

CS6533/CS4533 Lecture 8.1

Slides/Notes

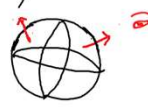
Hidden Surface Removal & BSP Trees; Shadow Projection & Making Decal in HW3 (Notes, Ch 11, Notes)

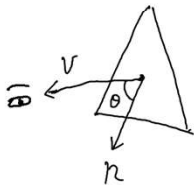
By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

1

[Below we look at occlusion culling Algorithms.]

1. Back-Face Removal:

Suppose we have closed surface of an obj (eg. sphere, cube, ...), where each polygonal face has a normal vector going outward. eg. 



polygon is facing forward iff $\theta \in [-90^\circ, 90^\circ]$

i.e. $\cos \theta \geq 0$.

i.e. $n \cdot v \geq 0$. ($n \cdot v = |n||v| \cos \theta$)

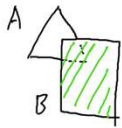
iff $n \cdot v \geq 0$.

If $(n \cdot v \geq 0)$ render the polygon; otherwise, don't.

2

2. Depth Sort and Painter's Algorithm.

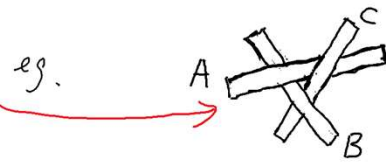
Draw polygons from the farthest to the closest (back to front)



The overlapped portions are over-written.
Only the closest portions are retained.

⇒ We need Depth Sort (sort polygons from the farthest to the closest)

Problem: We may have cyclic ordering.
i.e. Depth order may NOT exist

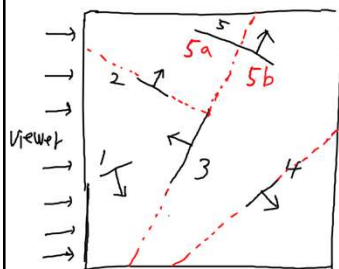


Partial solution: break into pieces. Q: How do we break?
A: space-partitioning tree such as BSP tree.

3

BSP Tree (Binary-Space Partition tree)

(Render back-to-front)

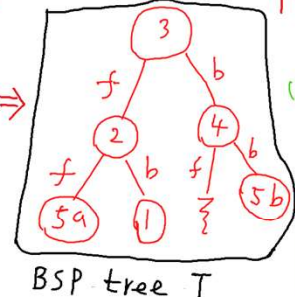
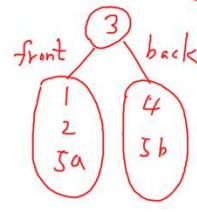


polygons 1~5 are projected onto the 2D plane as line segments.
↑ indicates the front side.

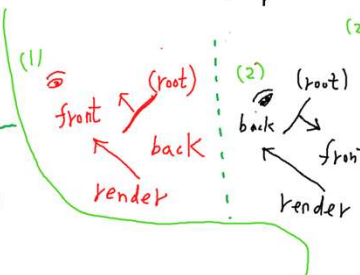
Using BSP_Render() on T:

- (A) 4 5b 3 1 2 5a
- (B) 4 5b 3 5a 2 1
- (C) 5a 2 1 3 4 5b

Which of (A), (B), (C) is correct?

