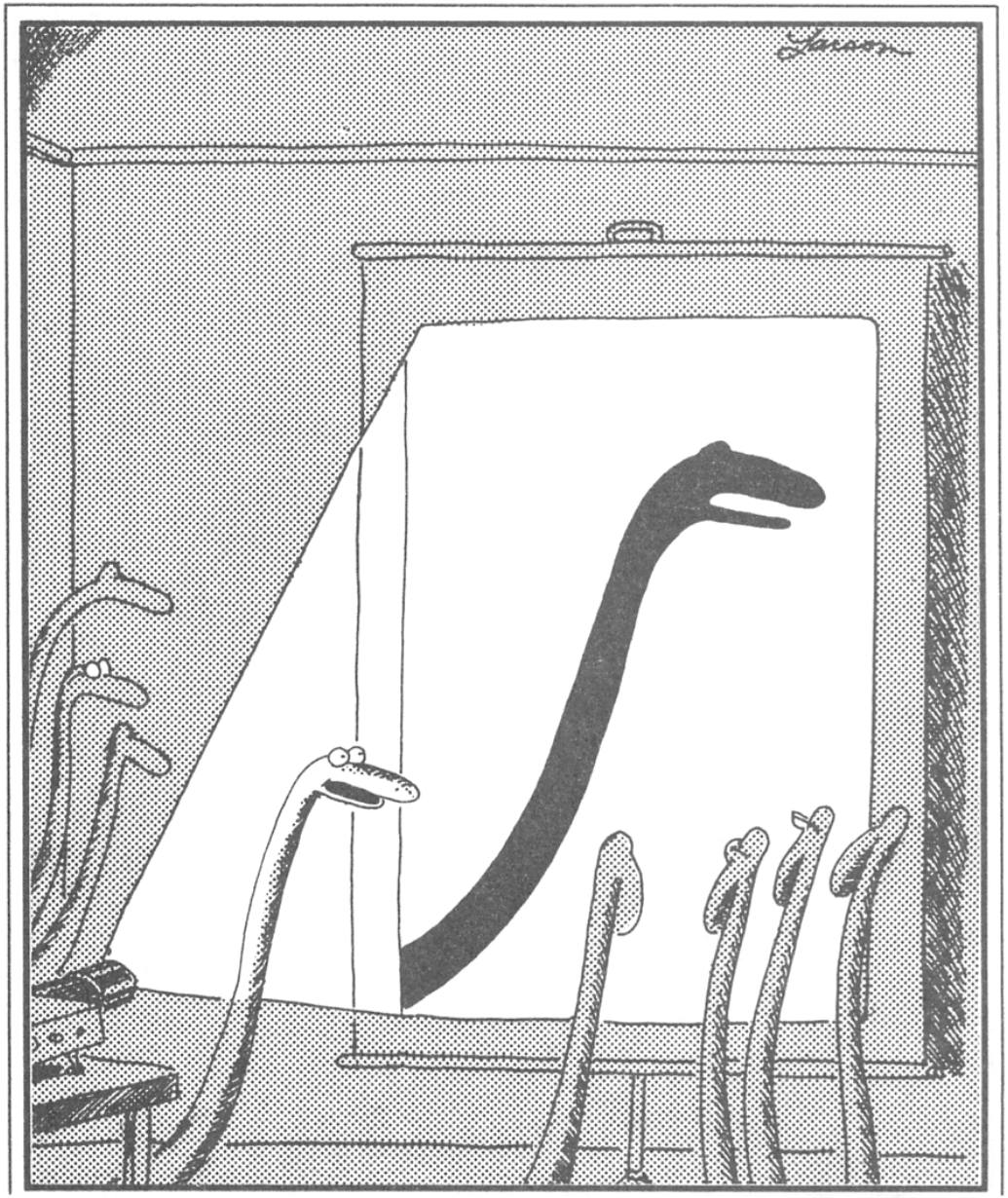
MIT EECS 6.837
Computer Graphics

Wojciech Matusik

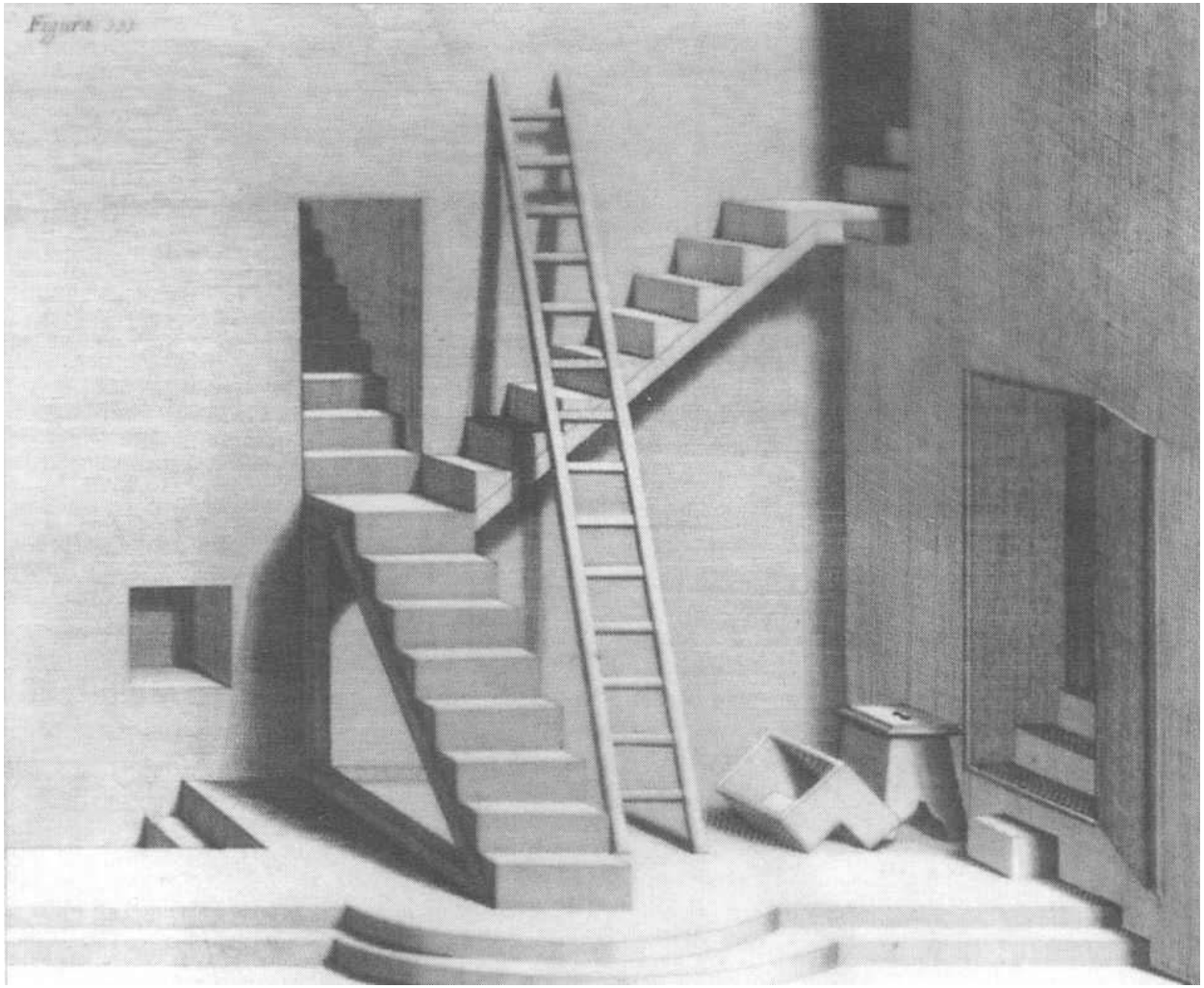
Real-Time Shadows



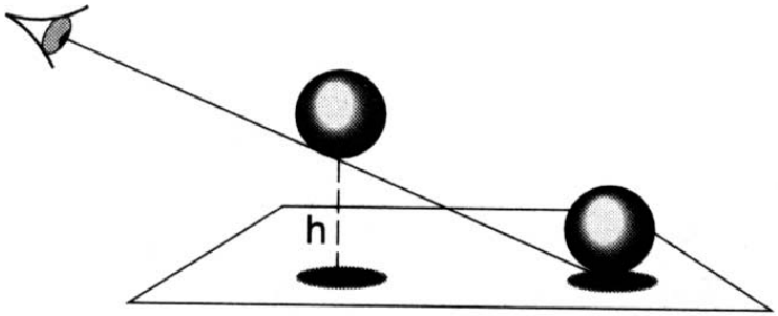
"Now this is… this is… well, I guess it's another snake."

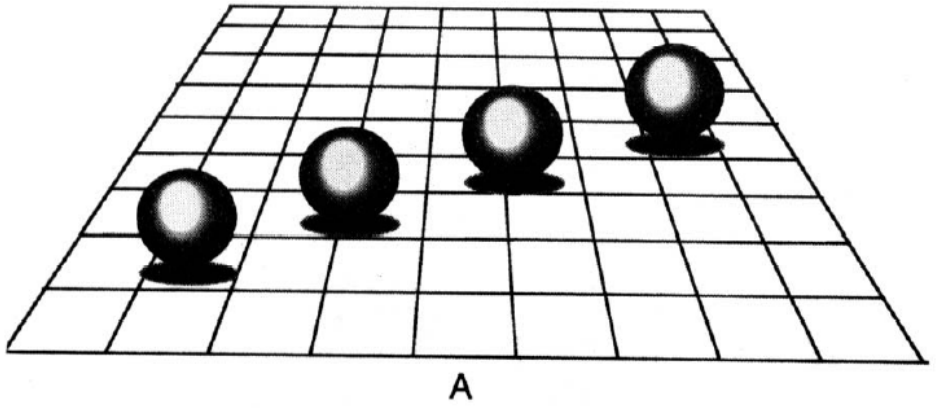
Why are Shadows Important?

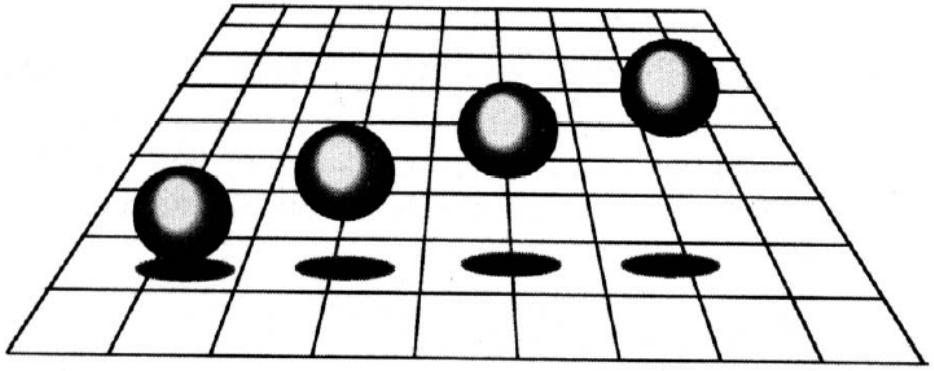
- Depth cue
- Scene Lighting
- Realism
- Contact points



Shadows as a Depth Cue



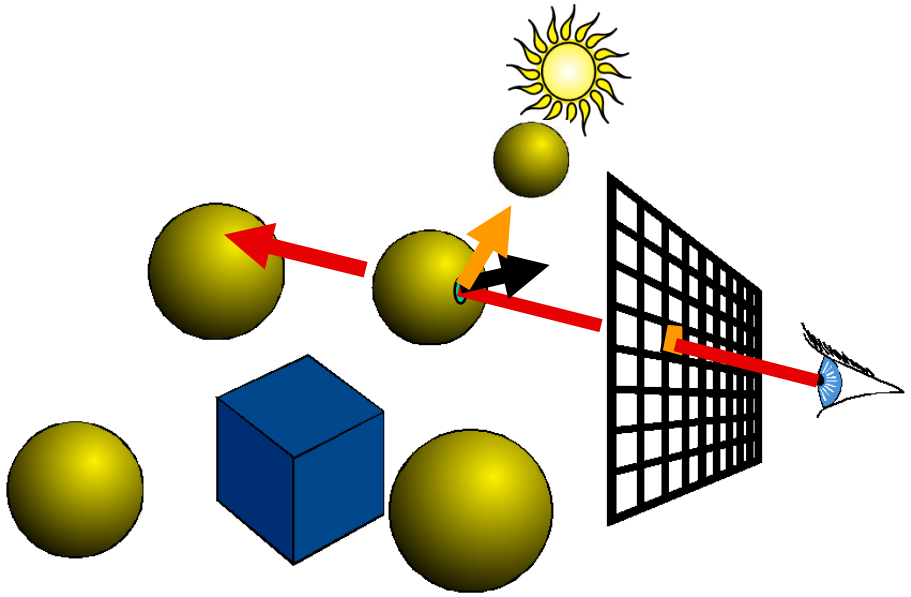




Shadows as the Origin of Painting

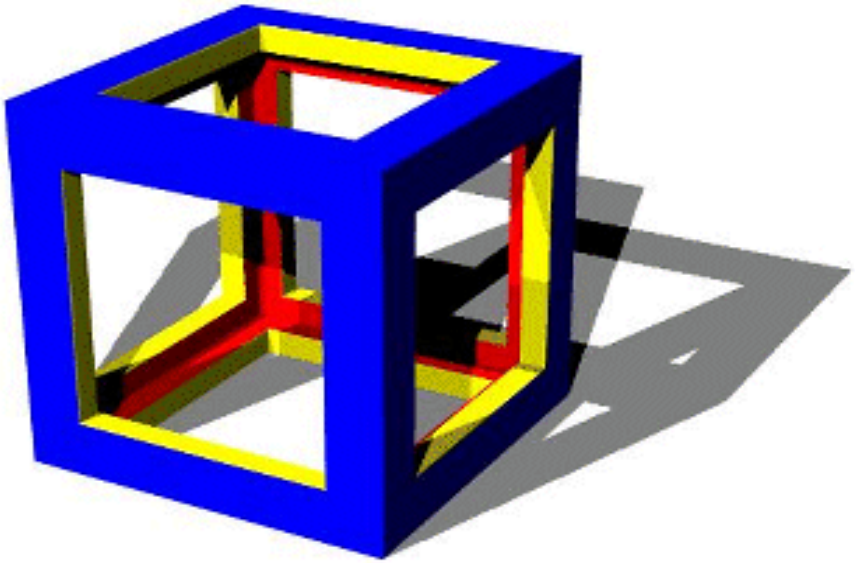
Shadow in Ray Tracing

- Trace secondary (shadow) rays towards each light source
- If the closest hit point is smaller than the distance to the light then the point is in shadow



