Lecture 17: Shadows

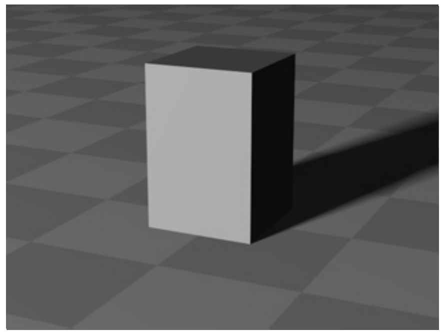
Fall 2004
Kavita Bala
Computer Science
Cornell University

Projects

- Proposals due today
- I will mail out comments
- Grading HW 1: will email comments asap

Why Shadows?

- Crucial for spatial and depth perception



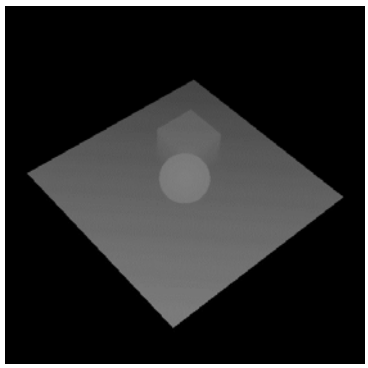
Shadows

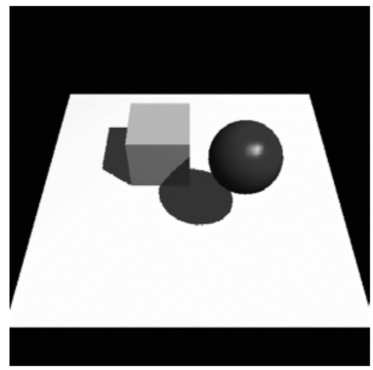
Methods for fast shadows:

- Shadow Maps
- Shadow Volumes

Shadow Maps

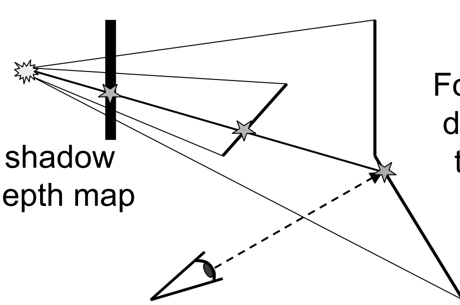
- Introduced by Lance Williams (SIGGRAPH 1978)
- Render scene from light’s view
  – black is close, white is far





Using the Shadow Map

- When scene is viewed, check viewed location in light's shadow buffer
  – If point's depth is (epsilon) greater than shadow depth, object is in shadow

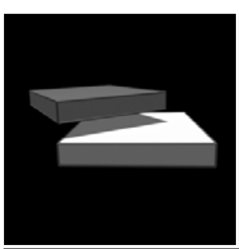


For each pixel, compare distance to light ✯ with the depth ✯ stored in the shadow map

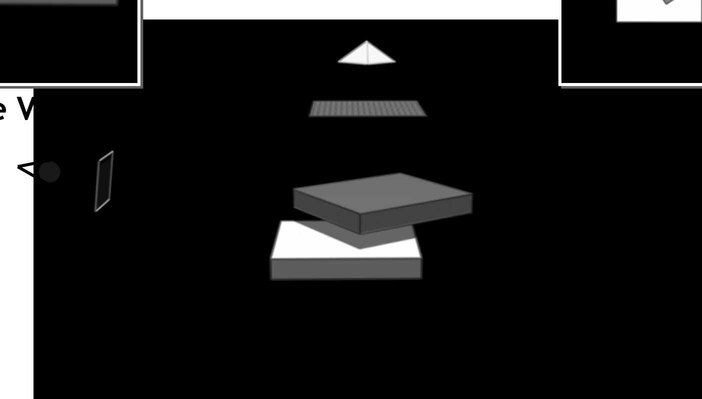
Shadow Mapping: Pass 1

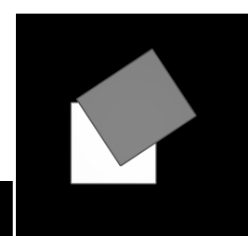
- Depth testing from light’s point-of-view
  - Two pass algorithm
- First, render depth buffer from light’s point-of-view
  - Result is a “depth map” or “shadow map”
  - A 2D function indicating the depth of the closest pixels to the light
  - This depth map is used in the second pass

