Lecture 19: Many Lights

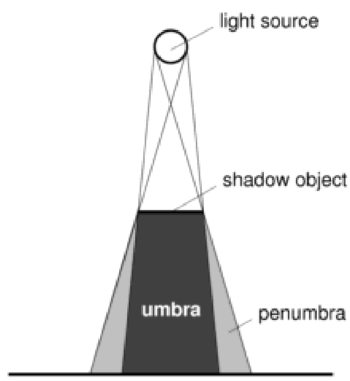
Fall 2004
Kavita Bala
Computer Science
Cornell University

Announcements

- HW 3 out
  - Due next Friday

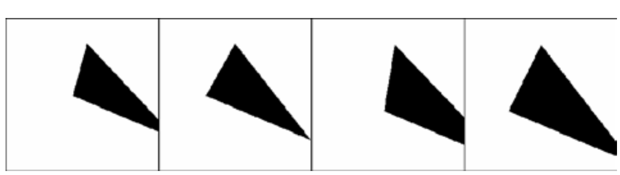
Soft Shadows

- Soft shadows appear natural
- Hard to get soft shadows in hardware
- Slow in software

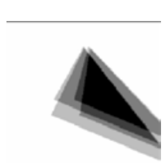


Heckbert and Herf

- Use accumulation buffer
- Render shadows from multiple point lights over the area light (like MC)
- Accumulate shadows

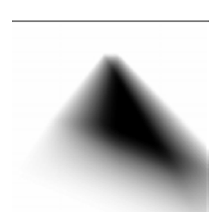


2 x 2 samples



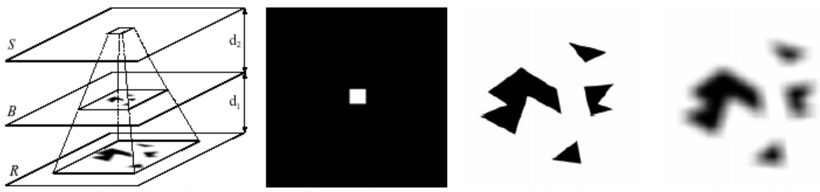
average

16 x 16 samples



Soler and Sillion

- Shadows as convolution



(a) Parallel configuration (b) Source image (c) Blocker image (d) Convolution

Penumbra Maps

- Wyman and Hansen
- Use shadow map and Haines technique for soft shadows on arbitrary surfaces
- Penumbra map
- Stores intensity of shadow
- Overall:
  - 3 pass: shadow map and penumbra map
  - Render image using depth from shadow map and intensity from penumbra map

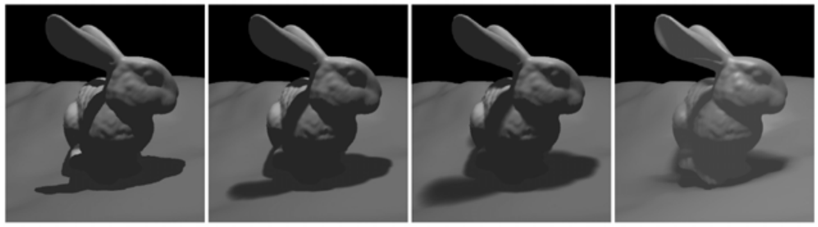


Figure 7: *Comparison of the Stanford Bunny with shadow maps (left), penumbra maps with two different sized lights (center), and a pathtraced shadow using the larger light (right). For this data set, we generate shadows using a 10k polygon model and render the shadows onto the full (~70k polygon) model.*

