polygons by farthest z-coordinate
2. Split polygons to resolve any overlap ambiguities
3. Scan convert each polygon from back to front

Binary Space Partitioning

1. Split space recursively into halves that partition the polygons.
2. From any viewpoint, all the polygons in one half-space can be displayed before any in the other.
3. Example of viewpoint-independent preprocessing.

Scan Line Algorithms

- Modified version of scan-line algorithms for simple polygons in 2-D.
- Keep track of which polygon we are in as we progress across the scan line.
- Sorted tables of edges for each scan line are utilized to make process efficient.
- Some cleverness is needed to handle interpenetrating polygons.

Area Subdivision

- Recursively divide screen into regions, stopping if some easy-to-decide criterion holds.
- Warnock's Algorithm - use equal area (quadtree-like) subdivisions
- Weiler-Atherton Algorithm - use polygonal subdivisions
- Subpixel division useful for anti-aliasing

Visible Surface Ray Tracing

- Throw a ray from the view center through each pixel, and see what it hits.
- Color the pixel acording to the nearest intersected object.
- Basic issue is determining intersection of object with ray.
- Easy for planes, spheres, more difficult for other objects.
- Efficiency:
  - Optimize intersection calculations.
  - Organize bounding volumes in hierarchies.
  - Or use spatial partioning.
- Antialiasing can be done by casting (adaptively) several rays per pixel.