How Shadow Maps Work



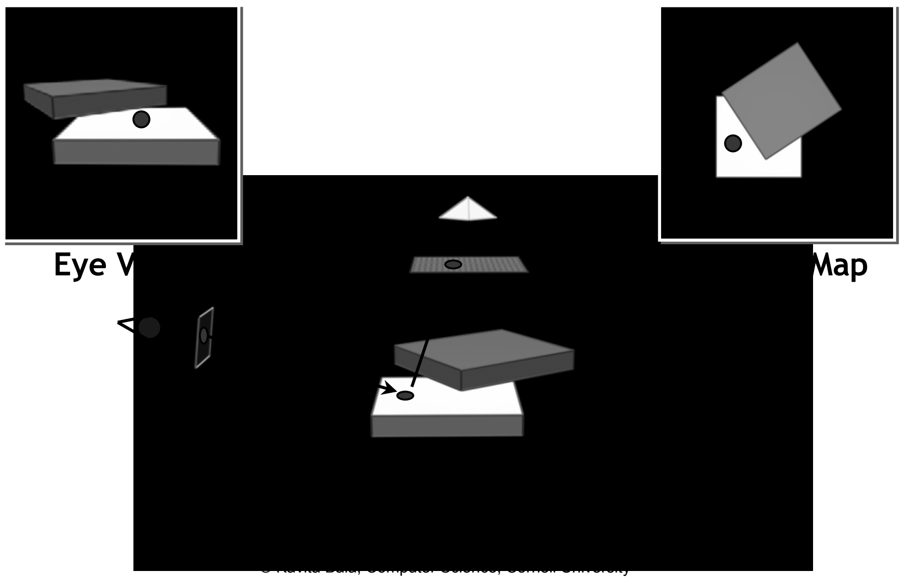
Eye V





Map

How Shadow Maps Work



Shadow Mapping: 2nd pass

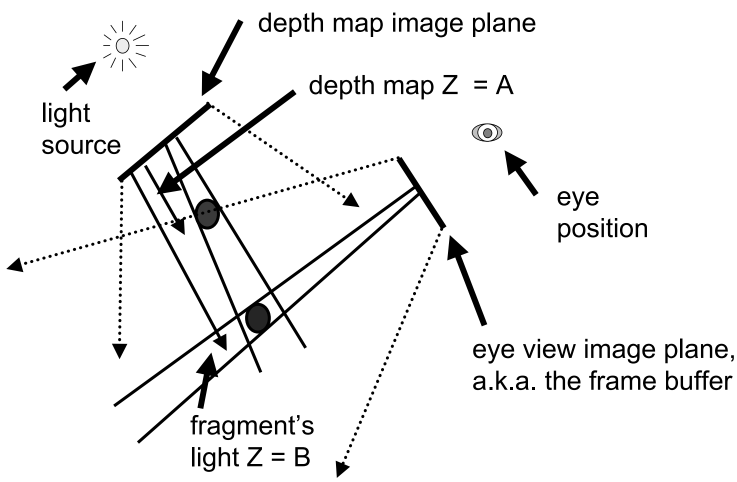
- Second, render scene from the eye's point-of-view
- For each rasterized fragment
  - determine fragment's XYZ position relative to the light
  - this light position should be setup to match the frustum used to create the depth map
  - compare the depth value at light position XY in the depth map to fragment's light position Z

Shadow Mapping: Comparison

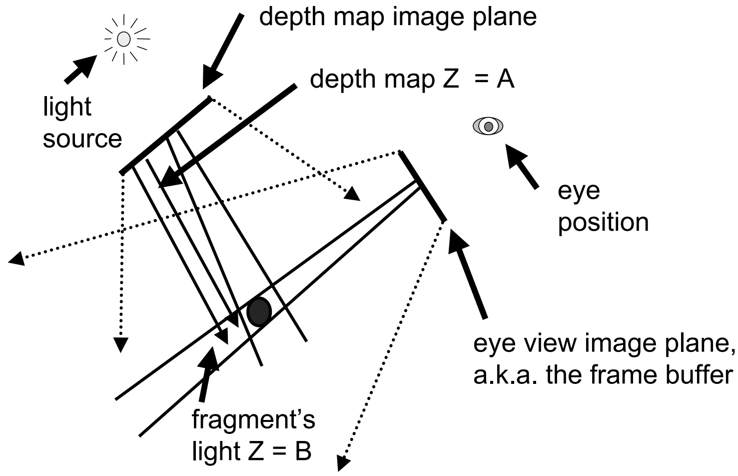
- Two values
  - A = Z value from depth map at fragment's light XY position
  - B = Z value of fragment's XYZ light position
- If (B > A),
  - There must be something closer to the light than the fragment
  - So, fragment is shadowed
- If A and B are approximately equal, the fragment is lit

