

View Volumes and Clipping

- Not display entire scene?
 - Objects behind observer
 - Objects too close to recognize
 - Objects too far to be worth viewing
- Must clip in 3-D
 - Against polyhedron faces?
 - May be done in hardware

1

Visible Surface Detection

- Can't see all sides of objects
 - Only want to draw the visible ones
- Object-space methods
 - Comparisons in scene coordinates
- Image-space methods
 - Comparisons at projection plane

2

Back Faces

These object faces are not visible.

These object faces (polygons) are visible.

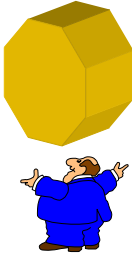
3

Back Face Detection (1)

Plane equations:

$$Ax + By + Cz + D < 0$$

if "inside" surface



If the observer's point is "inside", then the "outside" must be on the opposite side, so this is a back face and is not visible.

4

Back Face Detection (2)

View direction vector V

Surface normal vectors

$N_1 = (A_1, B_1, C_1)$

$N_2 = (A_2, B_2, C_2)$

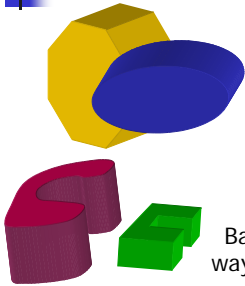
$$V \cdot N = V_z C$$

Let $V_z = -1$. If $C > 0$, vectors are in opposite directions, so face is visible. Otherwise, it is a back face or "edge on".

General rule: visible if $V \cdot N < 0$

5

Is Back Face Detection Enough?



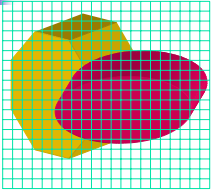
One object may obscure front faces of another

Other parts of complex objects may hide some front faces (fully or partially)

Back face detection is a quick way to eliminate some polygons from further processing

6

Depth-Buffer Method



Also known as "z-buffer".

Screen buffer: one color value per pixel

Depth buffer: one z-coordinate value per pixel

Set all depth-buffer values to maximum depth.
 Set all screen-buffer values to background color.
 Process each polygon pixel by pixel; if closer than current depth, update depth and store color.

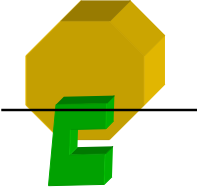
7

A-Buffer Method

- Similar to depth buffer
- Multiple polygons at a pixel?
 - Depth buffer stores only "closest"
 - Consider effect of transparent overlapping polygons?
- Maintain list of components at each pixel & combine

8

Scan-Line Method



Similar to polygon filling; use data from polygon edge table.

Keep track of all polygons the scan point is inside of.

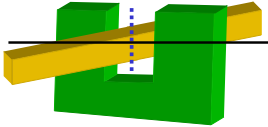
Store color of "closest" polygon at each point in scan line.

9

Scan-Line Method Problem

Assumes polygons are ordered in depth; $poly_1$ is either in front of or behind $poly_2$.

What about exceptions?



Solution: break polygons into parts at point where depths "cross".

10

Depth-Sorting Method

- So far, one pass through screen buffer
- What if more than one?
 - Draw all polygons fully, farthest first
 - Closer objects "draw over" others
- Polygons sorted by depth

11

Depth-Sort Rules

- Order S's by greatest depth
- If no depth overlap, OK
- If no x-y bound overlap, OK
- If S completely behind, OK
- If S' completely in front, OK
- If no x-y edge overlap, OK

12

Bounding-Box Overlap

If no x-y bound overlap, no need to reorder polygons.

Later, check polygon edge intersections for more precise answer.

13

Depth Comparison

Test "inside/outside" using plane equation and all vertices of other polygon.

14

Depth-Sort Algorithm

- Process S's by depth
 - If a test succeeds, S order OK
 - Draw polygon and proceed
 - If no success, swap S & S'
 - Repeat testing
- Watch out for loop
 - May need to split polygon into parts

15