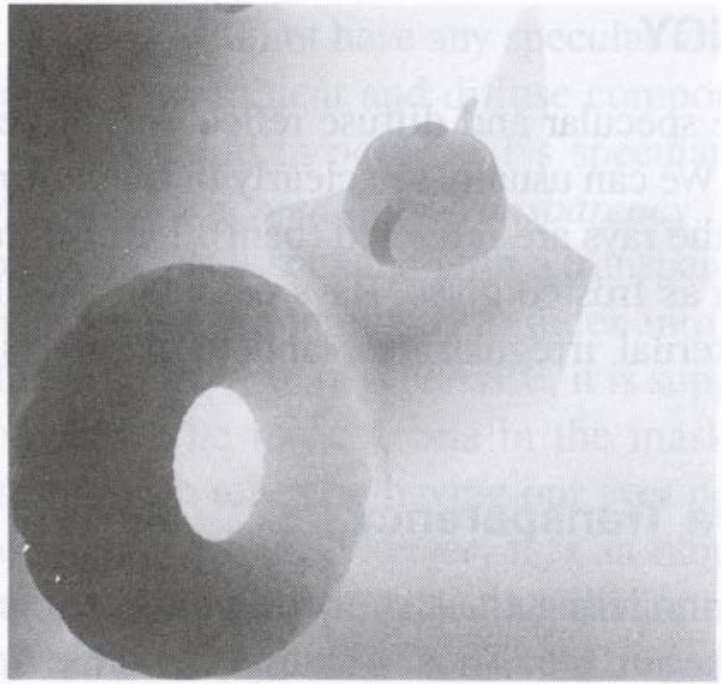
Today

- Shadow Maps
  - Shadow/View Duality
  - Texture Mapping
- Shadow Volumes
- Deep Shadow Maps
- Alias-free Shadow Maps



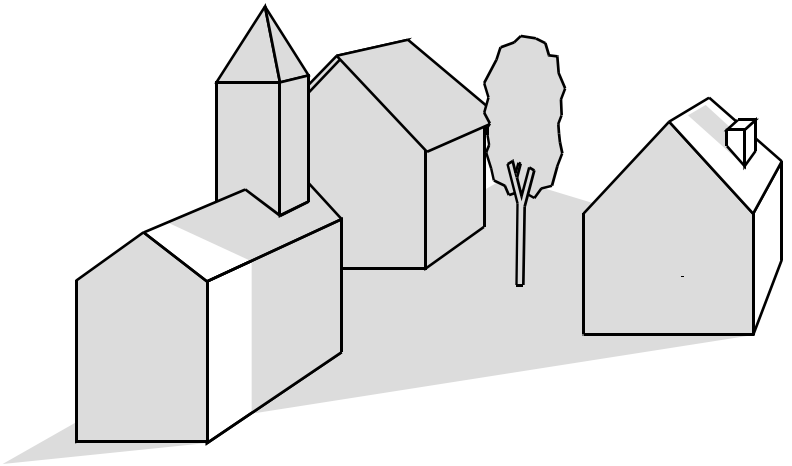
Shadow Maps Are Important

- Both in high-end production software...
  - Pixar's RenderMan
- ... and in real-time rendering (e.g. games)

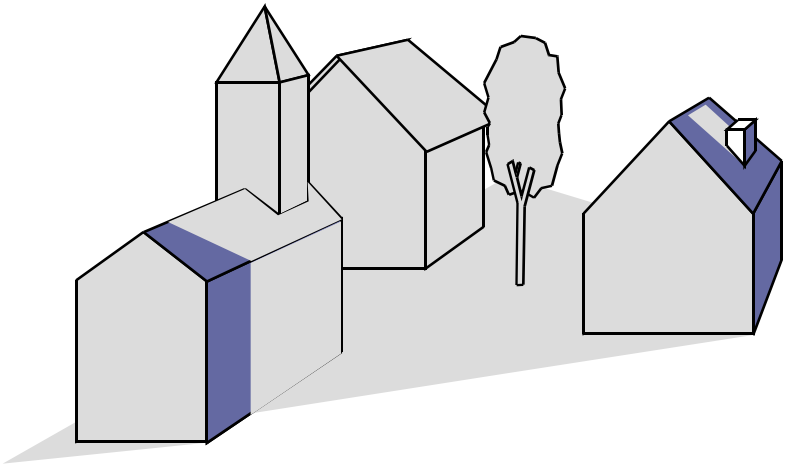


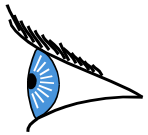
Shadow/View Duality

- A point is lit if it is visible from the light source



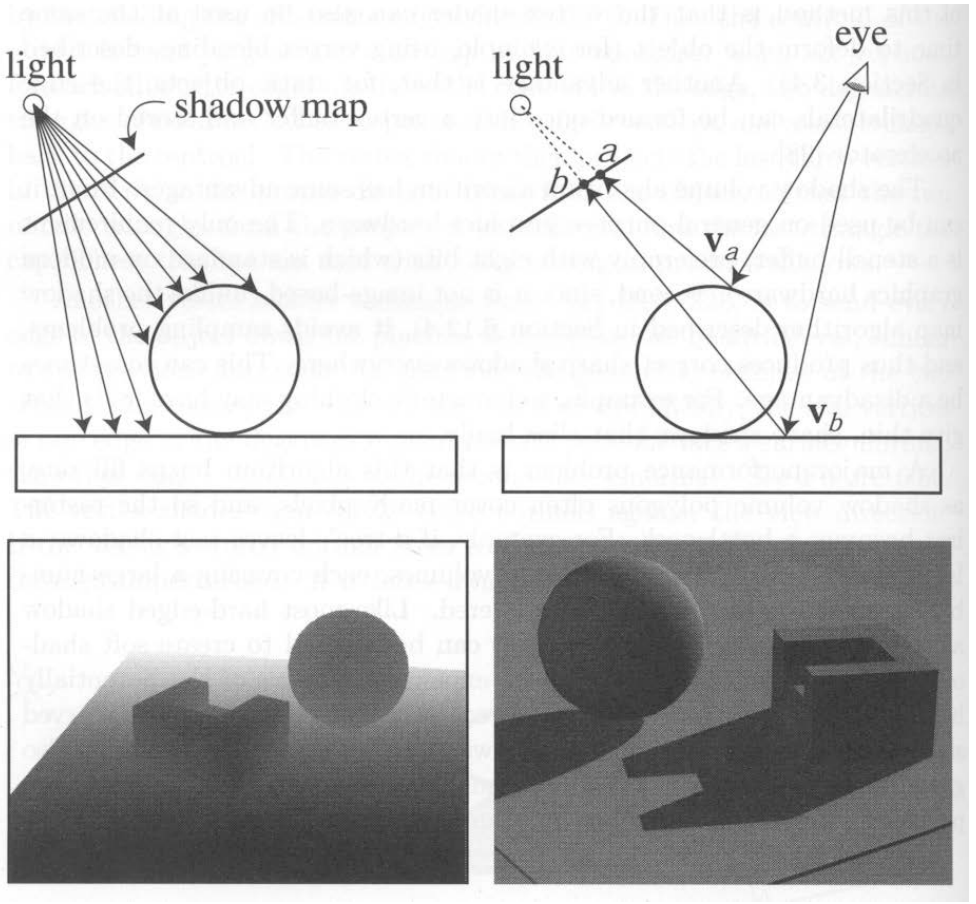
- Shadow computation similar to view computation





Shadow Mapping

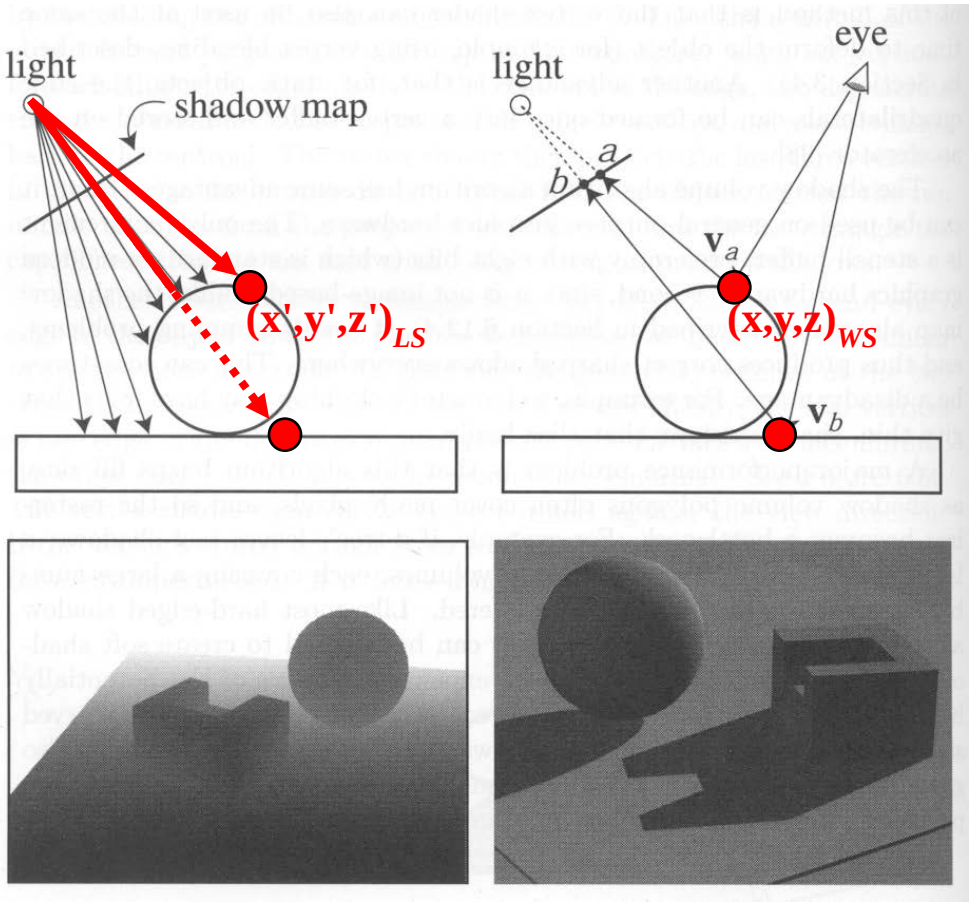
- Texture mapping with depth information
- 2 passes
  - Compute shadow map == depth from light source
    - You can think of it as a z-buffer as seen from the light
  - Render final image, check shadow map to see if points are in shadow



Foley et al. “Computer Graphics Principles and Practice”

Shadow Map Look Up

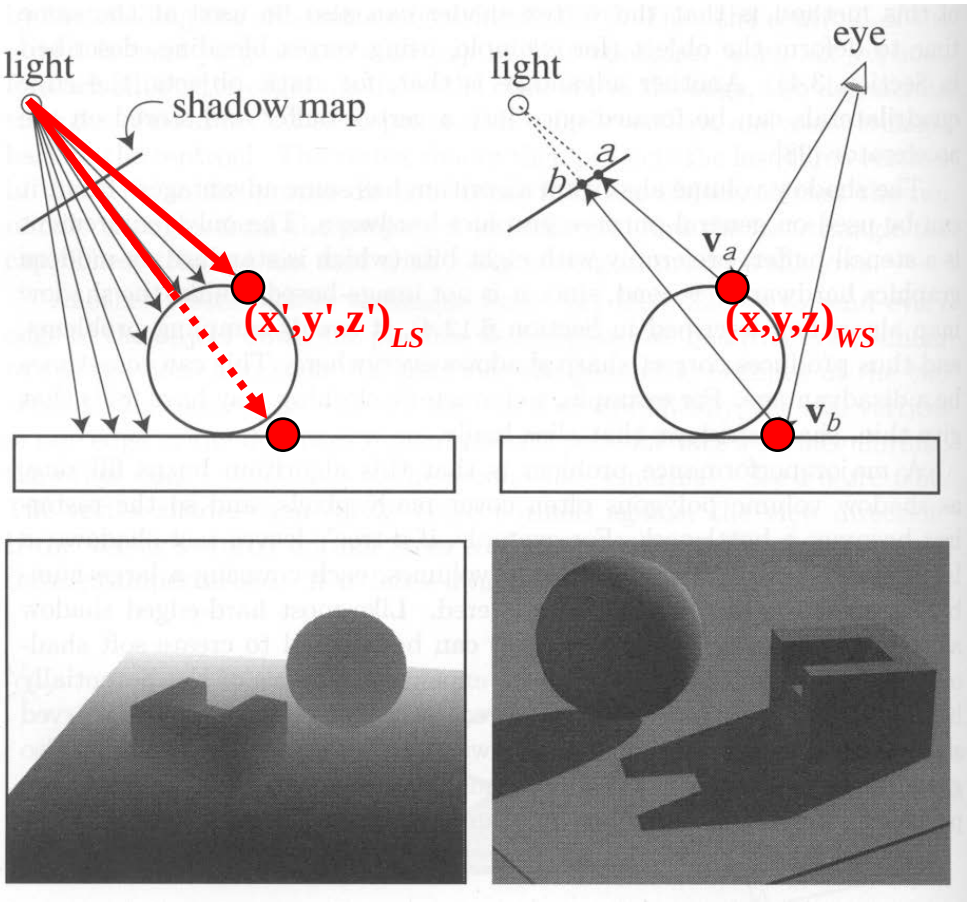
- We have a 3D point $(x,y,z)_{WS}$
- How do we look up the depth from the shadow map?
- Use the 4x4 perspective projection matrix from the light source to get $(x',y',z')_{LS}$
- ShadowMap(x',y') < z'?