Example: Shadowed

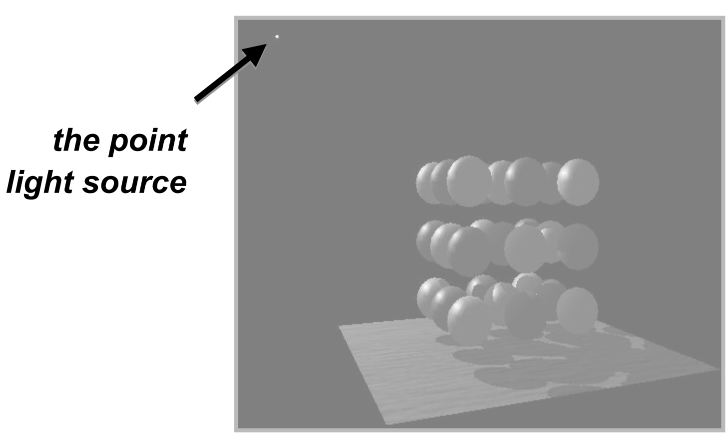
The A < B shadowed fragment case



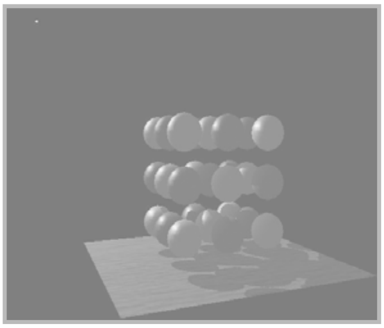
Example: Visible

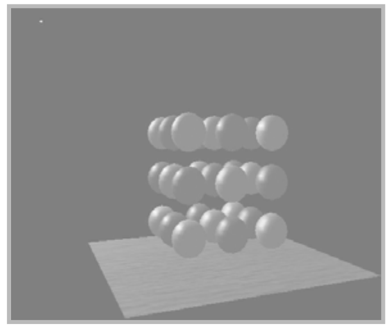


Example



Example





with shadows ***without shadows***

Shadow Map Issues

- Can only cast shadows over a frustum
  - Use 6 (like a cube map)

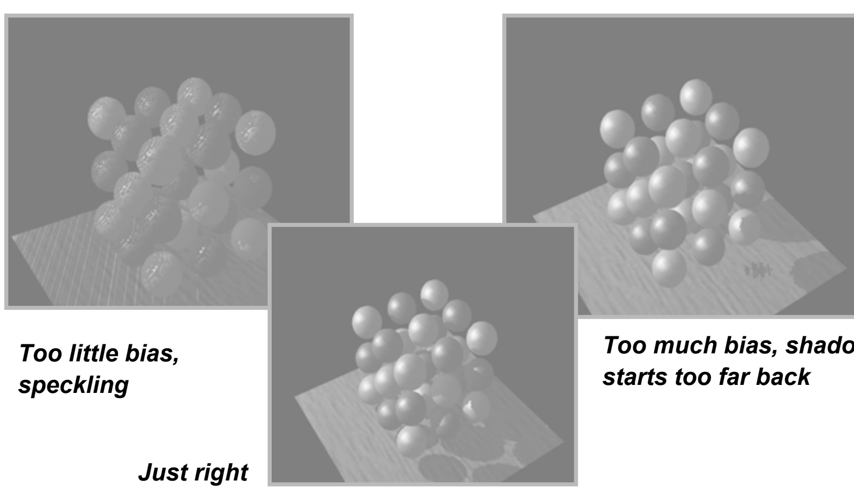
- Get speckling because of floating point errors
  - Use triangle ids
  - Use bias
    - If (B > A+bias) p in shadow

Shadow Map Issues

- Use triangle Ids
  - Meshes?
- Bias
  - If (B > A+bias) p in shadow
  - If b is large?
  - If b is small?

