More Visible Surface Detection

CS116B

Chris Pollett

Mar. 16, 2005.

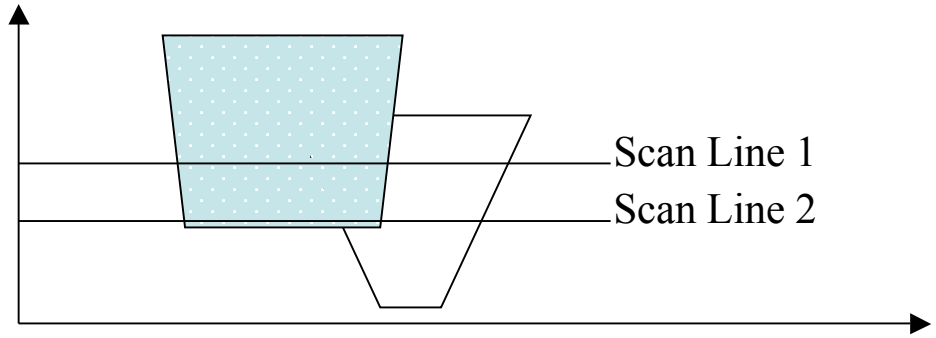
Outline

- The A-Buffer Method
- The Scan-Line Method
- The Depth Sorting Method
- BSP Trees, Area Subdivision, and Octrees
- Wire-frame Visibility Methods
- OpenGL Visibility Detection Functions

The A-Buffer Method

- Extends z-buffering, so get anti-aliasing and can handle non-opaque surfaces.
- Developed at Lucasfilms.
- ‘A’ is for accumulation.
- Each position in the buffer can now reference a linked-list of surfaces. This allows blending for transparency and anti-aliasing.
- Each position in the A-buffer store a real number for depth type and a pointer to surface info.
- A positive value indicates only one surface contributes, a negative value means to expect a linked list.
- Surface info includes RGB values, opacity, depth, percent of pixel covered, surface identifier, etc.

The Scan-Line Method



- Scan line by line through image.
- Use polygon tables, figure out intersection of scan with edges of polygons.
- Have flag that keeps track of which polygons are relevant. At a left hand intersection this flag is turned on and at the right hand side it is turned off.
- Compute depth values if on, to figure out what color to draw that portion of line.
- Can work out next scan line incrementally from the current scan line
- Cyclic overlaps can be handled by subdividing the relevant polygons.

Depth Sorting

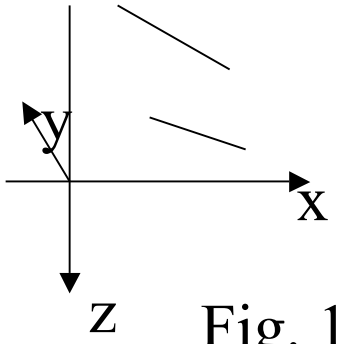


Fig. 1

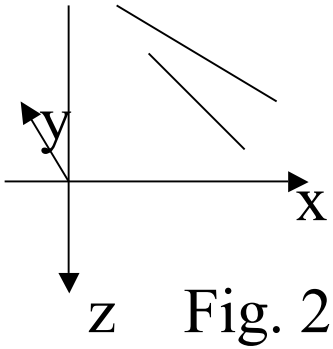


Fig. 2

- The depth sorting method:
  - sorts surfaces based on depth
  - surfaces are scan converted in order starting with the surface of greatest depth.
- This is sometimes called the *painter's algorithm*.
- Cycles through surfaces S one by one.
- If the depths of S and some other surface S' do not overlap (Fig 1), S's position in order is left unchanged.
- If overlaps occur, then check (a) bounding rectangles of S and S' in xy plane - if no overlap leave unchanged. (b) if complete overlap make nearest object later in list (c) if edge projections overlap might reorder which to draw first. -- The algorithm is not perfect (might get loops, so might subdivide).

BSP Tree, Area Subdivision, and Octrees

- To use BSP trees for objects visibility:
  - Insert objects into BSP tree according to planes, P1, P2, etc. Each plane has a front and a back and in tree this corresponds to a left or right pointer.
  - When we draw, we traverse the tree going down left edges first then right edge then parent. We render surfaces at leaves when get to them.
- Similar, idea works with octrees.
- Area subdivision is a similar technique applied to the view-plane area. Essentially split view plane recursively into quadrants until each quadrant is either a single surface, has no surface, or is one pixel in size. Then render quadrants.

Wireframe Visibility Methods

- Procedures for determining object edges are referred to as *wire-frame visibility methods, visible line methods*, or *hidden-line detection methods*.
- Two common techniques are wire-frame surface visibility and wireframe depth-cueing.
- In the first we compare endpoints of each edge with surfaces in scene. If both endpoints are in front or behind a surface, then easy. Otherwise, need to calculate point of intersection with surface to determine what to render.
- In the second technique, we adjust the color of line according to the function: $f_{depth}(d) =(d_{max}-d)/(d_{max}-d_{min})$.

OpenGL Visibility Detection Functions

- To eliminate back faces, we use the OpenGL functions:

  glEnable(GL_CULL_FACE);

  glCullFace(mode);

- Here can be GL_BACK,GL_FRONT, GL_FRONT_AND _BACK. GL_BACK eliminates back faces.
- To end this culling use:

  glDisable(GL_CULL_FACE);

OpenGL Depth-Buffer Functions

- To use depth-buffer visibility-detection, we need to first set it up:

```
glutInitDisplayMode(GLUT_SINGLE |GLUT_RGB|GLUT_DEPTH);
glClear(GL_DEPTH_BUFFER_BIT);
glEnable(GL_DEPTH_TEST);
// do stuff
glDisable(GL_DEPTH_TEST);
```

- To set up OpenGL to use wireframes can use
  glPolygon(GL_FRONT_AND_BACK, GL_LINE);
- To use depth cueing can do:

```
glEnable(GL_FOG);
glFogi(GL_FOG_MODE, GL_LINEAR);
//do stuff
glDisable(GL_FOG);
```