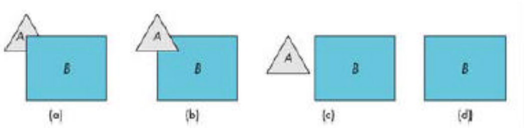
Hidden surface removal

Hidden and Visible Surfaces

- Visible surfaces
  - Parts of a scene visible from chosen viewpoint
- Hidden surfaces
  - Parts of a scene not visible from chosen viewpoint
- Subtle difference between hidden surface removal and visible surface determination
- Many algorithms: image space, object space and hybrid.
- Requirements for quality vs. speed.

Hidden Surface Algorithms

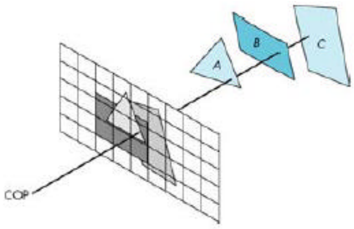
Object-space



- Comparison within real 3D scene
- Works best for scenes that contain few polygons

Image-space

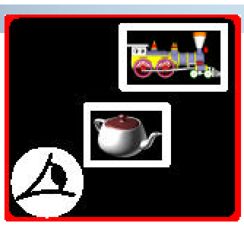
- Decide on visibility at each pixel position

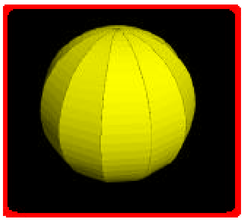


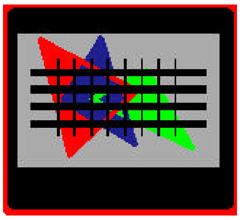
Visible surface determination

- Image Space Algorithms
  - Complexity = O(pixels * objects)
  - e.g. 1286*1024 pixels and 1 million polygons
  - Complexity = $O(1.3 * 10^{12})$
- Object Space Algorithms
  - Worst case might have to compare n objects with n-1 objects:
  - Complexity = $O(n^2)$
  - e.g. 1 million polygons
  - Complexity = $O(10^{12})$

Reducing complexity



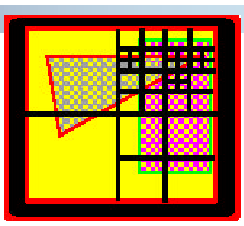


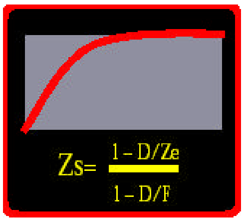


Exploit Coherence

- **Object Coherence:** If objects are well separated, compare objects not faces.
- **Face Coherence:** If faces vary smoothly, can modify face incrementally. Can put constraints on models, such as no interpenetration.
- **Edge Coherence:** An edge changes visibility only when it crosses another edge or face.
- **Scan-line Coherence:** Set of visible object spans typically does not vary much between scan lines.

Reducing complexity



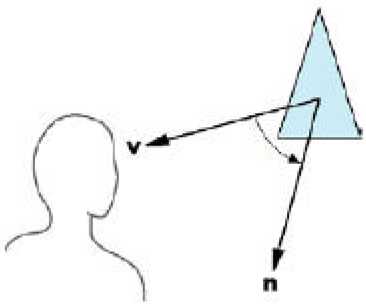


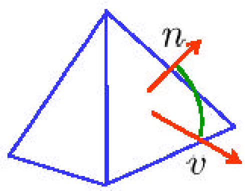
- **Area Coherence:** Groups of adjacent pixels often covered by the same visible face.
- **Depth Coherence:** Adjacent parts of the same surface are typically close in depth.
- **Frame Coherence:** In animation, frames adjacent in time are likely to be very similar.

Back-Face Removal

back-face culling

- We see a polygon if its normal is pointed toward the viewer.





Plane equations

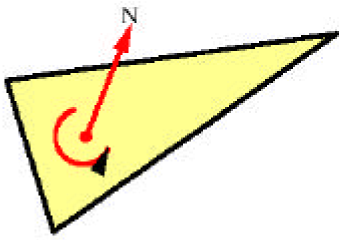
Finding Polygon Normals
Ax + By +Cz + D = 0

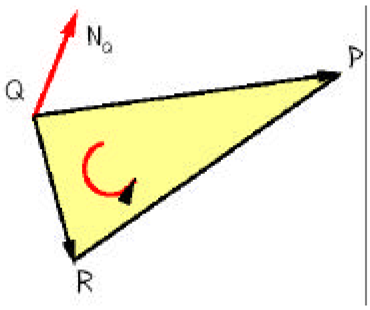
Normal Vector: n=(A, B, C)

n = (Q-R) X (Q-P)

- In many cases the vertex normals have been calculated by a polygoniser

- Normals and cross products must be normalised





Plane Equation

Plane equation : Ax + By +Cz + D = 0

- Can store plane as a point and a normal vector or store A,B,C,D. To determine coefficients given 3 non-colinear points:

  (A/D)xi + (B/D)yi + (C/D)zi = -1

Using cramer's rule:

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

Expanding:

A = y1(z2-z3)+y2(z3-z1)+y2(z3-z1)

etc.