Bias Issues

- How much polygon offset bias depends



Too little bias, speckling

Just right

Too much bias, shadow starts too far back

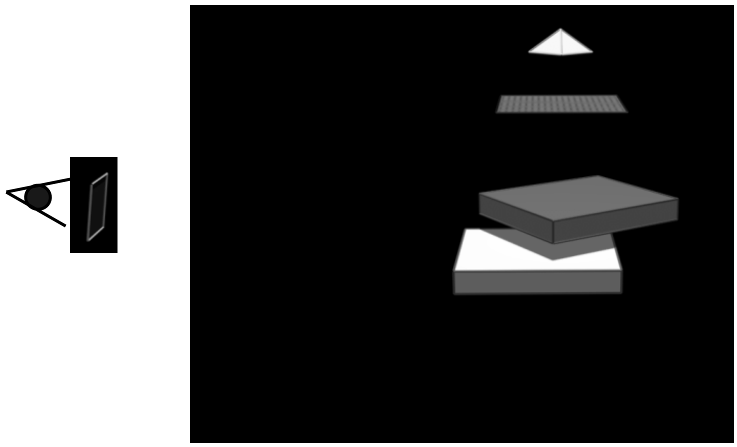
Shadow Maps on Hardware

- Shadow Maps use projective textures
- Treat texture as a light source (slide projector)
  - Do not need to specify texture coordinates explicitly
  - Spotlights

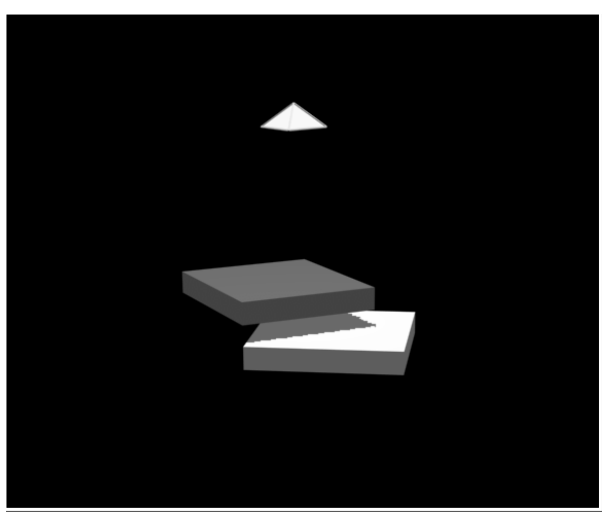
Properties of Shadow Maps

- One shadow map per light
- Render scene twice per frame
  - If static, can reuse
- Advantages
  - Fast
  - Easy to implement
- Disadvantages
  - Bias
  - Aliasing
  - Hard shadows

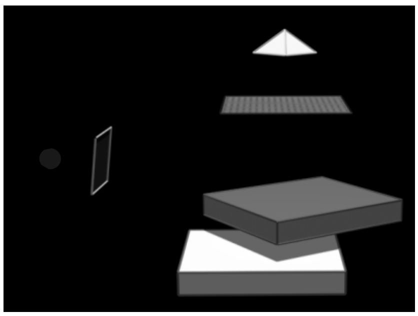
Aliasing (Distant)

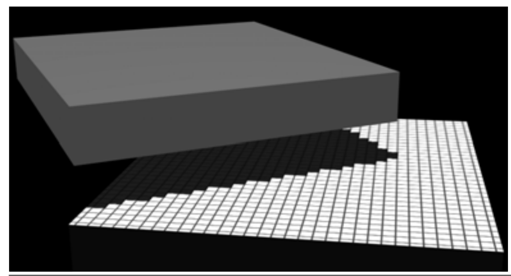


Aliasing in Eye View (Distant)