Foley et al. “Computer Graphics Principles and Practice”

Shadow Map Look Up Questions?

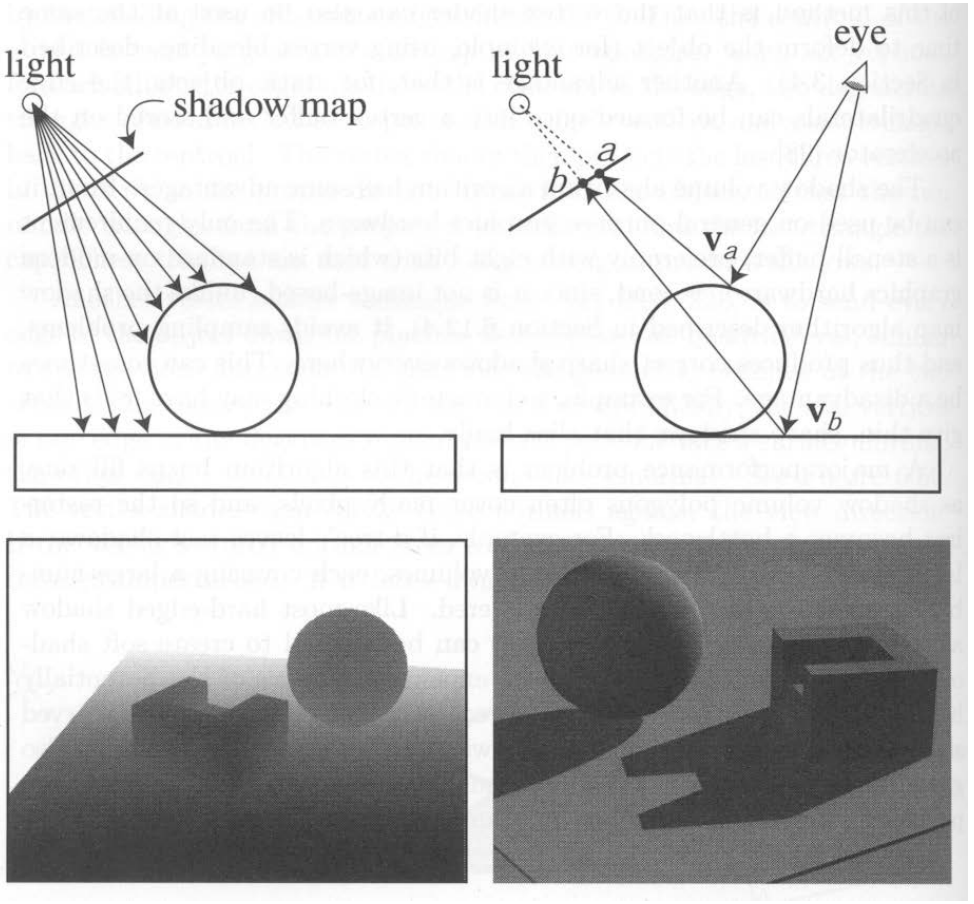
- We have a 3D point $(x,y,z)_{WS}$
- How do we look up the depth from the shadow map?
- Use the 4x4 perspective projection matrix from the light source to get $(x',y',z')_{LS}$
- ShadowMap(x',y') < z'?



Foley et al. “Computer Graphics Principles and Practice”

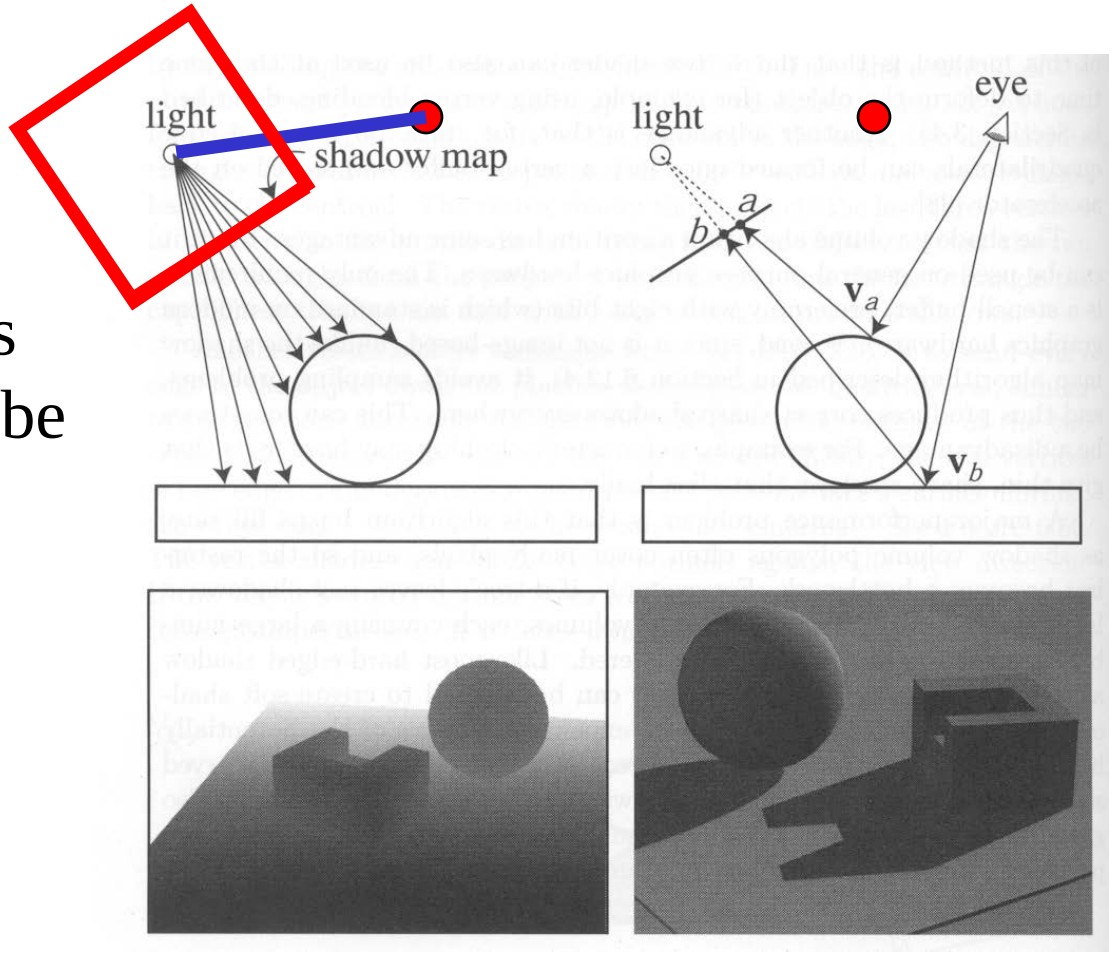
Limitations of Shadow Maps

1. Field of View
2. Bias (Epsilon)
3. Aliasing



1. Field of View Problem

- What if point to shadow is outside field of view of shadow map?
  - Use 6 shadow maps on the faces of a cube
  - ... or use only spot lights



2. The Bias (Epsilon) Nightmare

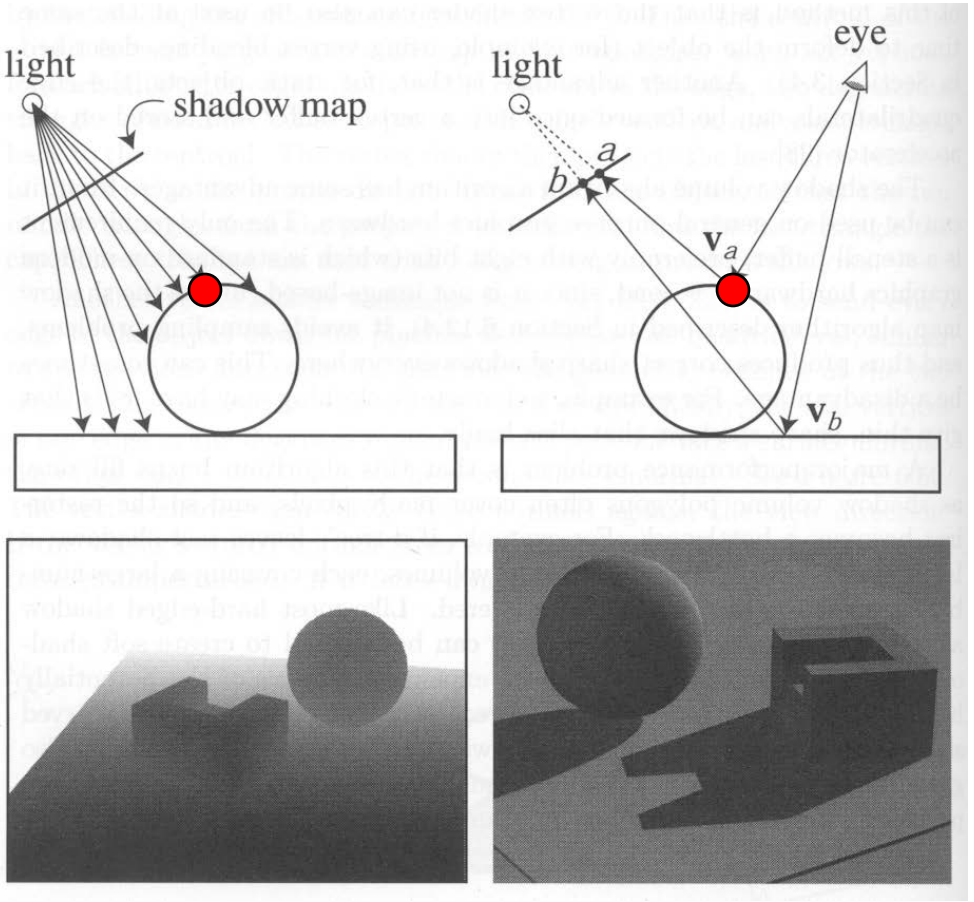
- For a point visible from the light source

  ShadowMap(x',y') ≈ z'

  – But due to rounding errors the depths never agree exactly

- How can we avoid erroneous self-shadowing?

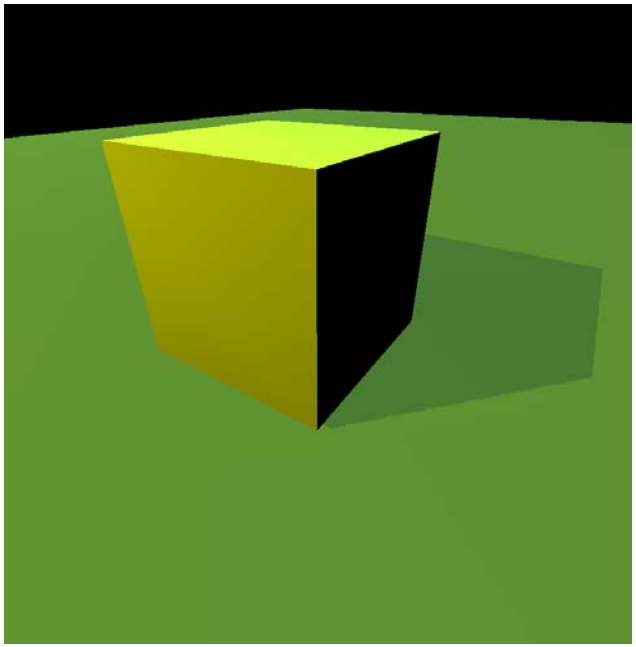
  – Add bias (epsilon)



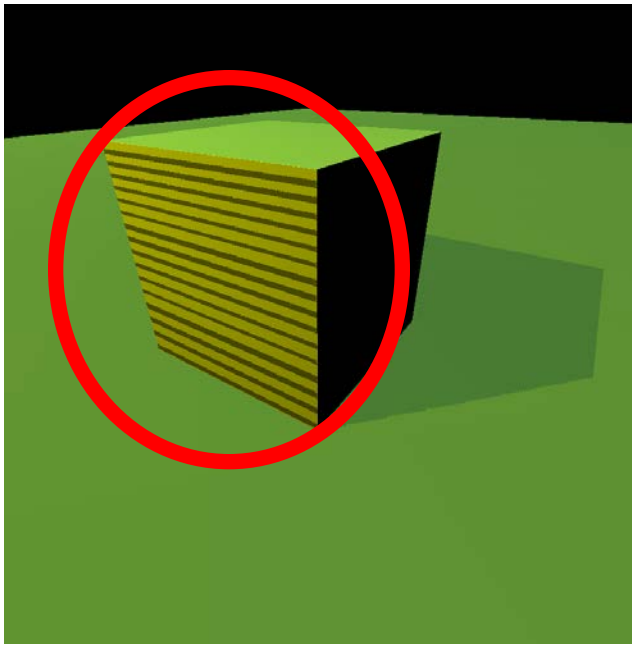
2. Bias (Epsilon) for Shadow Maps

ShadowMap(x',y') + bias < z'

