

Visible Surface Determination

- Object *versus* Image-space Algorithms
- List Priority
- Space Partitioning
- Visible Surface Ray Tracing
- Depth Buffering
 - Z-buffering
 - W-buffering

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Algorithms Classification

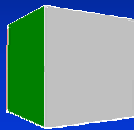
- **Object Space:** order in which the primitives are rendered is decided in object space
 - List priority (Painter, Occlusion-compatible order)
 - Binary space-partitioning
- **Image Space:** visibility is decided on a per-pixel basis
 - Visible surface ray tracing
 - Depth buffering

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

List Priority

- Painter's Algorithm
 - Primitives are sorted by their z coordinates in camera space and rendered in back to front order

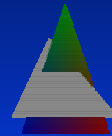


CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Painter's Algorithm (Cont.)

- Interpenetrating polygons need to be split before they can be rendered.



CSE528 Fall 2000

Copyright © Manuel M. Oliveira

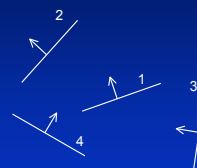
Binary Space-Partitioning [Fuchs 80]

- Creates a binary tree of polygons (BSP tree)
- At each node, its left sub-tree contains the polygons in front of (behind) the plane of the node polygon, while its right sub-tree contains the polygons behind (in front of) such plane.
- Efficient data structures for static scenes
- Interpenetrating polygons need to be split

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

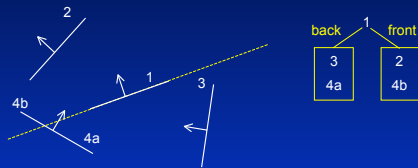
BSP Tree - Example



CSE528 Fall 2000

Copyright © Manuel M. Oliveira

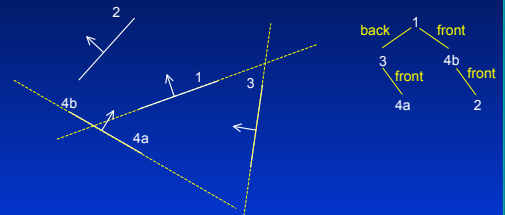
BSP Tree - Example



CSE528 Fall 2000

Copyright © Manuel M. Oliveira

BSP Tree - Example

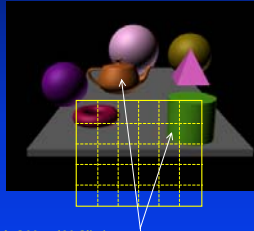


CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Visible Surface Ray Tracing

- Trace rays through the centers of each pixel of the image plane and find the closest intersection with the objects in the scene



CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Depth Buffering

- Keeps track of the current depth associated with each pixel
- Color and depth values are interpolated and updated during rasterization
- Depth-buffering algorithms
 - Z-buffer
 - W-buffer

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Z-Buffer

- According to the Projection Matrix, the expression for z after perspective division is given by

$$P' = \begin{bmatrix} \frac{2n}{(r-l)} & 0 & \frac{(r+l)}{(r-l)} & 0 \\ 0 & \frac{2n}{(t-b)} & \frac{(t+b)}{(t-b)} & 0 \\ 0 & 0 & -\frac{(f+n)}{(f-n)} & -\frac{2fn}{(f-n)} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_{ccs} \\ y_{ccs} \\ z_{ccs} \\ 1 \end{bmatrix}$$

$$z_{scs} = \frac{(f+n)}{(f-n)} + \left(\frac{1}{z_{ccs}} \right) \left(\frac{2fn}{f-n} \right) \quad \text{with} \quad -1 \leq z_{scs} \leq 1$$

inside the view volume

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Z-Buffer (Cont.)

- The resolution of a Z-buffer depends on the ratio Z_n/Z_f
- For lower values of this ratio, the transformed z values concentrate around 1, introducing visibility artifacts across the entire view volume.

$$\lim_{n \rightarrow 0} \left(\frac{f+n}{f-n} \right) + \left(\frac{1}{z_{ccs}} \right) \left(\frac{2fn}{f-n} \right) = 1$$

- Low ratio values require a larger number of bits in the z representation in order to solve visibility appropriately.

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Z-Buffer (Cont.)

- Object rendered using OpenGL with different values Z_n and Z_f



Znear = 0.1
Zfar = 4000
ratio: 2.5×10^{-5}



Znear = 0.5
Zfar = 4000
ratio: 12.5×10^{-5}



Znear = 0.1
Zfar = 2500
ratio: 4×10^{-5}

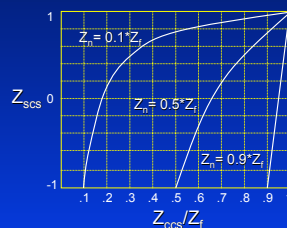


Znear = 0.5
Zfar = 2500
ratio: 2×10^{-4}

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Z-Buffer Resolution [Blinn 98]



CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Z-Buffer (Cont.)

- In order to simplify depth comparison, z-buffer hardware usually encodes the computed z values using integers
- Map the visible interval $[Z_n, Z_f]$ to $[0, 1]$ and quantize and store depth as integer i , so that $z \approx 1/i$
- Bigger integer values represent depth close to Z_n
- The fewer bits available for integer representation, the lower the z-buffer resolution, specially near Z_n
- Function $f(x) = 1/x$ is not linear and, therefore, quantization bins are not equally spaced

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

W-Buffer

- $W = -Z_{ccs}$ and can be used for solving visibility

$$V = \begin{bmatrix} \frac{2n}{(r-l)} & 0 & \frac{(r+l)}{(r-l)} & 0 \\ 0 & \frac{2n}{(t-b)} & \frac{(t+b)}{(t-b)} & 0 \\ 0 & 0 & -\frac{(f+n)}{(f-n)} & -\frac{2fn}{(f-n)} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_{ccs} \\ y_{ccs} \\ z_{ccs} \\ 1 \end{bmatrix}$$

- During rasterization, the w values associated with the vertices of the polygon require perspective correct interpolation

CSE528 Fall 2000

Copyright © Manuel M. Oliveira

Z-Buffer x W-Buffer

- W is defined in terms of the Z coordinate in camera/eye space and therefore its value is independent of Z_n
- W-buffer is the best choice if one needs to make Z_n very small (e.g., room walkthrough)
- Z-buffer is the best choice if Z_n can be a "noticeable fraction of Z_f " [Blinn 98] (e.g., CAD application)
- If $Z_n/Z_f > 0.5$, z-buffer is the algorithm of choice

CSE528 Fall 2000

Copyright © Manuel M. Oliveira