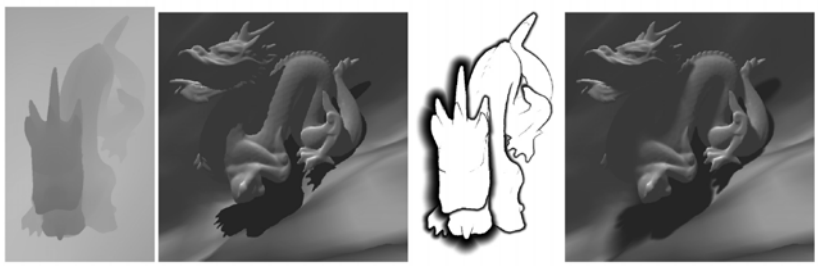
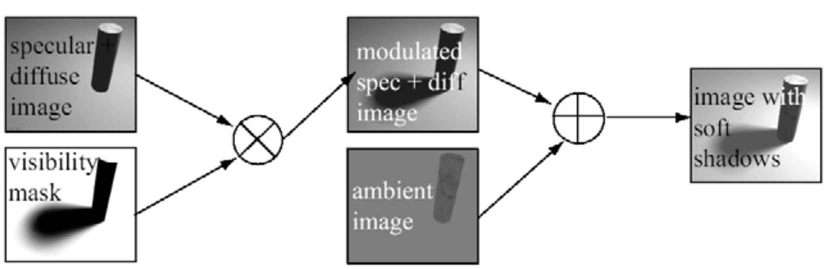


Figure 8: *Using a standard shadow map results in hard shadows (left), add a penumbra map to get soft shadows (right). Using a 10k polygon dragon model for the shadows and a 50k polygon model to render, we get 14.5 fps at 1024x1024.*

Geometry-Based Soft Shadow Volume

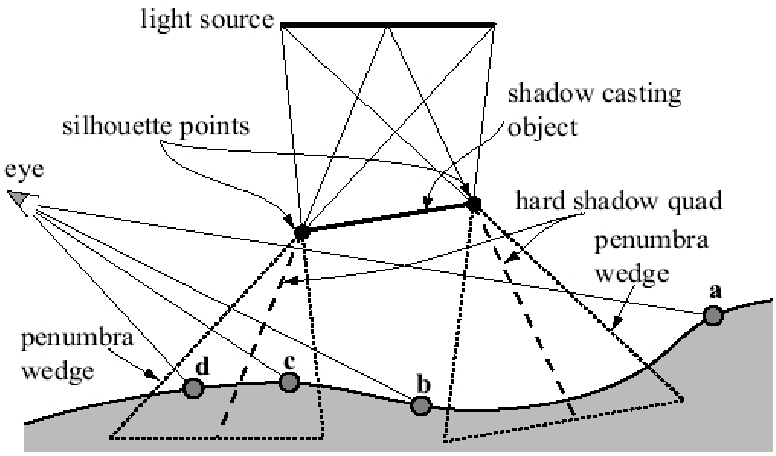
- Assarsson and Moller
- Shadow volume approach



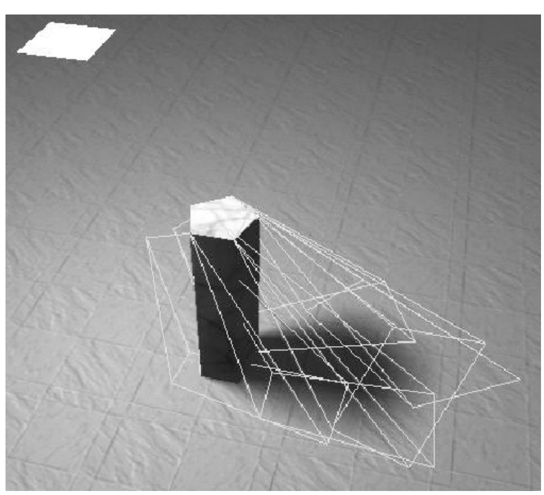
V-buffer stores visibility factor [0,1]

Computing Visibility: 2 passes

- Shadow volume quads are rendered into the V-buffer: overestimates umbra
- Penumbra wedges are rendered to compensate



Method Details: Wedge Example



512 x 512 at 5 FPS (software)

Visibility Passes

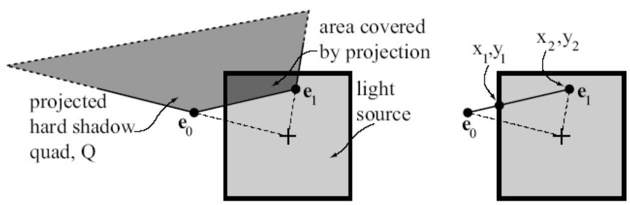
Pass 1: Render shadow volume quads

Pass 2 : Compute visibility for each pixel inside the shadow wedges:

Point p = (x,y,z): find visibility of p

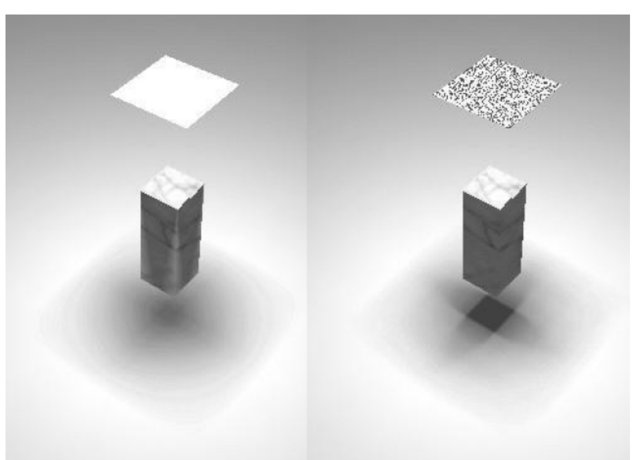
Precompute 4D coverage textures to accelerate visibility computation

Can handle textured lights, video textures



Assumptions

- Silhouettes are constant
- Overlapping objects



Results

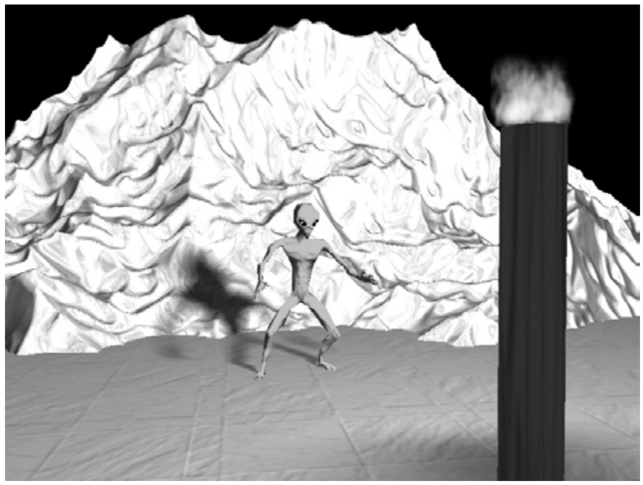


Image of fire used as light source

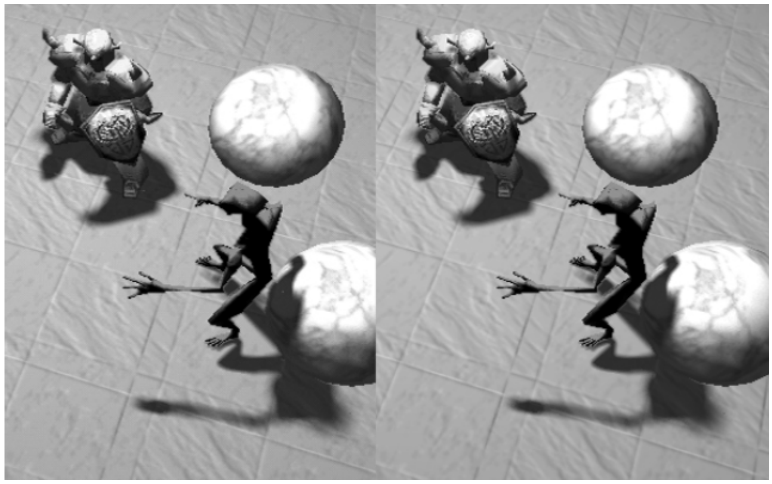
Results

Soft Shadow Volume | 256 Samples | 1024 Samples

Resolution: 512 x 512 @ 0.14 FPS

100 x 100 @ 3.00 FPS
256 x 256 @ 0.51 FPS

Results



Software Implementation　　　Hardware Implementation

Summary

- Hard shadows
  - Adaptive shadow maps
  - Edge-and-point rendering
  - Silhouette shadow maps
  - …
- Soft shadows
  - Accumulation Buffer
  - Convolution
  - Penumbra Maps
  - Penumbra Wedges
  - …