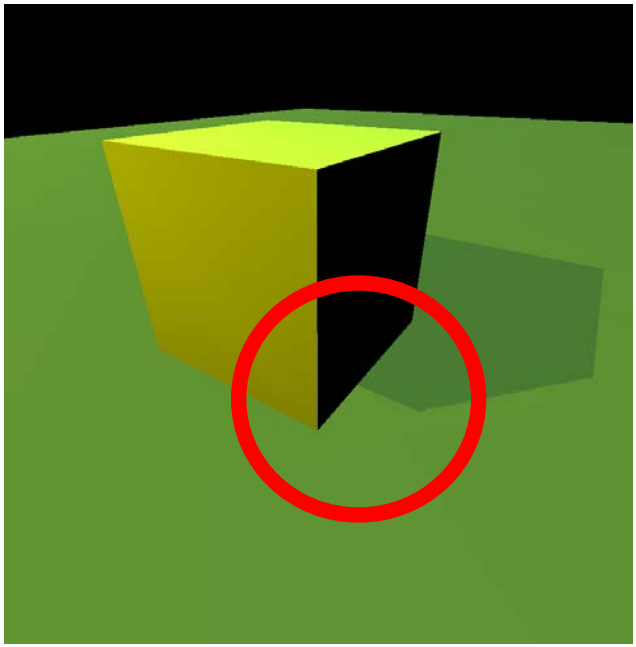
Choosing a good bias value can be very tricky



Correct image



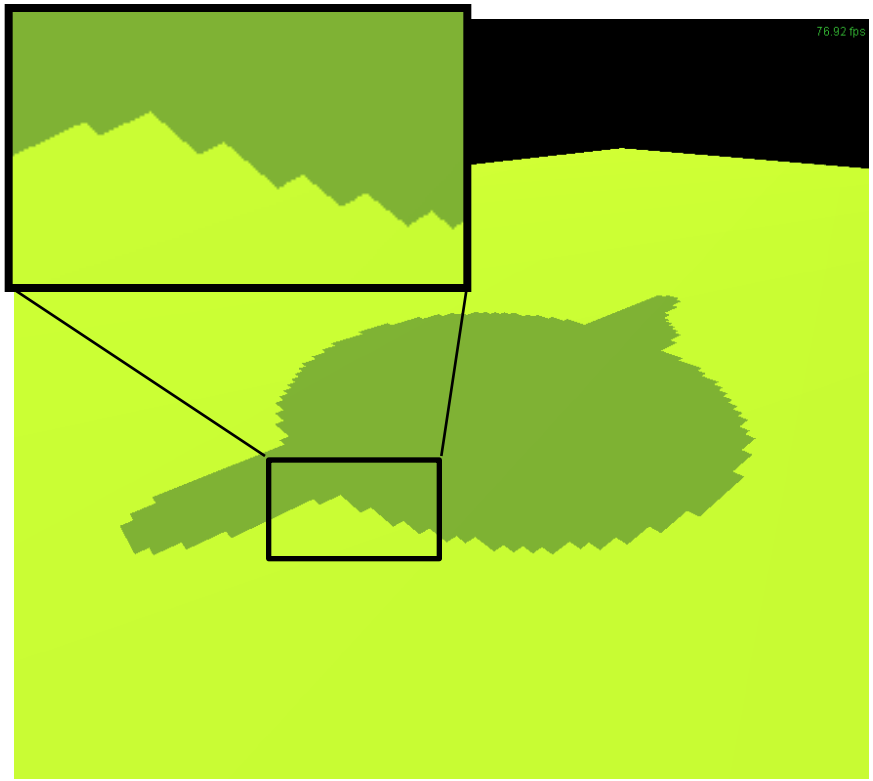
Not enough bias
("surface acne")



Way too much bias

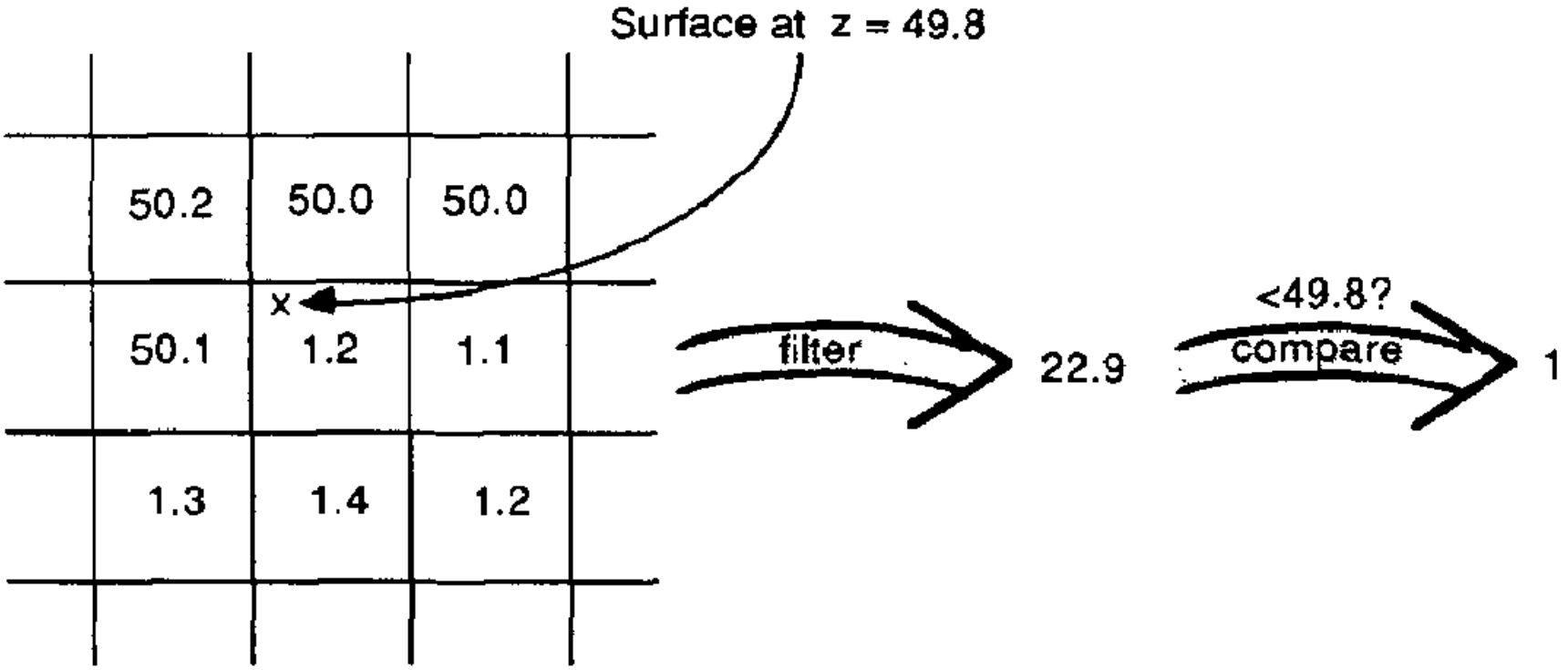
3. Shadow Map Aliasing

- Under-sampling of the shadow map
  - Jagged shadow edges



3. Shadow Map Filtering

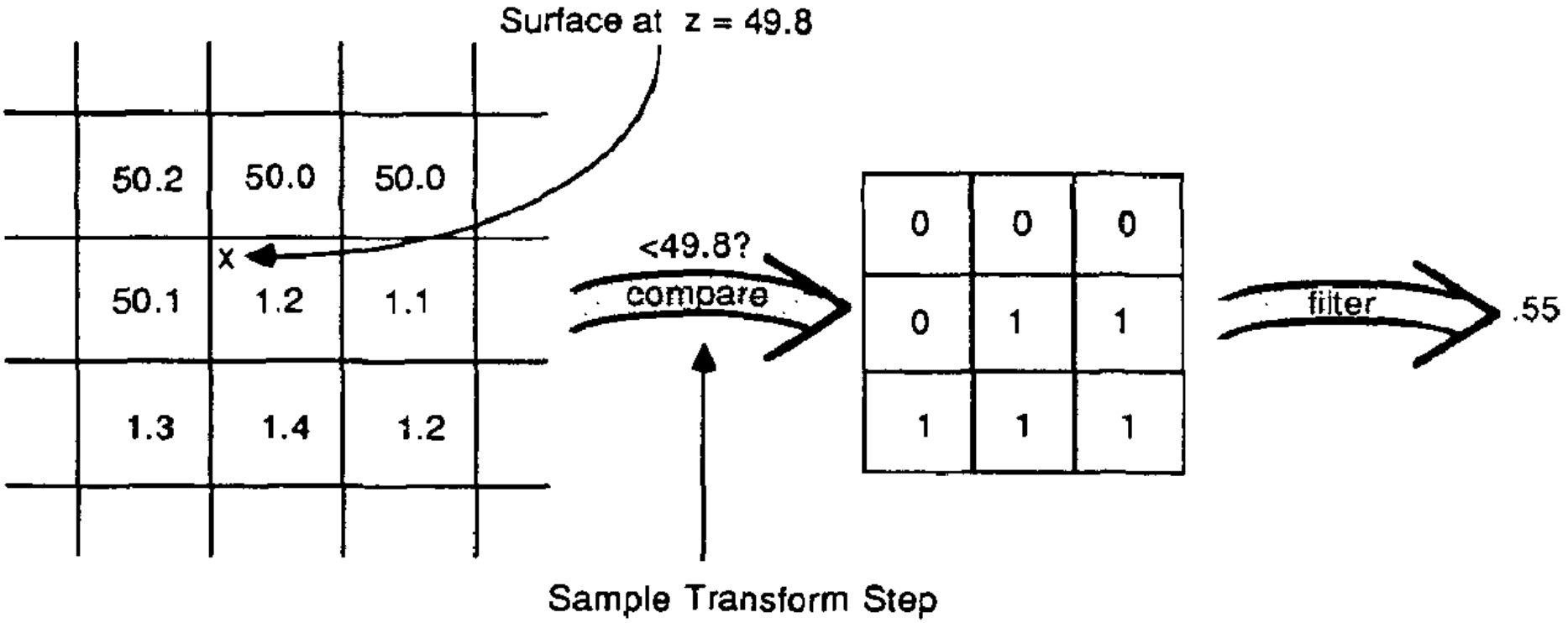
- Should we filter the depth?
  (weighted average of neighboring depth values)
- No... filtering depth is not meaningful



a) Ordinary texture map filtering. Does not work for depth maps.

3. Percentage Closer Filtering

- Instead we need to filter the *result* of the shadow test (weighted average of comparison results)



3. Percentage Closer Filtering

- 5x5 samples
- Nice antialiased shadow
- Using a bigger filter produces fake soft shadows
- Setting bias is tricky

3. Percentage Closer Filtering

- 5x5 samples
- Nice antialiased shadow
- Using a bigger filter produces fake soft shadows
- Setting bias is tricky

Questions?