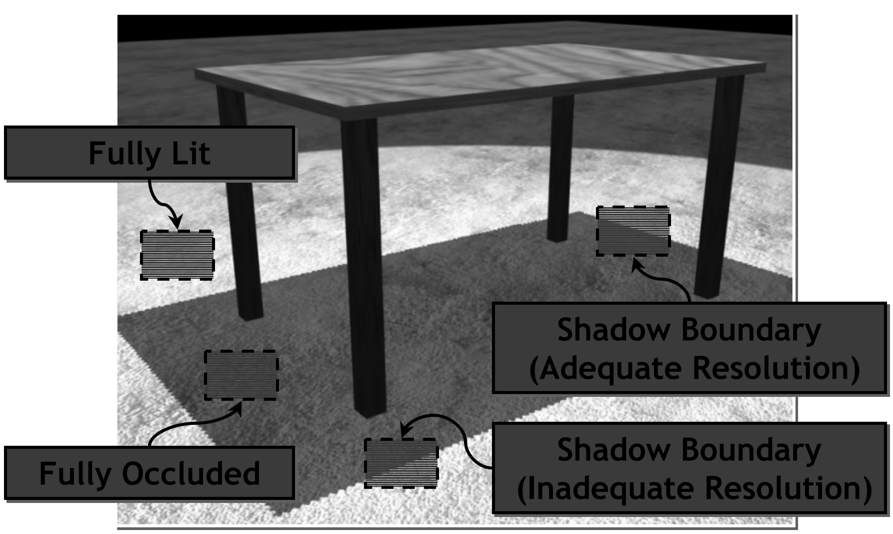


Aliasing (Close)

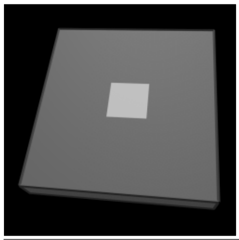




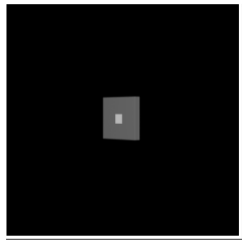
Where does aliasing occur?



Why does Aliasing arise?

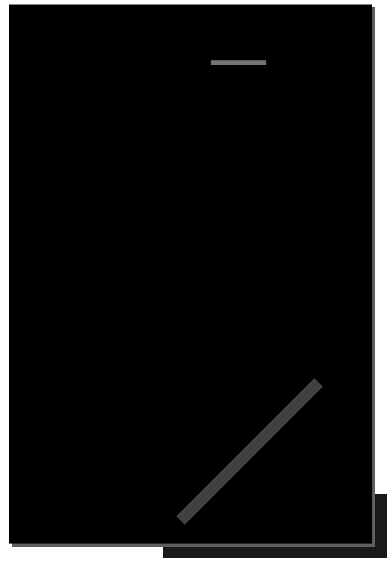


Eye View



Shadow Map

Eye View Projected Area != Shadow Map Projected Area



Shadows

Methods for fast shadows:

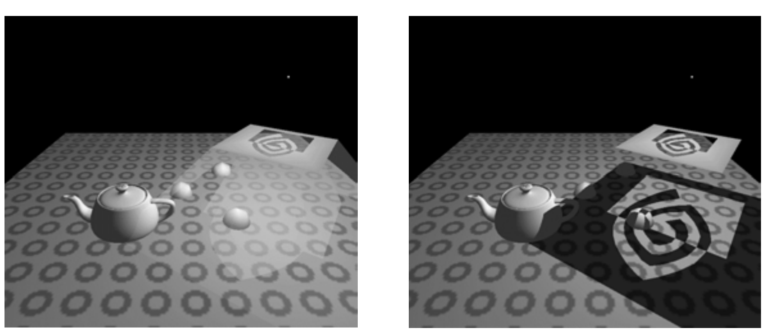
- Shadow Maps
- Shadow Volumes

Shadow Volumes

- Crow 1977
- Accurate shadows

Shadow Volumes

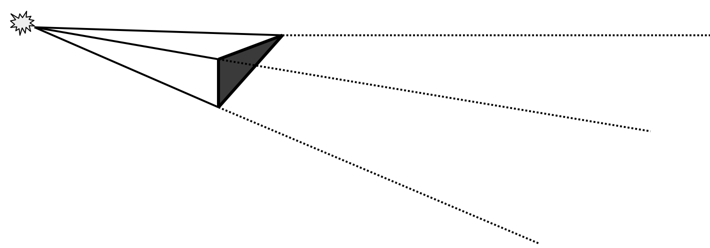
- Clever counting method using stencil buffer
- Can cast shadows onto curved surfaces



Mark Kilgard, NVIDIA Inc.