Many Lights

Motivation

- Most techniques work for single light source
- Many light sources
  - Treat it is a single integration domain
  - Importance sample lights
  - Importance sampling (with visibility) still hard problem

Research on many lights

- Ward '91
- Shirley, Wang, Zimmerman '94
- Fernandez, Bala, Greenberg '02
- Wald and Slusallek '03
- Environment Map Sampling…

Ward '91

- Many lights in RADIANCE
- But all contributions not important
- Ignore some lighting at a point
  - User-defined cutoff: x%
- Sort lights according to potential contribution
  - Include G, cosine, L
  - EXCLUDE visibility

Ward '91

- Go through sorted list from the biggest potential contribution
  - Keep running count of visible contribution: V
  - Remainder of list (if fully visible) = R
  - Stop if R < x% of V

Ward '91

- But just can't ignore remainder R
- Estimate remainder using visibility statistics from previous shadow tests: hack!
- Performance: 2x to 5x
- But, requires computing all potential contributions
  - Can be expensive for many lights

Shirley, Wang, Zimmerman ‘94

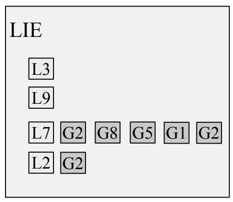
- Try to avoid linear cost of evaluating lights
- Separate lights into
  - Set of important lights (a small set)
  - Set of “dim” lights (large set)
- Construct pdf using:
  - all important lights
  - 1 out of all the dim lights
- Importance sample these lights

Shirley, Wang, Zimmerman ‘94

- Region of influence for important lights
  - Octree cells in region of influence have light in important set
- However, the partitioning into important and dim sets remains hard
- Also, still are not taking visibility into account

Fernandez, Bala, Greenberg '02

- Local Illumination Environment (LIE):
  lights and blockers that affect octree cell



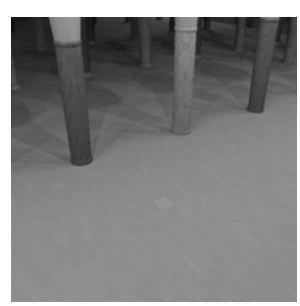
Takes visibility into consideration!

Fernandez, Bala, Greenberg '02

- All lights/shadows are not visually important
- Weber's law: 2% cutoff

=

…



All lights

Using Masking

- Bright lights can mask out shadow details
- Weber's Law: variations in lighting are not visible if ambient lighting is bright enough
  - Conservative: 2% cutoff
- LIE: remove relatively dim lights (fully/partially visible)
  - Cheaper shading
  - Maximum light contribution < 2% of dimmest point in cell
  - Actually, cumulative maximum light contribution < 2% of dimmest point in cell

Using Masking

Remove from LIE if
cumulative maximum light contribution < 0.02x40

	Light 1	Light 2	Light 3	Light 4	Light 5	Total
Point 1	0	0.1	16	30	20	66.1
Point 2	0	0	17	0	25	42
Point 3	0.1	0.1	9.8	12	18	40
Point 4	0.1	0.5	14	32	23	69.6
Max	0.1	0.5	17	32	25	
Sorted	~~0.1~~	~~0.5~~	17	25	32	

