Graphics Lecture 8

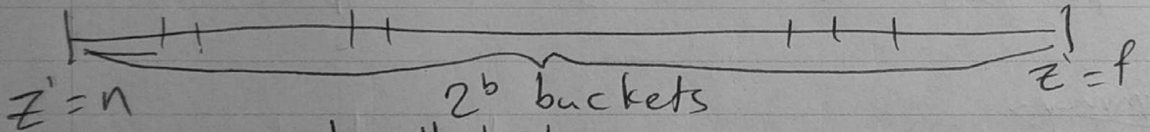
May 24th, 2019

Z-Buffer Algorithm

- Image Precision
- View dependent
- Geometry independent, don't have to load all the polygons
  - → after rendering z-buffer can be saved and used later to merge in other objects whose z can be computed.

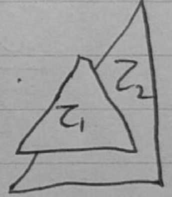
Z-Buffer Relation (Digression, not asked on exam)

- bit Plane – RGB 24 bits – Z-buffer 8, 10, 24, 32
  A – transparencies



b = # bits

$\Delta z' = \frac{f \cdot n}{2^b}$ if $|z_1 < z_2'| < \Delta z$



- z-buffer may draw triangles in the wrong order

Can use 2 z-buffers

$\Delta z_n \simeq \frac{n}{2^b}$ $\Delta z_f = \frac{f^2}{n 2^b}$

What might be good values for n and f on a 18-bit z-buffer of image that we want is the inside of a sports stadium seen by members of the crowd

Δz = function(n, f, b) objects bigger than $\Delta z_f = 0.01$m
$n = 1.5$m distance between two spectrums.

$\Delta z_n = \frac{1.5}{2^{18}} = 5.7 \times 10^{-6}$ $f = ((0.01)(1.5)(2^{18}))^{\frac{1}{2}}$
$= 62.20$m

Binary Space Partitioning Trees (BSP Trees)

Sources: Computer Graphics, Foley et al 1990

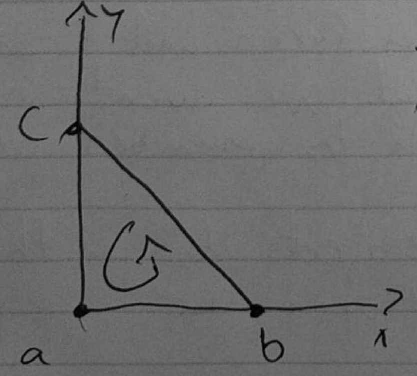
- efficient for calculating the visibility relationships among a static group of 3D polygons as seen from an arbitrary viewpoint
- tree is built recursively ⇒ displaying the polygons without obscurring the wrong ones works. No matter which one we pick as the root. Display order may be different.
- Display (tree traversal) is also performed recursively
  - if the viewer is in the root's front halfspace then the algorithm must first display polygons in its back half space then the root, then finally the polygons in the front half space.
  - otherwise, must first display polygons in the root's front half space, then the root and finally the polygons in the root's back half space.

Summary

- object precision - view independent
- geometry dependent
- what does it have in common with z-buffer?
  - display's polygons in a back to front order,

exam q → possibly occuring more distant ones later. Like z-buffer shading calculations may be performed more than once for each period.

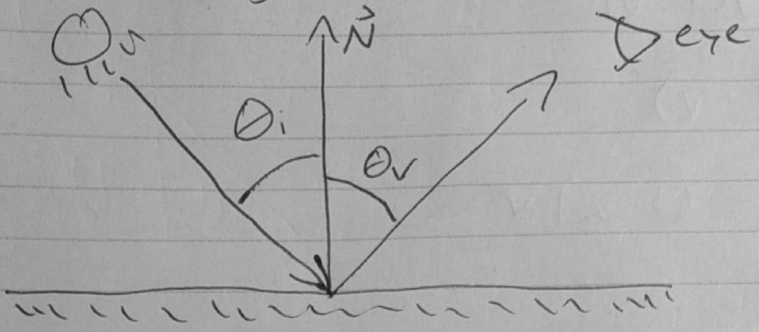
Note



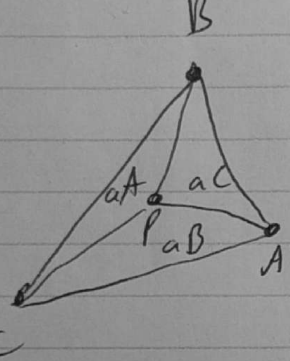
$\vec{N} = (b-a) \times (c-a) = (0,0,1)$ ⊙