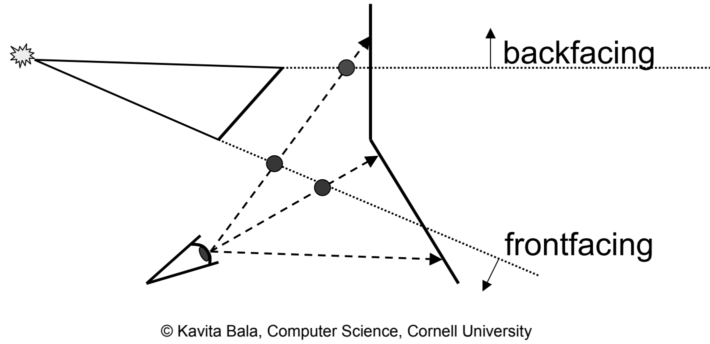
Volume Concept

- Create volumes of space in shadow from light
- Each triangle creates 3 projecting quads



Using the Volume

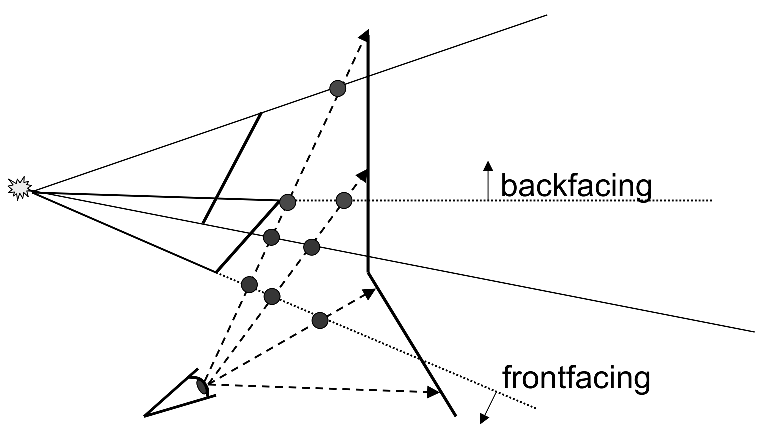
- To test a point, count the number of polygons between it and eye
- If more frontfacing than backfacing polygons, then in shadow
- Done with clever counting method using the stencil buffer



Algorithm

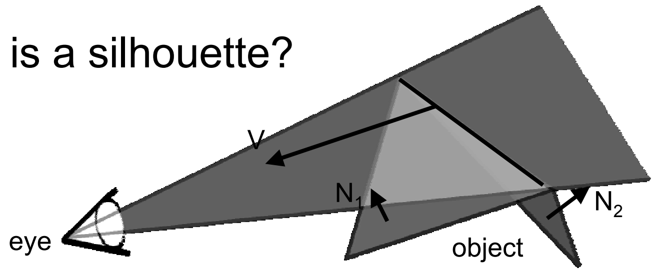
- Finding volumes
  - Project out shadow volumes
- Rendering
  - Render scene into z-buffer, freeze z-buffer
  - Draw front-facing volumes in front of pixel
    - increment stencil
  - Draw back-facing volumes in front of pixel
    - decrement stencil
  - If (cnt == 0) lit else shadow

Multiple Shadow Volumes



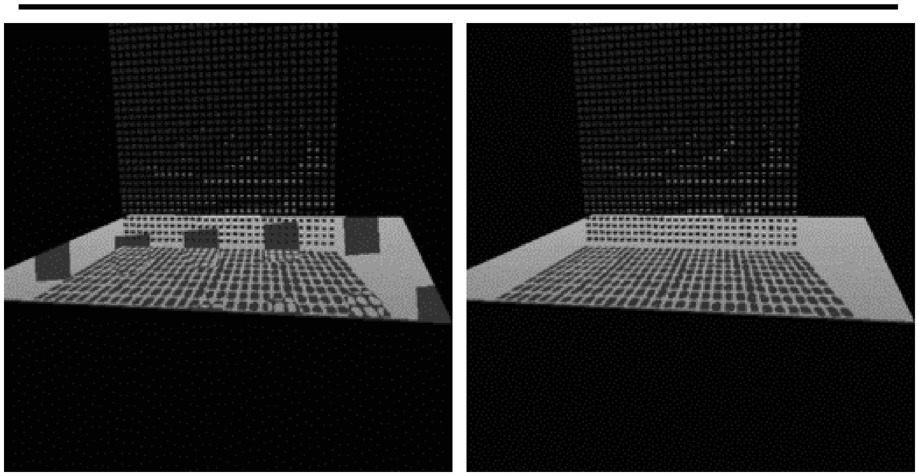
Shadow Volumes Properties

- Performance: Use the silhouette for speed
- What is a silhouette?