Fernandez, Bala, Greenberg '02

Image

Without Masking

With Masking

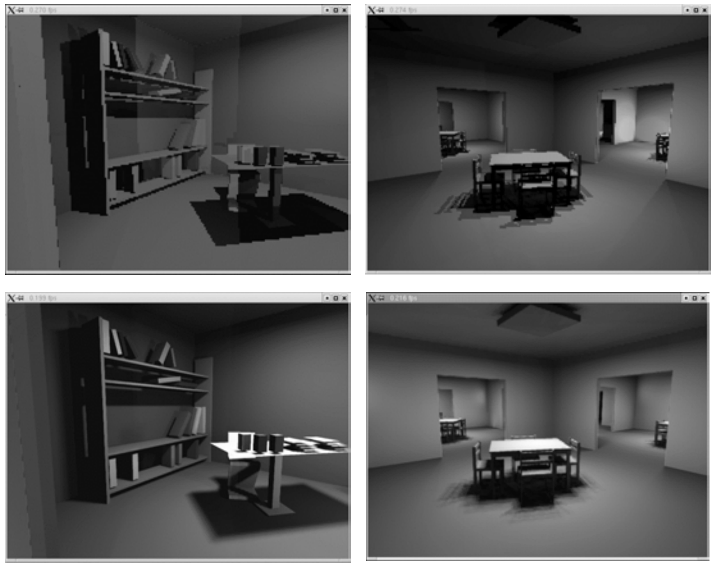
Cost

Cheap Expensive

Wald Slusallek '03

- PDF for sampling visible lights in interactive setting
  - Assume significant occlusion
  - Each room influenced by few lights
- → 2-step algorithm (every frame)
- 1st step: Determine important (unoccluded) lights by crude path tracing
- 2nd step: Importance samples these lights
  - Completely ignore (probably) occluded lights

Wald Slusallek ‘03



Rendering w/ Environment Maps

- High lighting complexity

- Rich: captures real world

Ambient Occlusion

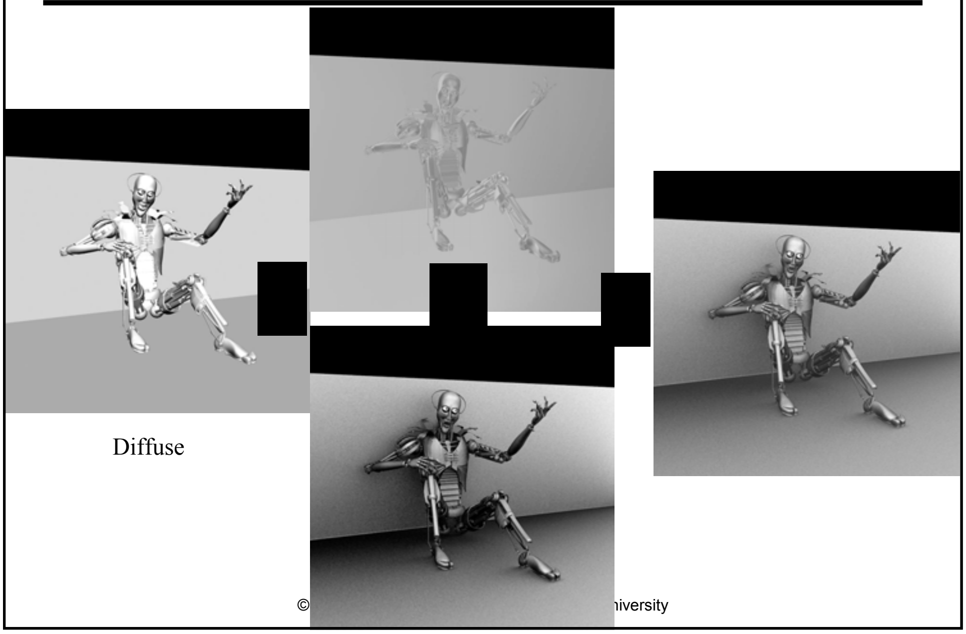
- Interactive hardware rendering with many lights?
- Traditionally “fake” diffuse illumination using an ambient term
- But this just results in a constant additon
- Ambient occlusion adds some visibility to the fake diffuse illumination

Ambient Occlusion

- Pre-compute the ambient term
- At each vertex, shoot rays over hemisphere (cosine weighted)
  - MC sampling: sample hemisphere
- Does it hit a surface or escape? Compute average visibility (V = 1 – hits/samples)
- Ambient Out = Ambient In * V

© Kavita Bala, Computer Science, Cornell University

Ambient Occlusion Example



Diffuse

Problem

- Can move object around without deforming it
- But, slow!
- How to render interactively with many lights?
  – Open question