$\vec{N} = (c-a) \times (b-a) = (0,0,-1)$ ⊗

Shading Introduction



Gouraud Shading



$aA = \Delta PBC / \Delta ABC = \alpha$

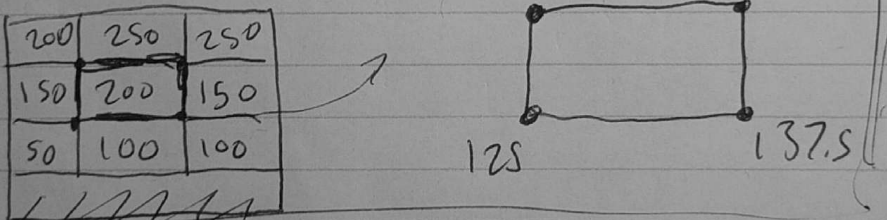
$aB = \Delta APC / \Delta ABC = \beta$

$aC = \Delta ABP / \Delta ABC = \gamma$

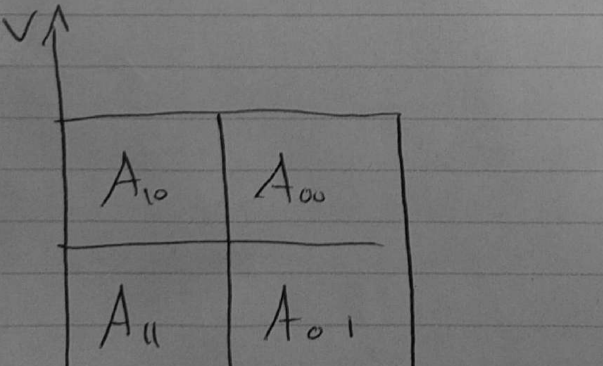
$\alpha + \beta + \gamma = 1$

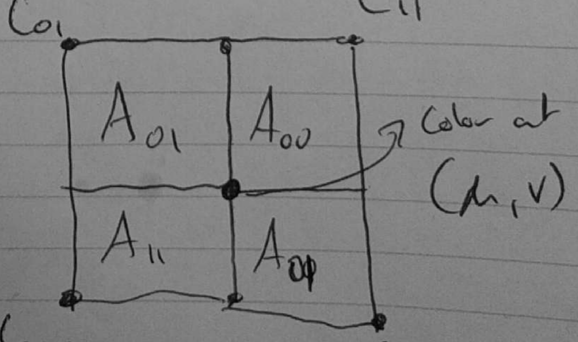
Parametric Bilinear Interpolation

Ⓐ



200	250	250
150	200	150
50	100	100





$u, v \in [0, 1]$

$A_{00} = (1-\mu)(1-v)$

$A_{10} = \mu(1-v)$

$A_{01} = \cancel{\mu(1-} (1-\mu) v$

$A_{11} = \mu v$

$\text{color}(\cancel{\mu, v}) = C_{00} A_{00} + C_{01} A_{01} + C_{10} A_{10} + C_{11} A_{11}$