Shadows in Production

- Often use shadow maps
- Ray casting as fallback in case of robustness issues

Figure 12. Frame from *Luxo Jr.*

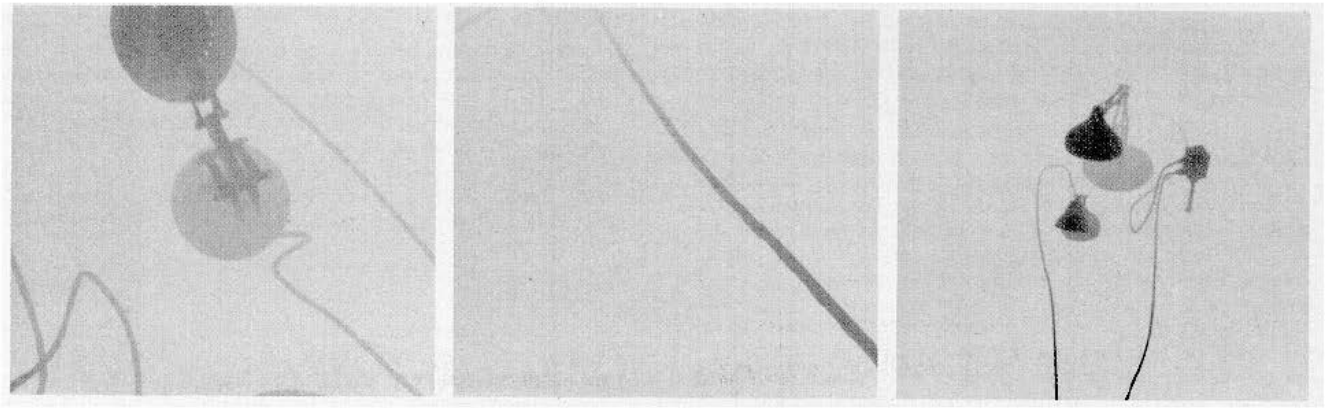
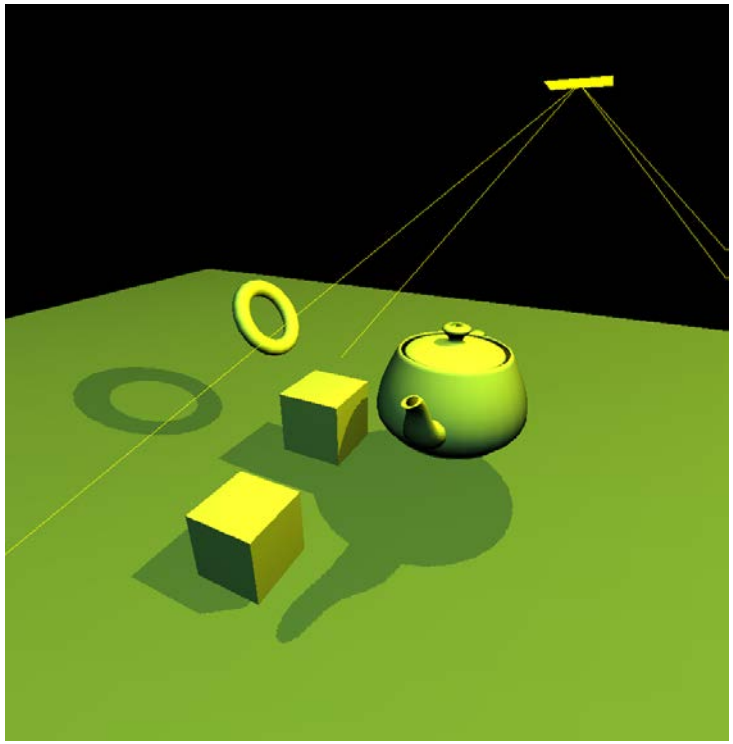
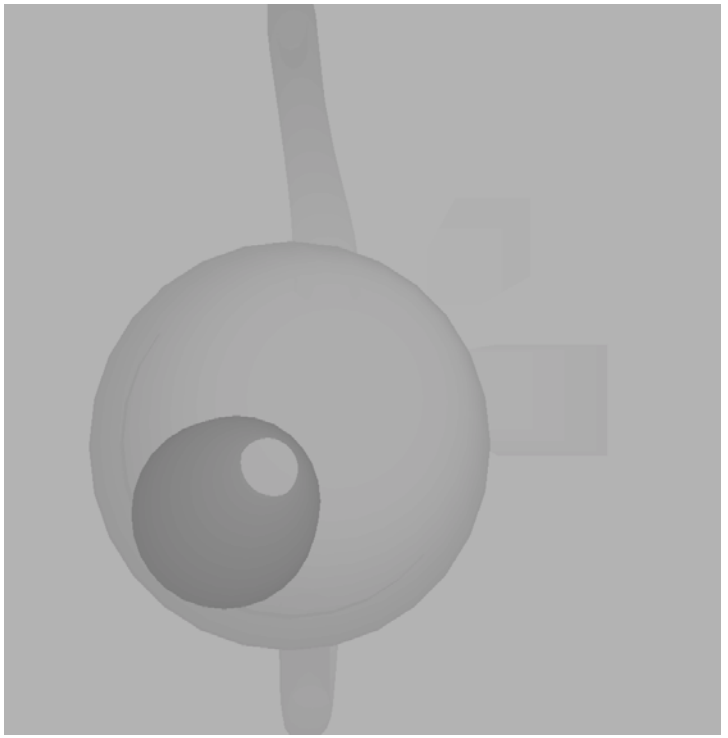


Figure 13. Shadow maps from *Luxo Jr.*

Hardware Shadow Maps

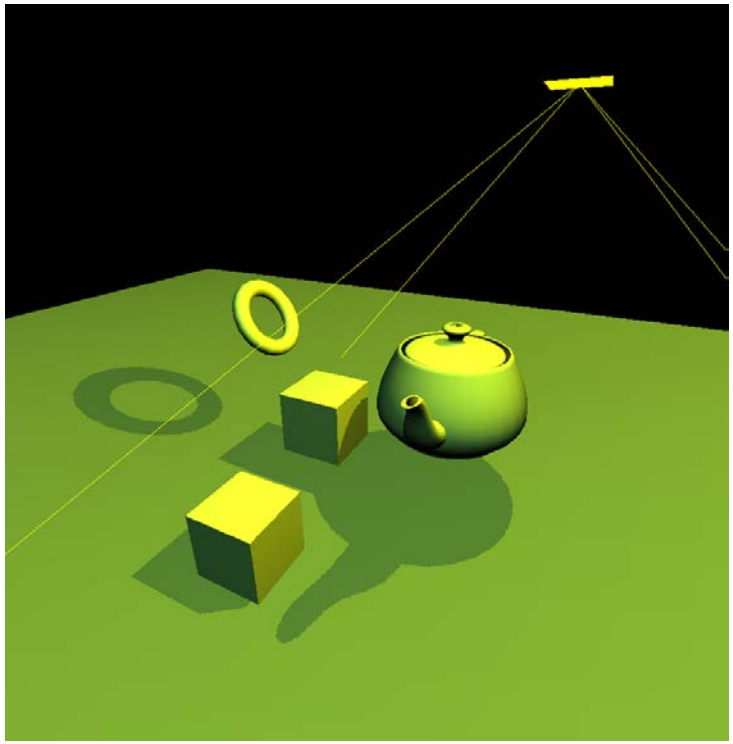
- Can be done with hardware texture mapping
  - Texture coordinates u,v,w generated using 4x4 matrix
  - Modern hardware permits tests on texture values

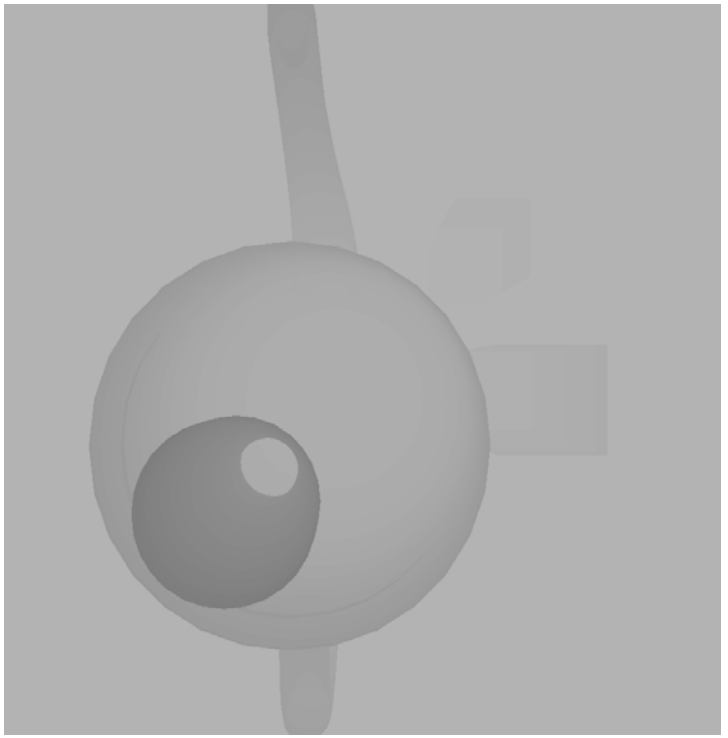




Hardware Shadow Maps Questions?

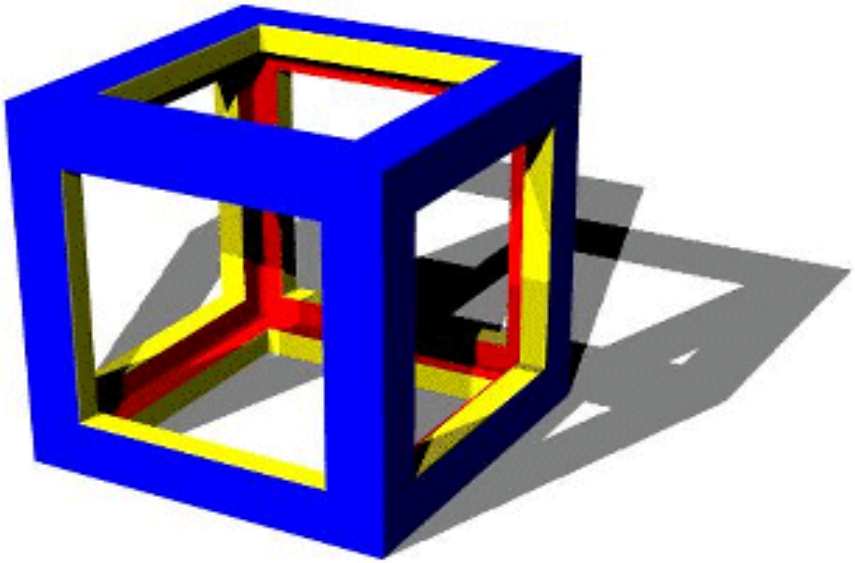
- Can be done with hardware texture mapping
  - Texture coordinates u,v,w generated using 4x4 matrix
  - Modern hardware permits tests on texture values





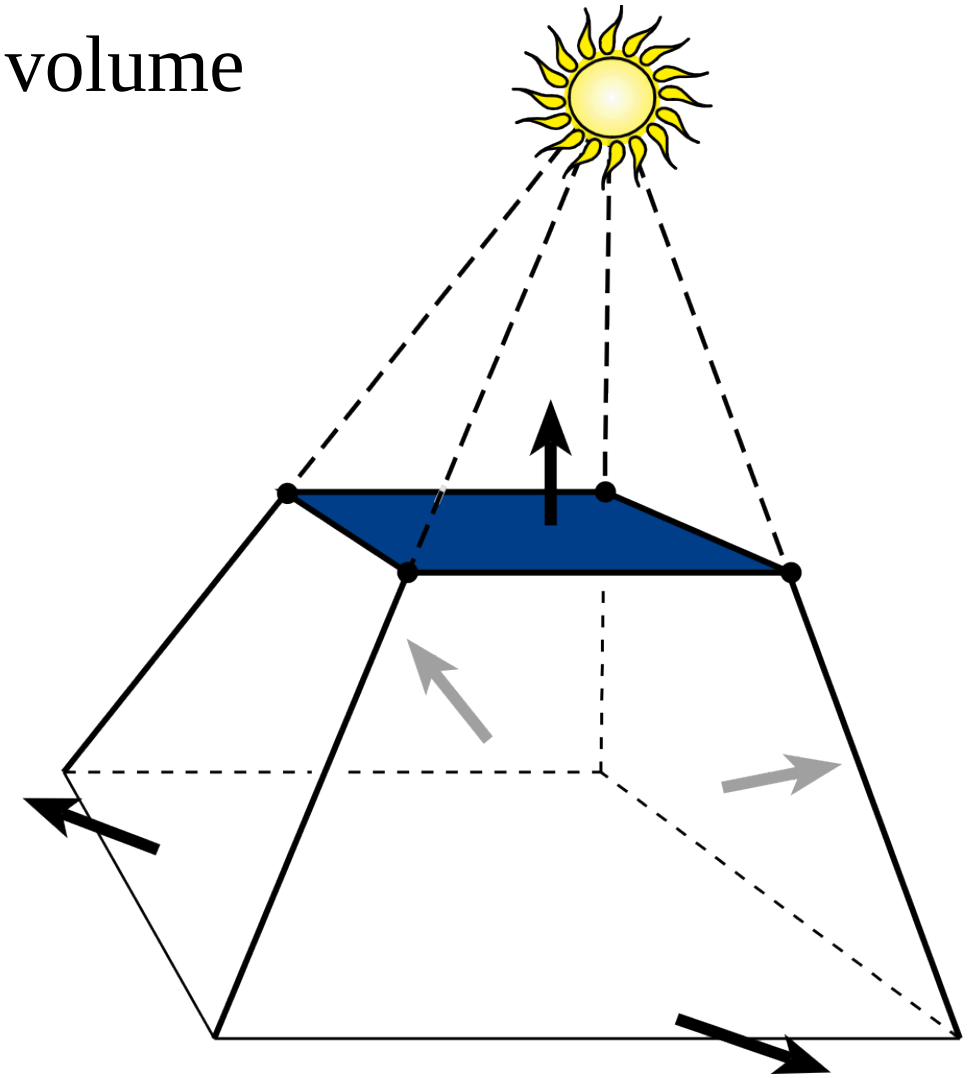
Today

- Shadow Maps
- Shadow Volumes
  - (The Stencil Buffer)
- Deep Shadow Maps
- Alias-free Shadow Maps



Shadow Volumes

- Explicitly represent the volume of space in shadow
- For each polygon
  - Pyramid with point light as apex
  - Include polygon to cap