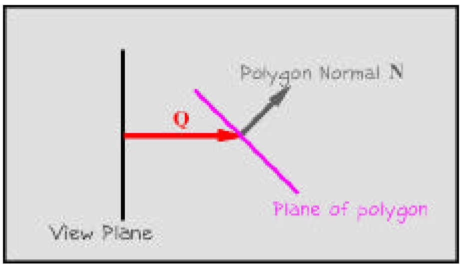
Back face culling

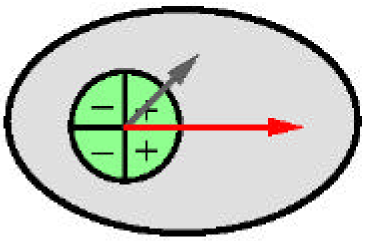
Many renderers consider counter-clockwise polygons as outwards facing.

Take dot product with View Plane Normal

If (+)ve or zero discard polygon.

Q = Normalized View Plane Normal





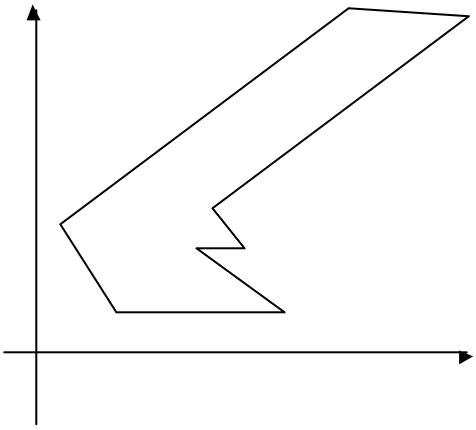
dotp = **N.Q**

if dotp (+)ve Backfacing
if dotp (-)ve Frontfacing
if dotp = 0 Edge On

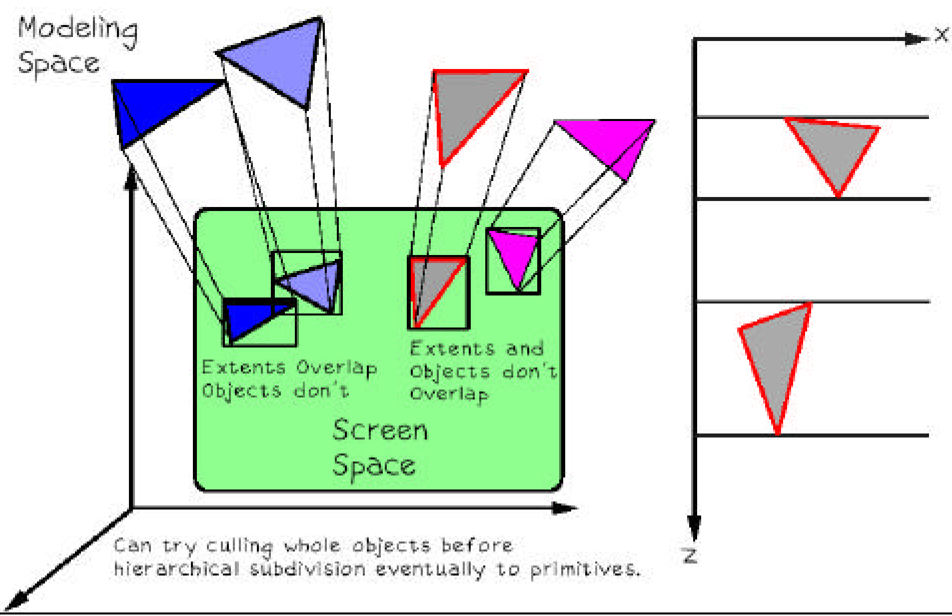
Assuming perspective projection. Polygon is backfacing if Z component of normal is (-)ve in the eye system.