μ, v

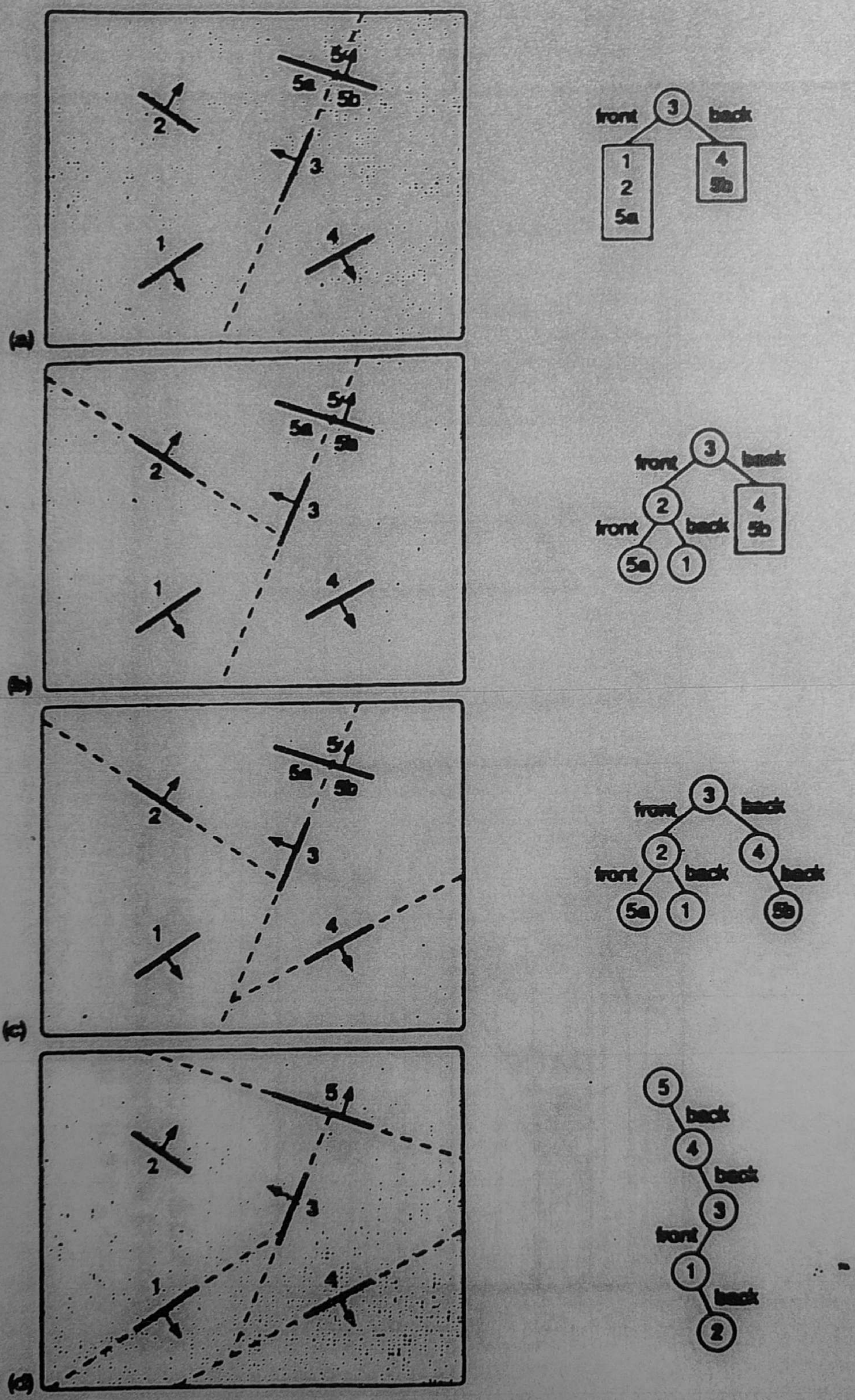


Fig. 15.31 BSP trees. (a) Top view of scene with BSP tree before recursion with polygon 3 as root. (b) After building left subtree. (c) Complete tree. (d) Alternate tree with polygon 5 as root. (Based on [FUCH83].)

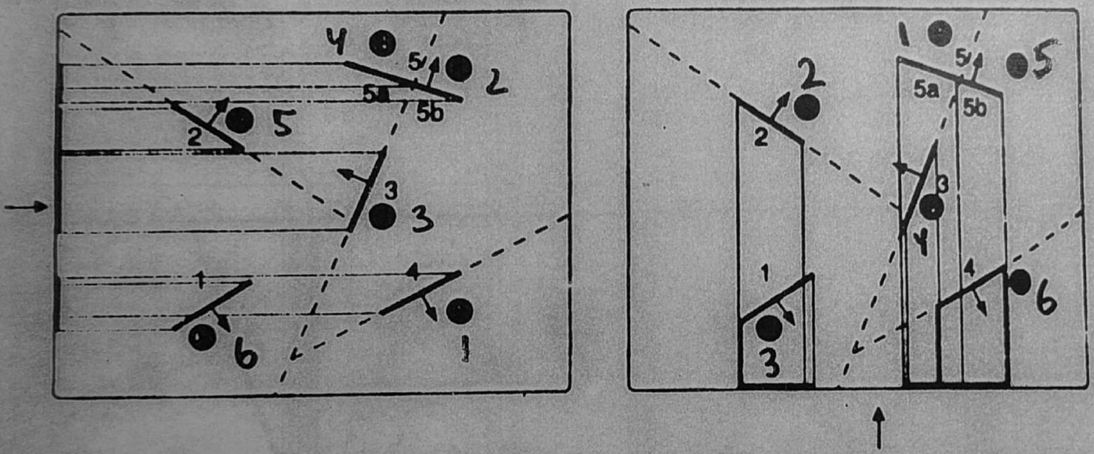


Fig. 15.33 Two traversals of the BSP tree corresponding to two different projections. Projectors are shown as thin lines. White numbers indicate drawing order.

Scanned by CamScanner