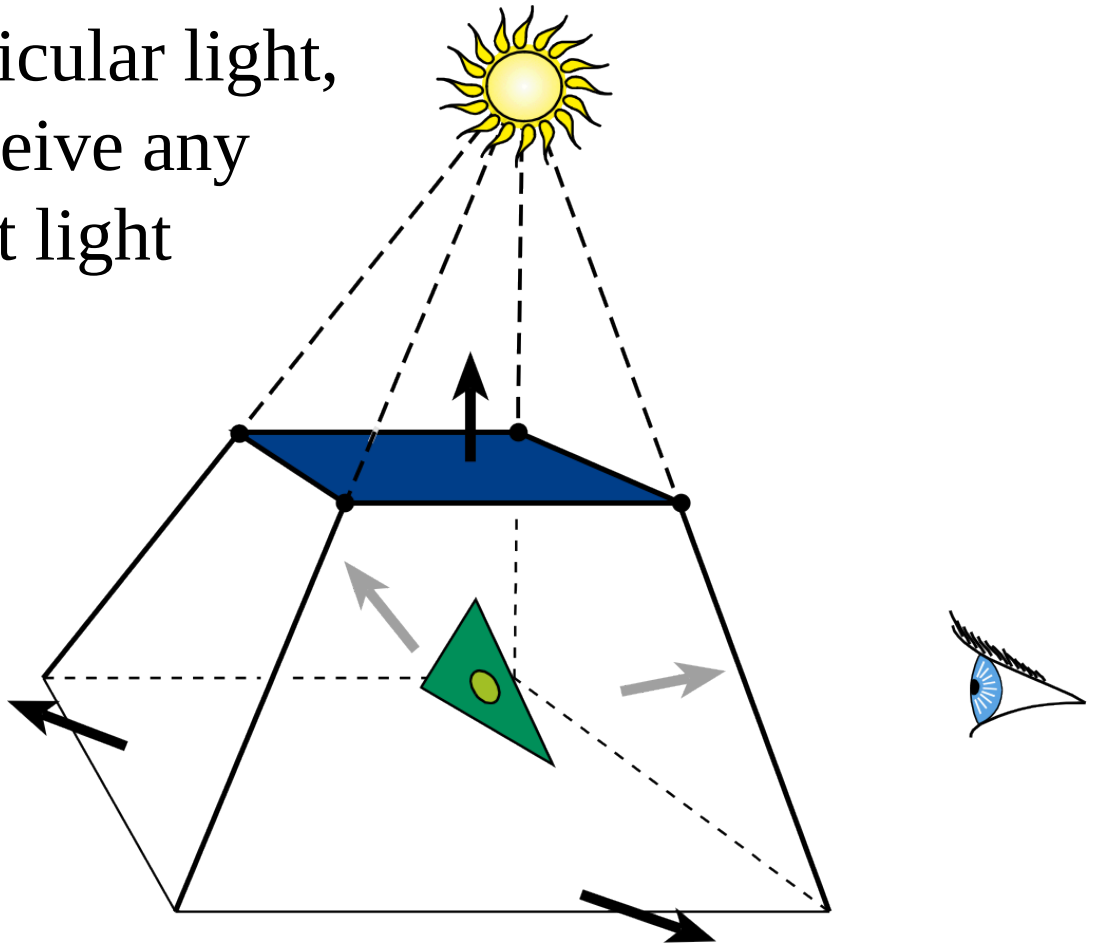


Shadow Volumes

- If a point is inside a shadow volume cast by a particular light, the point does not receive any illumination from that light

- Cost of naive implementation:

  #polygons * #lights



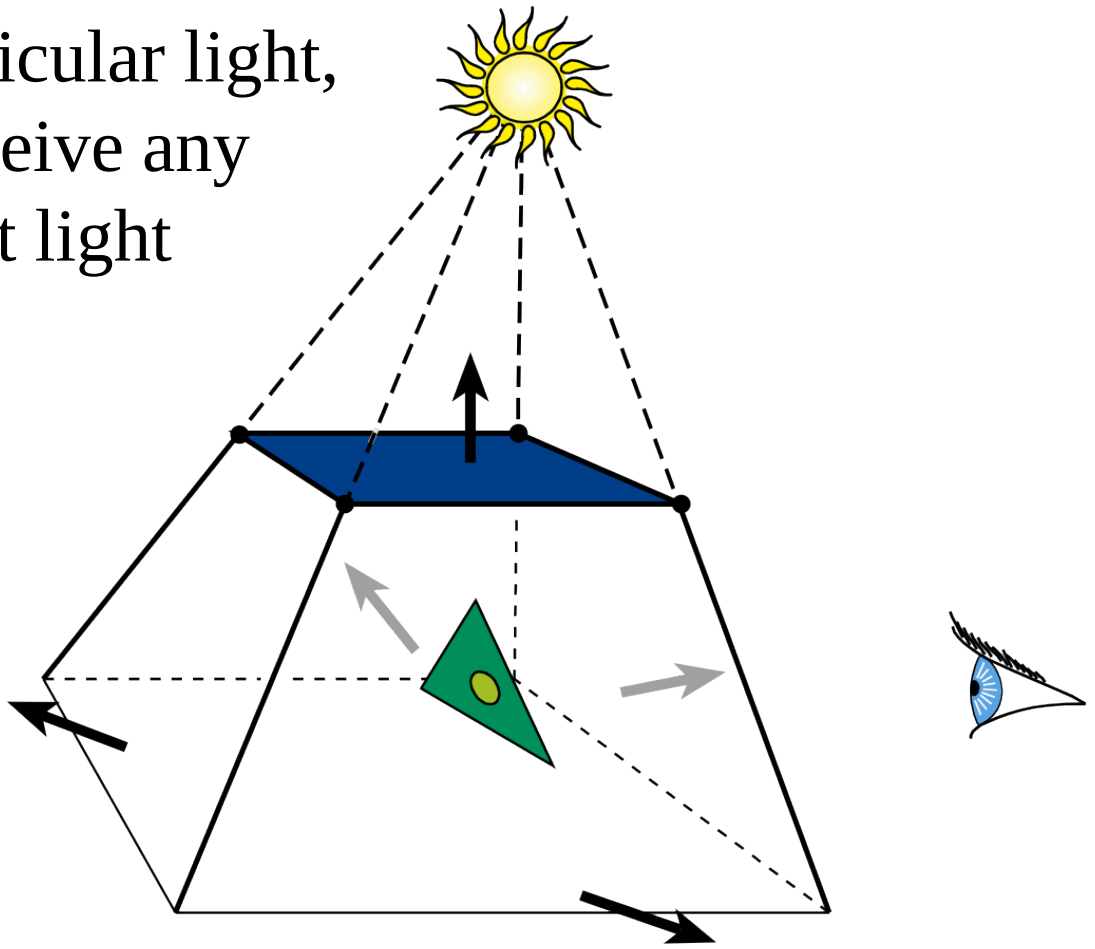
Shadow Volumes

Questions?

- If a point is inside a shadow volume cast by a particular light, the point does not receive any illumination from that light

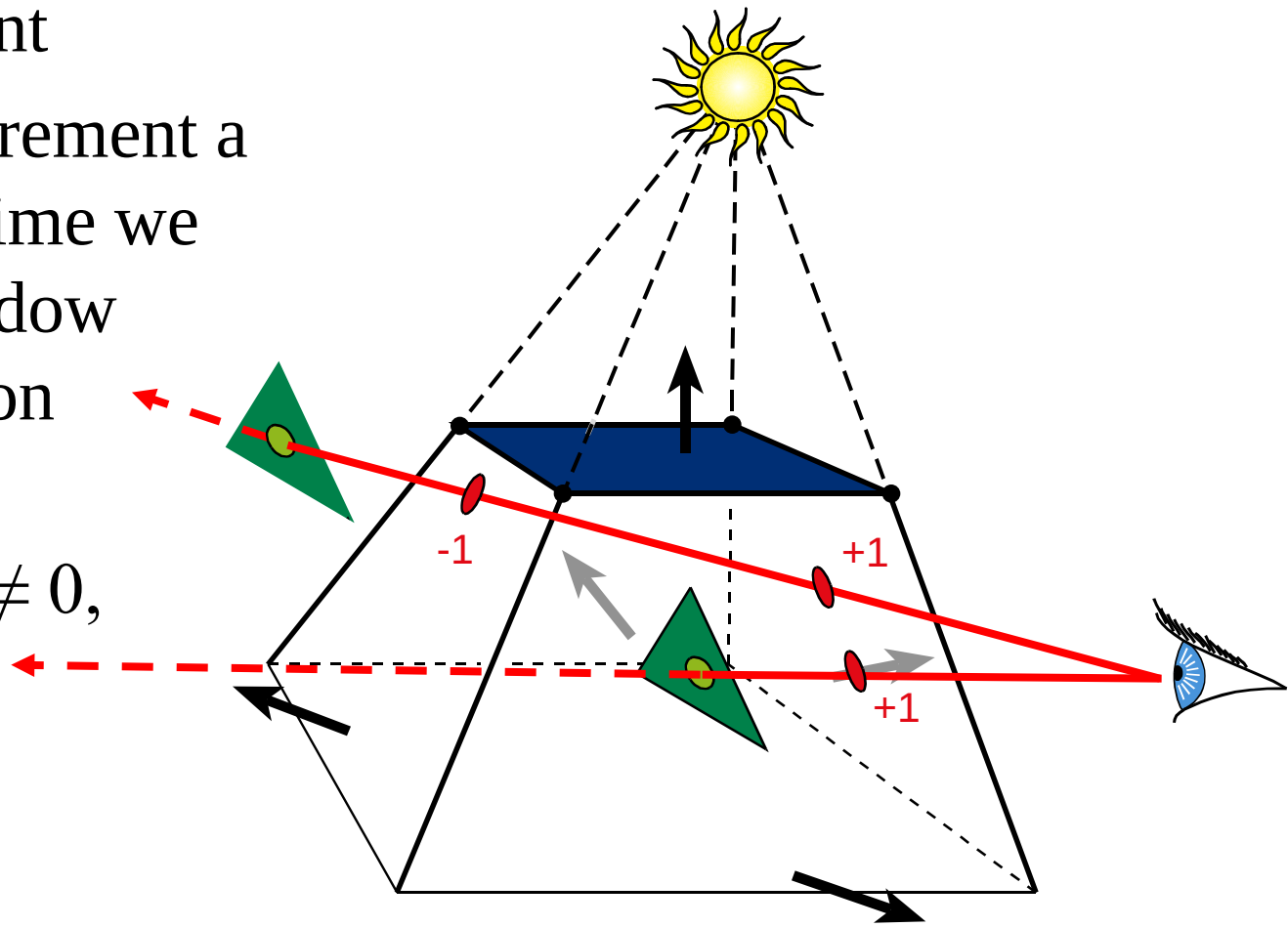
- Cost of naive implementation:

  #polygons * #lights



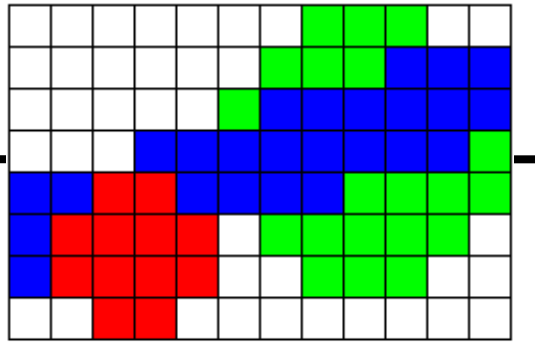
Better Shadow Volumes

- Shoot a ray from the eye to the visible point
- Increment/decrement a counter each time we intersect a shadow volume polygon
- If the counter ≠ 0, the point is in shadow

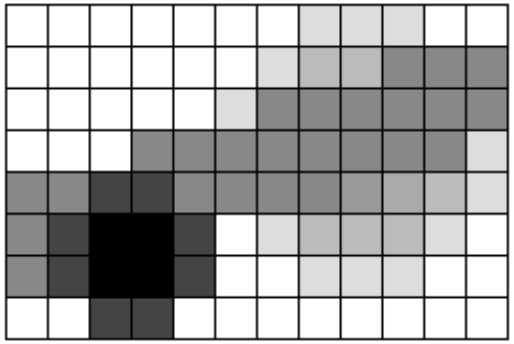


Stencil Buffer

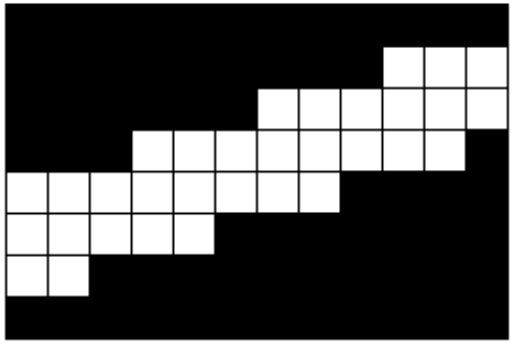
- Tag pixels in one rendering pass to control their update in subsequent rendering passes
  - "For all pixels in the frame buffer" → "For all *tagged* pixels in the frame buffer"
- Can specify different rendering operations for each case:
  - stencil test fails
  - stencil test passes & depth test fails
  - stencil test passes & depth test passes



frame buffer



depth buffer



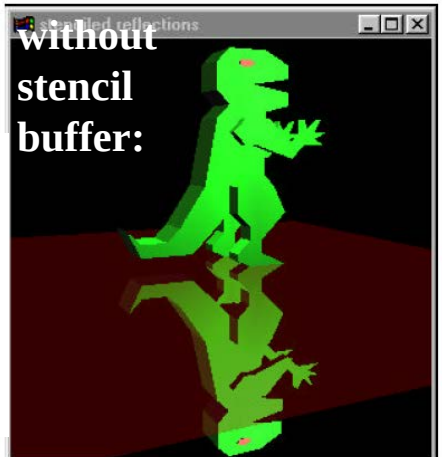
stencil buffer

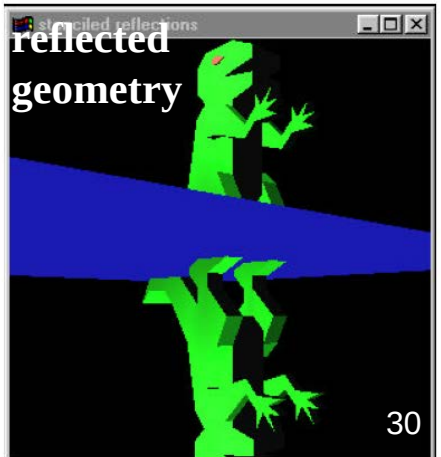
Stencil Buffer – Real-time Mirror

- Clear frame, depth & stencil buffers
- Draw all non-mirror geometry to frame & depth buffers
- Draw mirror to stencil buffer, where depth buffer passes
- Set depth to infinity, where stencil buffer passes
- Draw reflected geometry to frame & depth buffer, where stencil buffer passes