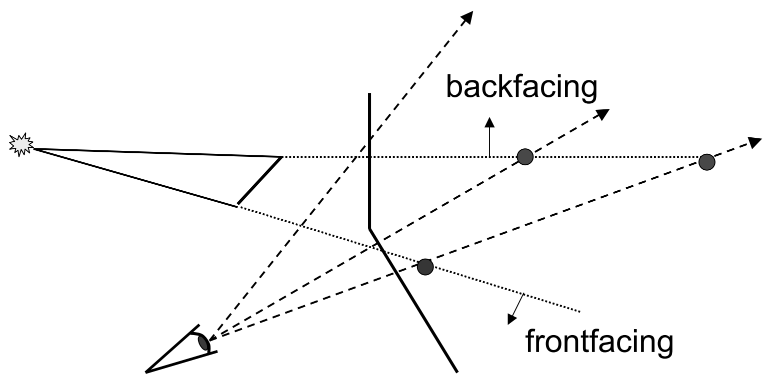
$N_1 \cdot V > 0$ (forward facing)
$N_2 \cdot V < 0$ (backward facing)

Near Plane Clip Issues



- Near plane clip discards part of shadow volume, messes up count

Z-fail Approach



But

- Far clipping plane problems?
- Use homogeneous coordinate to map to infinity

Performance

- Have to render lots of huge polygons
  - Front face increment
  - Back face decrement
  - Possible capping pass
- Uses a LOT fill rate
- Gives accurate shadows
  - IF implemented correctly
- Need access to geometry if want to use silhouette optimization

Summary

- Shadow maps
  - Render scene twice per frame
    - If static, can reuse
  - Uses projective texturing, requires hardware support/shaders
- Shadow volumes
  - Use stencil buffers

© Kavita Bala, Computer Science, Cornell University

Comparison

- Shadow Maps
  - Adv: Fixed resolution, fast, simple
  - Disadv: Bias, aliasing
- Shadow Volumes
  - Adv: Accurate, high-quality
  - Disadv: Fill-rate limited, hard to implement robustly

Approaches to Improve Shadows

- Hard Shadows
  - Adaptive Shadow Maps [Fernando, Fernandez, Bala, Greenberg]
  - Shadow Silhouette Maps[Sen, Cammarano, Hanrahan]
- Hard and Soft Shadows
  - Edge-and-Point Rendering [Bala, Walter Greenberg]
- Soft Shadows
  - Next time

Adaptive Shadow Maps: Motivation

- Fernando, Fernandez, Bala, Greenberg [SIG01]

- Shadow maps require too much tweaking
  - Where to place light?
  - What resolution to use?

- Goals:
  - Address the aliasing problem
  - No user intervention
  - Interactive frame rate

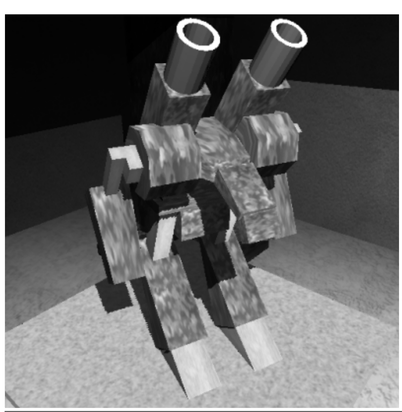
Adaptive Shadow Maps

- Idea:
  - Refine shadow map on the fly
- Goal:
  - Shade each eye pixel with a different shadow map pixel
- Implementation:
  - Use hierarchical structure for shadow map
  - Create/delete pieces of shadow map as needed
  - Exploit fast rendering and frame buffer read-backs

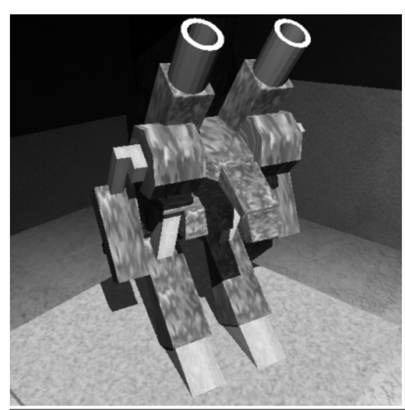
ASM Data Structure

- Simple 2D tree:

Results: Images (Robot)