Back face culling

- object is (approximated by) a solid polyhedron => faces completely enclose its volume



Culling by comparing extents



Depth sort

Or painters algorithm

- determine a visibility ordering for objects which will ensure a correct picture if objects are rendered in that order
- If no objects overlap in depth (z), then it is only necessary to sort them by increasing z (furthest to closest) and render them
- Otherwise, it will be necessary to modify (by splitting) the objects to get an ordering

Depth sort

tests for visibility:

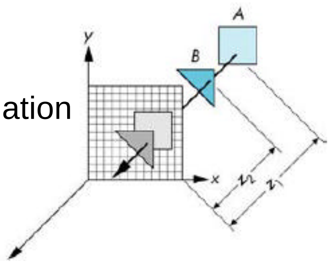
1. do x or y -extents not overlap?
2. is S entirely on the opposite side of S' plane, from the viewpoint?
3. is S entirely on the same side of S' plane, from the viewpoint?
4. do the projections of polygons onto the xy plane not overlap?

Z-buffer method

- A commonly used image-space approach to hidden-surface removal
- It is also referred as Depth-Buffer method
- Use the intensity color of the nearest 3D point for each pixel

What is an efficient way for it?

- 2 buffers,
  - frame buffer – stores image information
  - z-buffer – depth information
  - with the same resolution

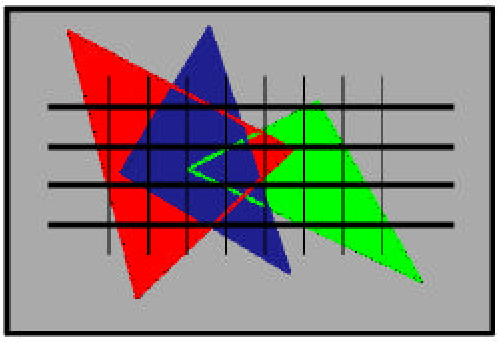


Scan Line Algorithms

E.g. Z buffer, A buffer etc.

Z-Buffer

1. Store Background Colour in buffer.
2. Scan Convert Each Polygon
3. At each Pixel determine if Z value (in eye system) is nearer (lower z) than current stored z value. If it is lower than swap current colour for stored colour.



Z-Buffer Algorithm

```
for all positions (x,y) in the view screen
   frame(x,y)=I_background
   depth(x,y)=max_distance
end
for each polygon in the mesh
   for each point(x,y) in the polygon-fill algorithm
        compute, z, the distance of corresponding 3D-point from COP
        if depth(x,y) > z // a closer point
                depth(x,y)=z
                frame(x,y)=I(p) //shading
        endif
   endfor
endfor
```

Determining Z-Depth

If we have the plane equation:

$Ax + By + Cz + D = 0$ Normal Vector: N=(A, B, C)

Insert known x,y into plane eqn. and solve for z:

$z = (-ax - by - d)/c$

Then at $(x_1 + Dx, y_1)$

$z' = z_1 - aDx/c$ a/c is constant for the plane $Dx = 1$

So incrementing:

$z_i+1 = z_i$ - a/c across scan line

$z_j+1 = z_{i\,j}$ - b/c between scanlines

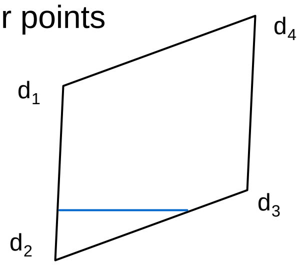
Z-buffer Alternative method

After the algorithm

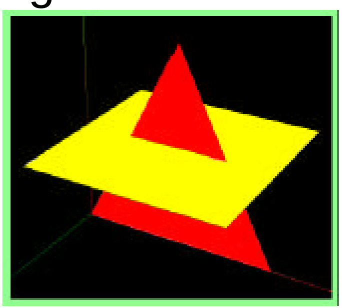
- Frame buffer contains intensity values of the visible surface
- z-buffer contains depth values for all visible points

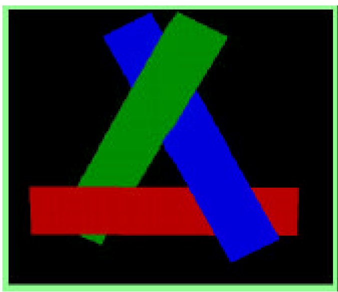
Alternative method for computing z-depth the step in algorithm

- We know d_1, d_2, d_3 and d_4 from vertices of the mesh
- Use linear interpolation for other points



Advantages & Disadvantages of Z buffer





- Needs large memory to keep Z values
- Can be implemented in hardware
- Can do any number of primitives
- Handles cyclic and penetrating polygons
- Handles Polygon Stream in any order
- Transparency?