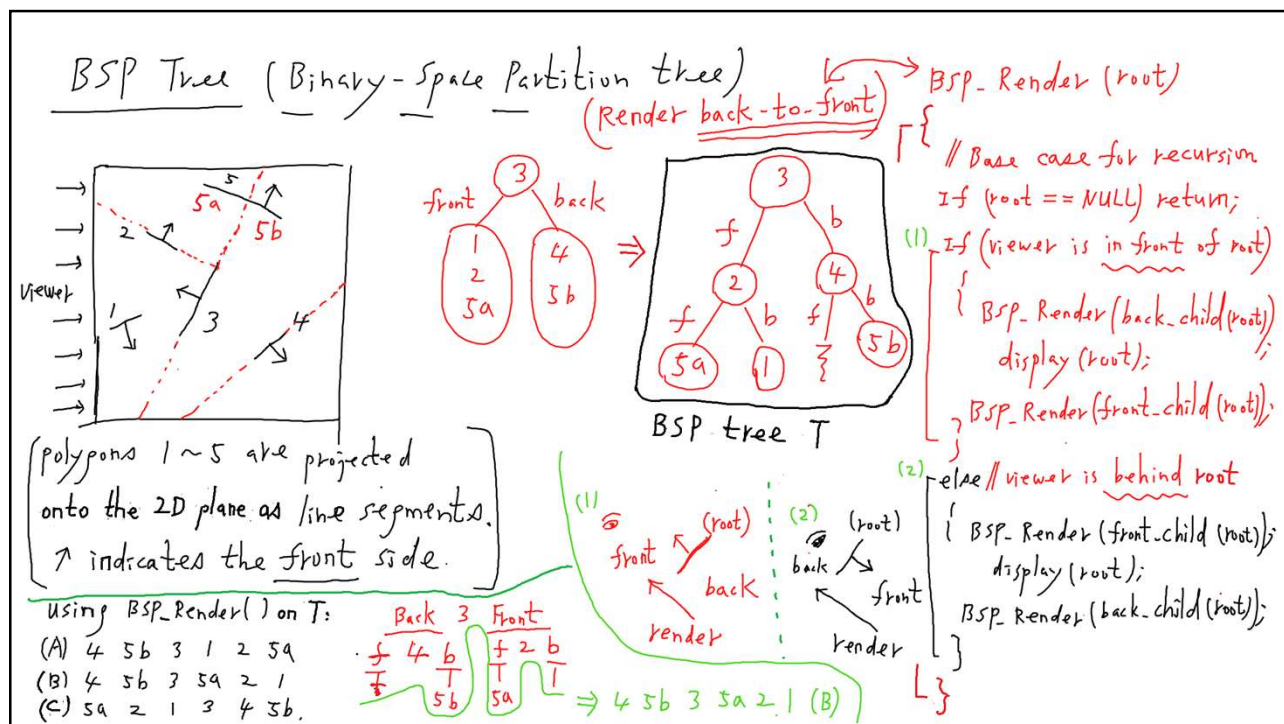


BSP tree T

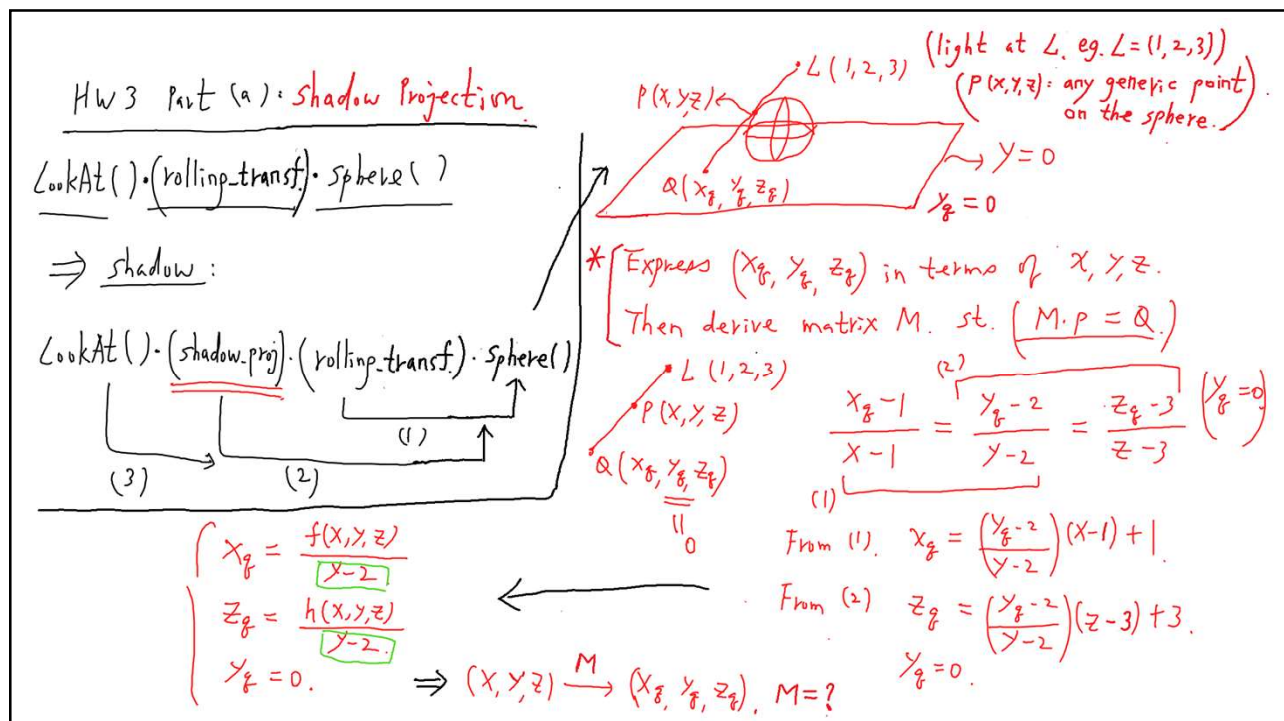


BSP_Render (root)
 {
 // Base case for recursion
 If (root == NULL) return;
 (1) If (viewer is in front of root)
 {
 BSP_Render (back_child (root));
 display (root);
 BSP_Render (front_child (root));
 }
 (2) else // viewer is behind root
 {
 BSP_Render (front_child (root));
 display (root);
 BSP_Render (back_child (root));
 }
 }

4



5



6

* Key: Use Homogeneous Coordinate System!

$$\begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f(x,y,z)}{x-z} \\ 0 \\ \frac{h(x,y,z)}{x-z} \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f(x,y,z) \\ 0 \\ h(x,y,z) \\ y-z \end{bmatrix} = \begin{bmatrix} \text{constant entries} \\ \text{independent} \\ \text{of } x, y, z \\ 4 \times 4 \\ M \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(Perspective Division (P.D.))

Recall:

$w \neq 0, w \neq 1$

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \xrightarrow{(P.D.)} \begin{bmatrix} X/W \\ Y/W \\ Z/W \\ 1 \end{bmatrix}$$

* Note: The 16 entries in the 4×4 matrix M must be constants independent of x, y, z so that M can be applied to all different points (with different x, y, z) on the sphere.

7

* HW3 Part (b): Making Decal

shadow is the decal on top of the ground

There are 2 buffers: frame buffer & z-buffer.

We can enable/disable writing to these buffers.

0. Always enable z-buffer testing

1. Disable writing to z-buffer.

Draw ground (only to frame buffer).
NOT to z-buffer.

2. Enable writing to z-buffer.

Draw shadow (to both buffers)

* (ground is NOT in z-buffer to block shadow.
so shadow is always drawn on top of ground.)

(Now, we want to put back ground to z-buffer while preserving the current content of frame buffer i.e. draw ground only to z-buffer.)

3. Disable writing to frame buffer.

Draw ground. (only to z-buffer)

4. Enable writing to frame buffer.

Resume normal ops.

⌈ : critical section.

⌋ : Any other obj's should be drawn outside ⌈

8