See NVIDIA's stencil buffer tutorial
http://developer.nvidia.com

also discusses blending, multiple mirrors, objects behind mirror, etc…





Shadow Volumes w/ the Stencil Buffer

Initialize stencil buffer to 0

Draw scene with ambient light only

Turn off frame buffer & z-buffer updates

Draw front-facing shadow polygons

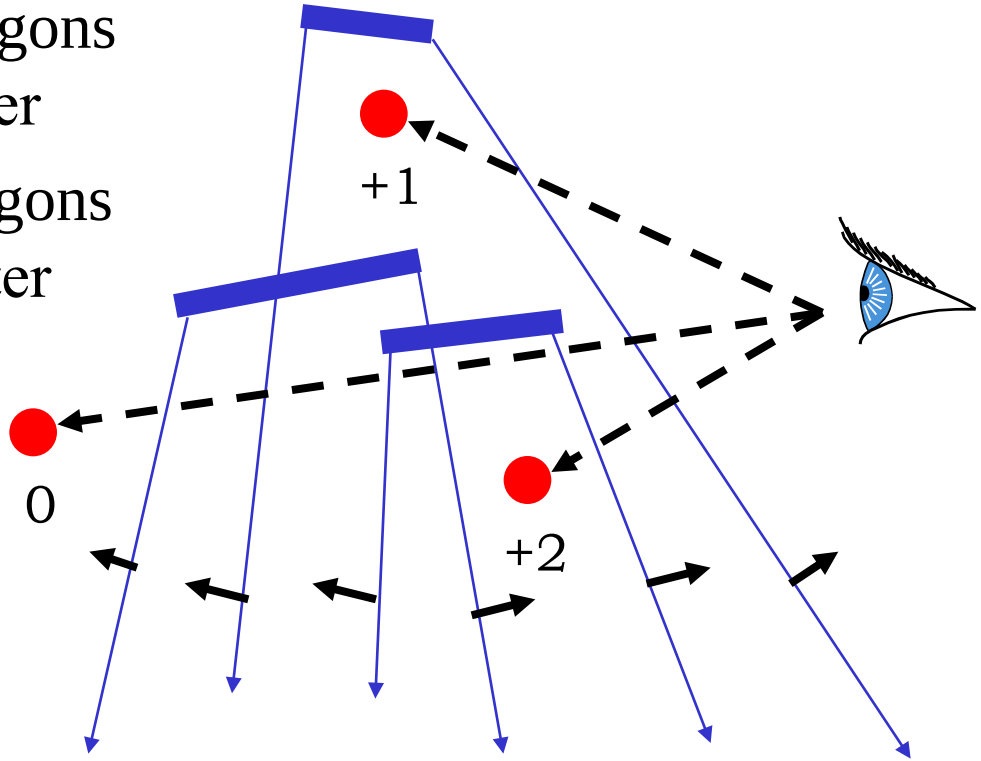
If z-pass → increment counter

Draw back-facing shadow polygons

If z-pass → decrement counter

Turn on frame buffer updates

Turn on lighting and redraw pixels with counter = 0



Shadow Volumes w/ the Stencil Buffer

Initialize stencil buffer to 0

Draw scene with ambient light only

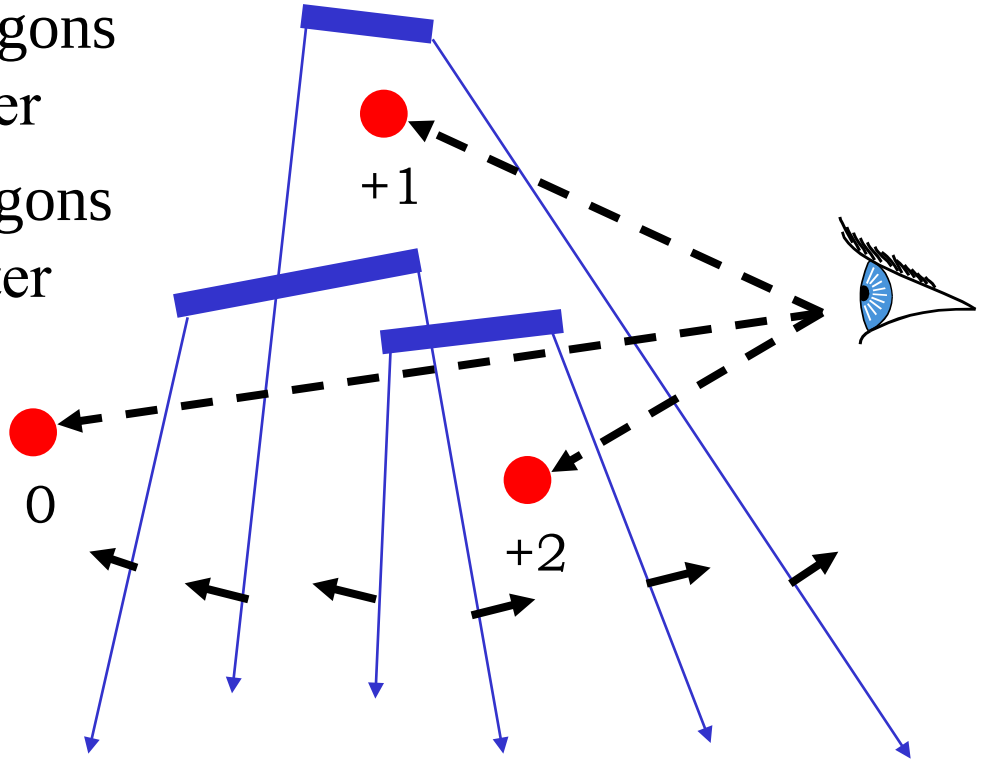
Turn off frame buffer & z-buffer updates

Draw front-facing shadow polygons
    If z-pass → increment counter

Draw back-facing shadow polygons
    If z-pass → decrement counter

Turn on frame buffer updates

Turn on lighting and
    redraw pixels with
    counter = 0

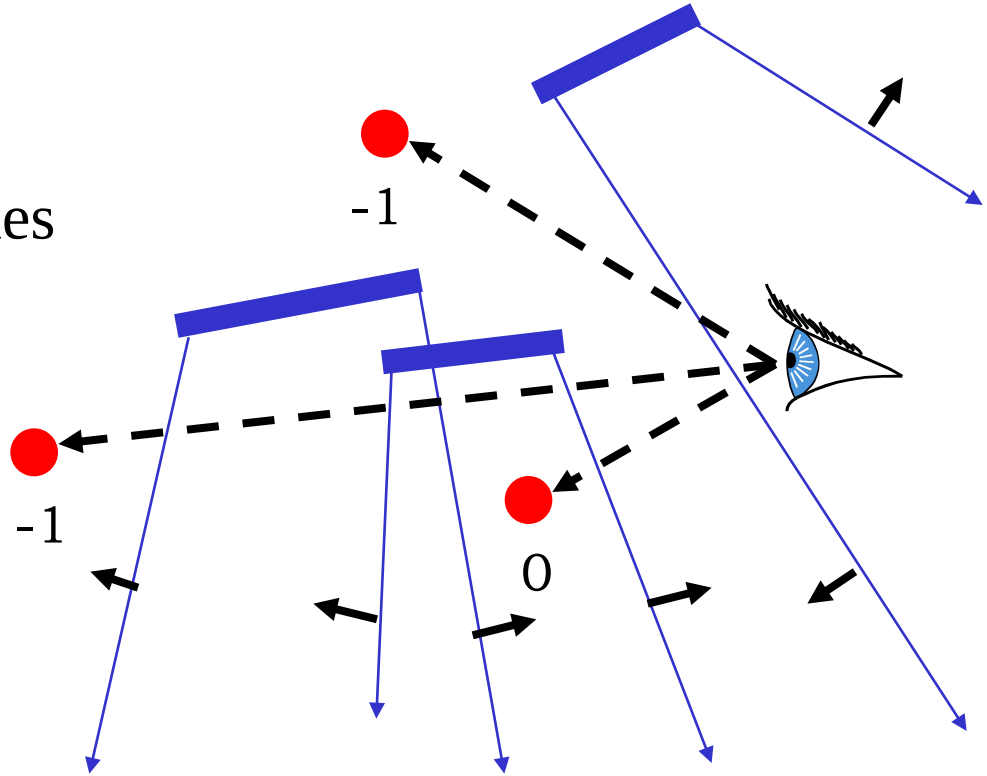


Questions?

If the Eye is in Shadow...

- ... then a counter of 0 does not necessarily mean lit

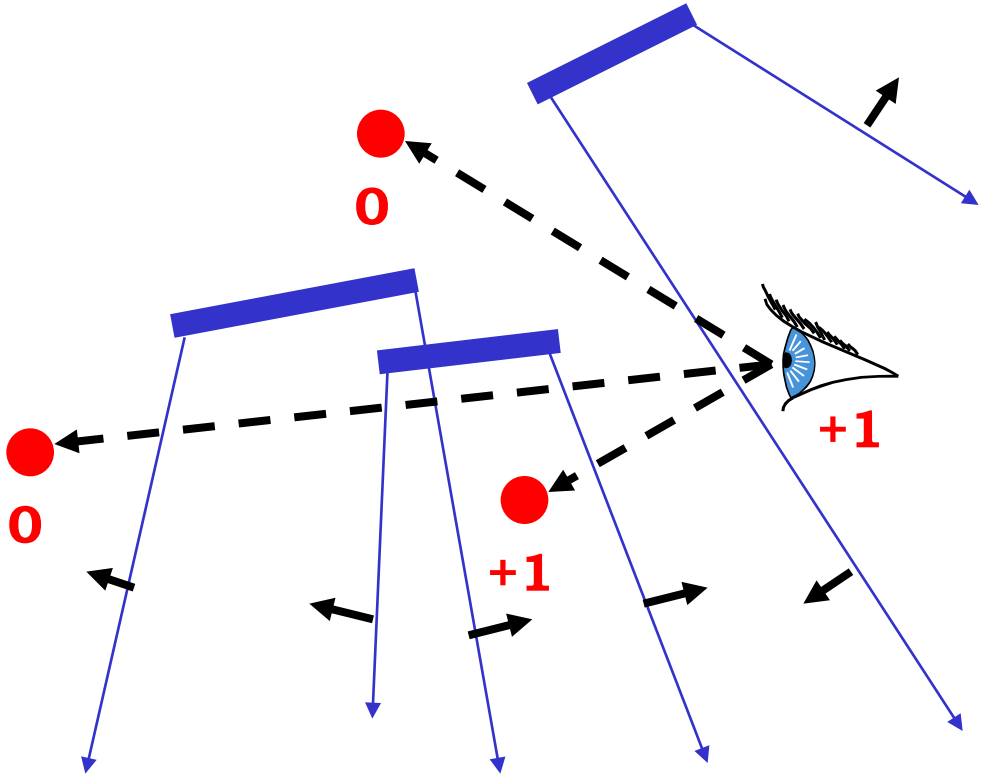
- 3 Possible Solutions:
  1. Explicitly test eye point with respect to all shadow volumes
  2. Clip the shadow volumes to the view frustum
  3. "Z-Fail" shadow volumes



1. Test Eye with Respect to Volumes

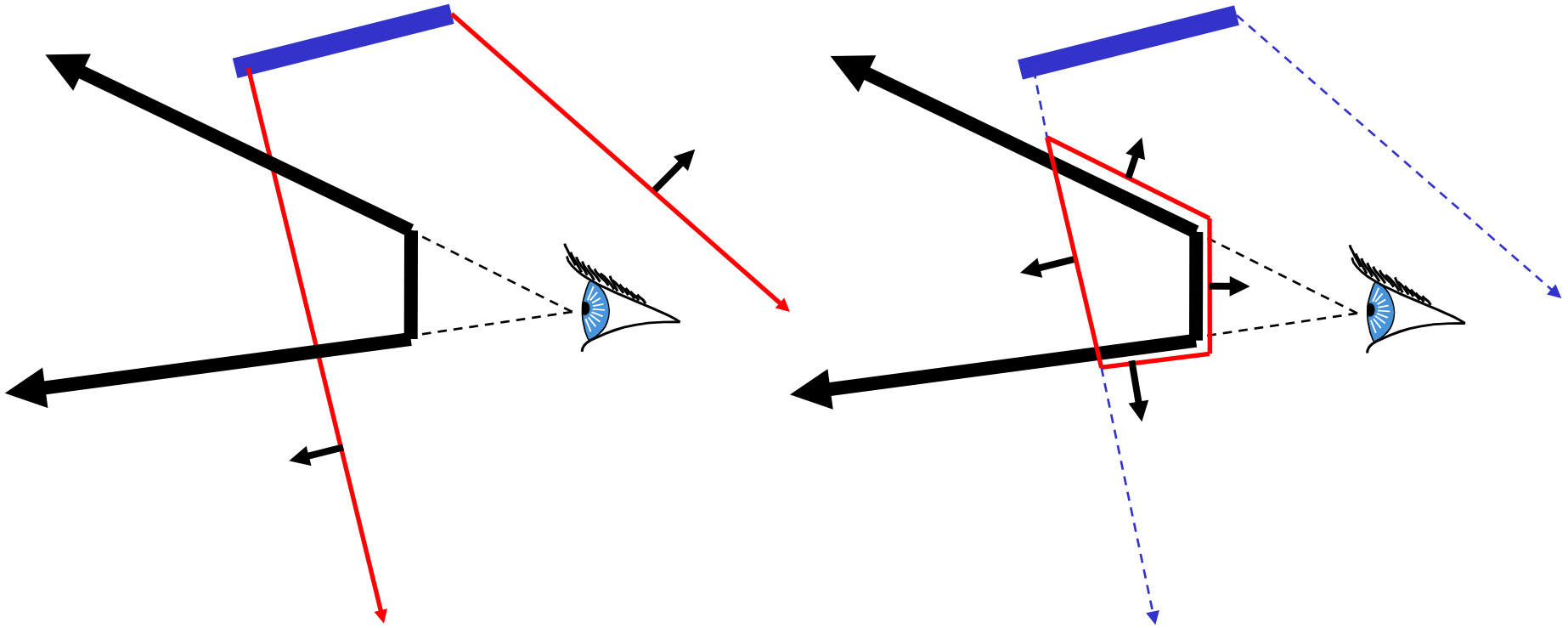
- Adjust initial counter value

Expensive



2. Clip the Shadow Volumes

- Clip the shadow volumes to the view frustum and include these new polygons
- *Messy CSG*



3. "Z-Fail" Shadow Volumes

Start at infinity

...