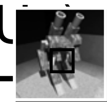


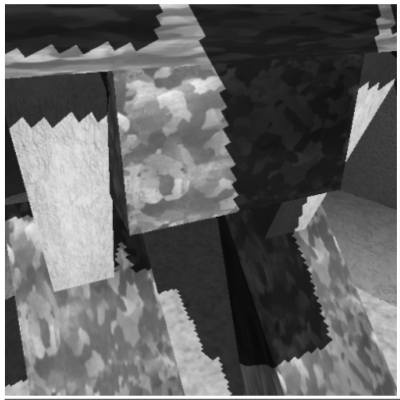
Conventional Shadow Map
(2048 x 2048 pixels)
16 MB Memory Usage



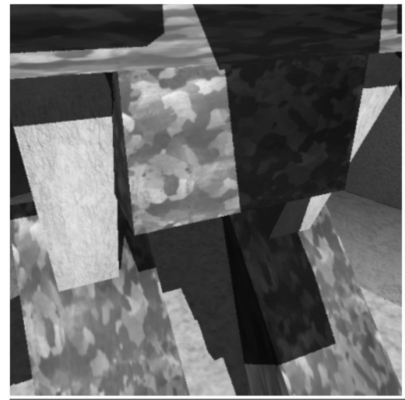
Adaptive Shadow Map
(Variable Resolution)
16 MB Memory Usage

Results: Images (Robot Close-U





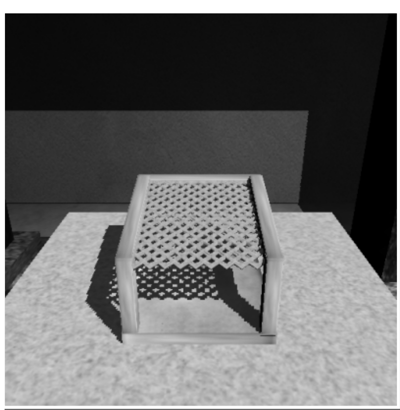
Conventional Shadow Map
16 MB Memory Usage



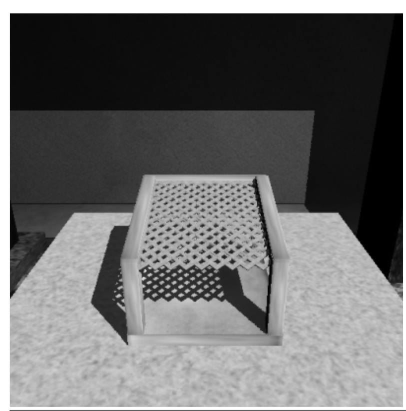
Adaptive Shadow Map
16 MB Memory Usage

Equivalent Conventional Shadow Map Size:
65,536 × 65,536 Pixels

Results: Images (Mesh)

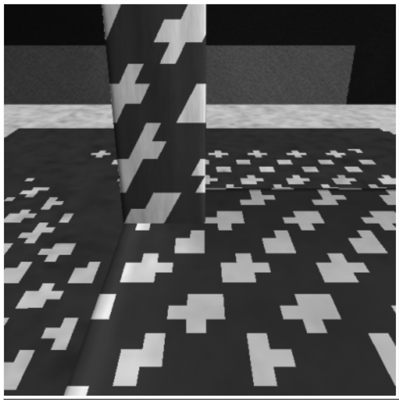


Conventional Shadow Map
(2048 x 2048 pixels)
16 MB Memory Usage

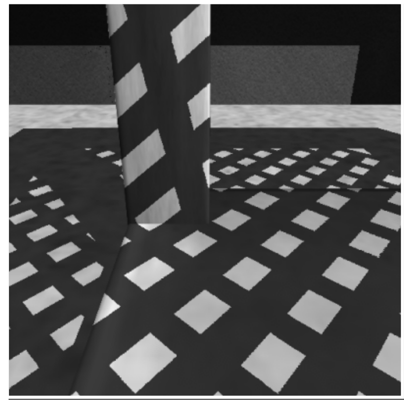


Adaptive Shadow Map
(Variable Resolution)
16 MB Memory Usage

Results: Images (Mesh Close-Up)



Conventional Shadow Map
16 MB Memory Usage



Adaptive Shadow Map
16 MB Memory Usage

Equivalent Conventional Shadow Map Size:
65,536 × 65,536 Pixels