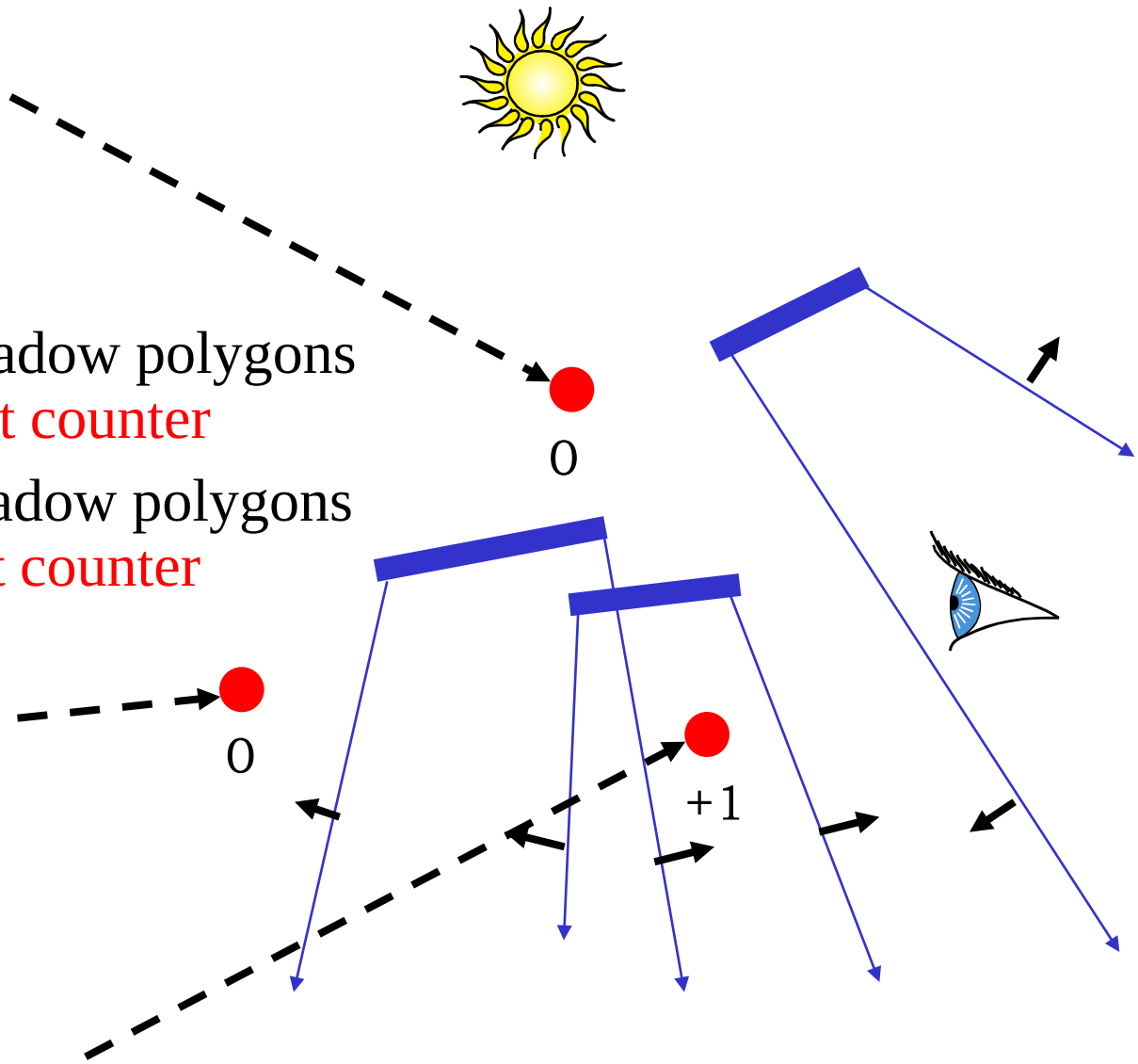
Draw front-facing shadow polygons

If z-fail, decrement counter

Draw back-facing shadow polygons

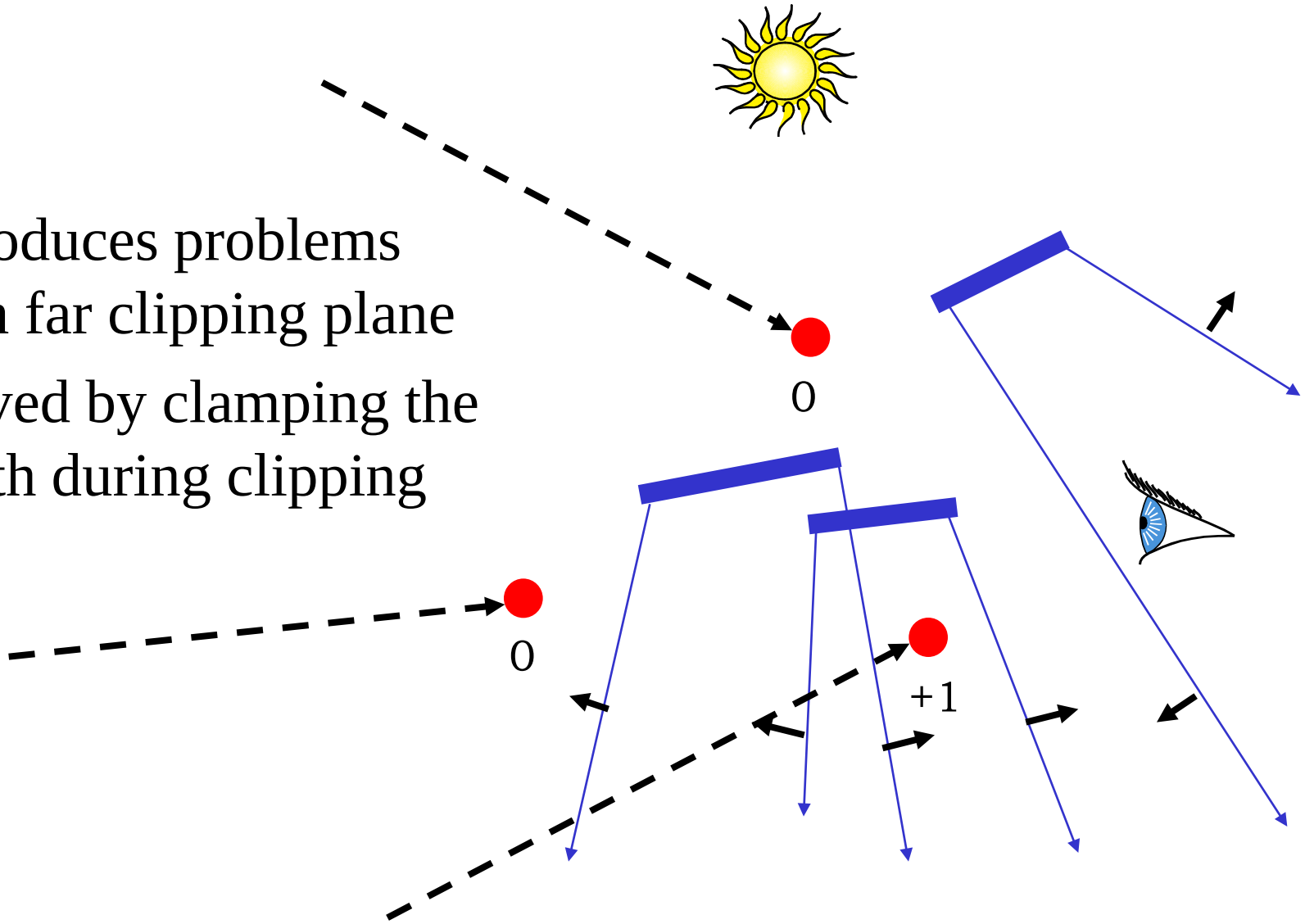
If z-fail, increment counter

...



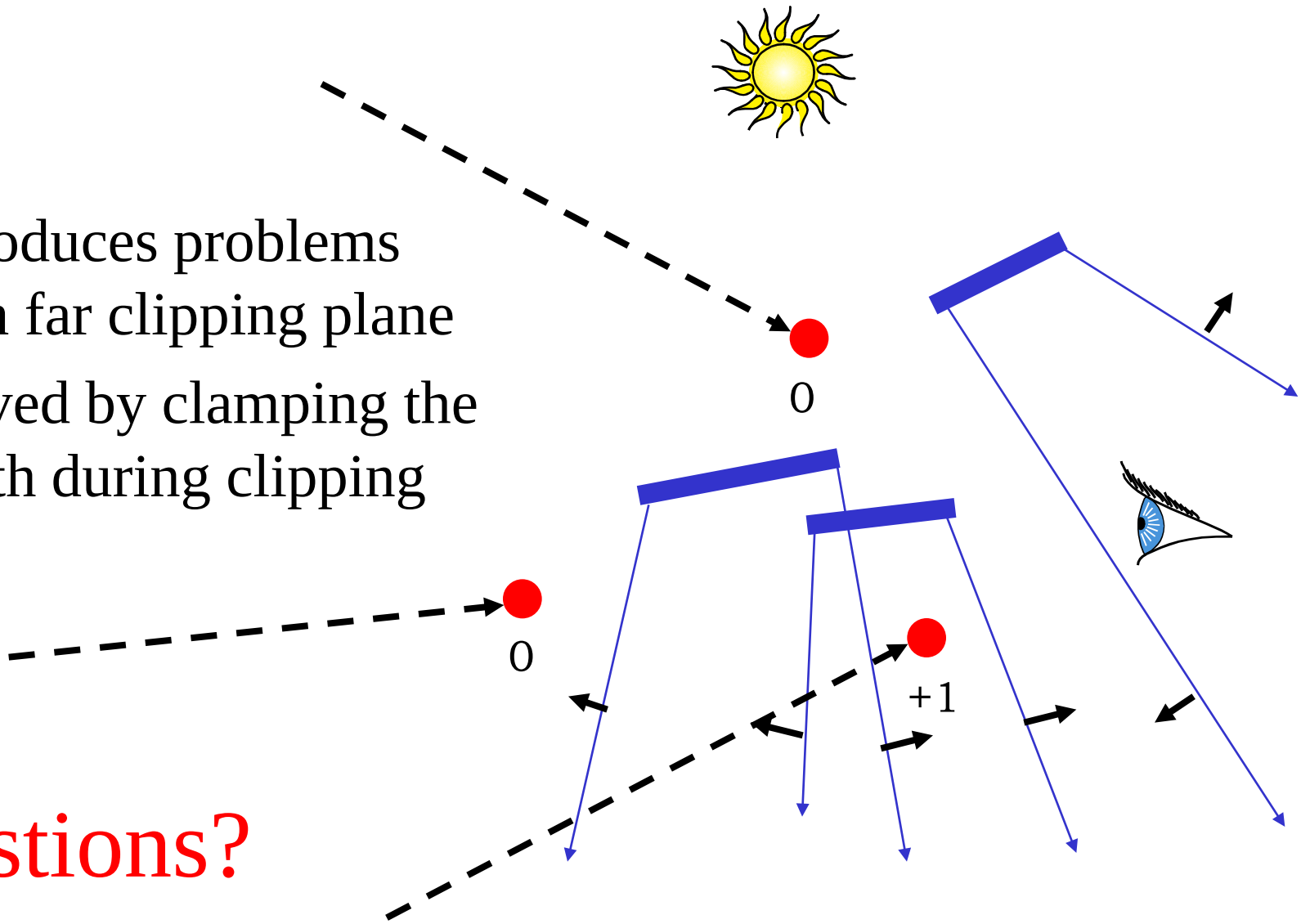
3. "Z-Fail" Shadow Volumes

- Introduces problems with far clipping plane
- Solved by clamping the depth during clipping



3. "Z-Fail" Shadow Volumes

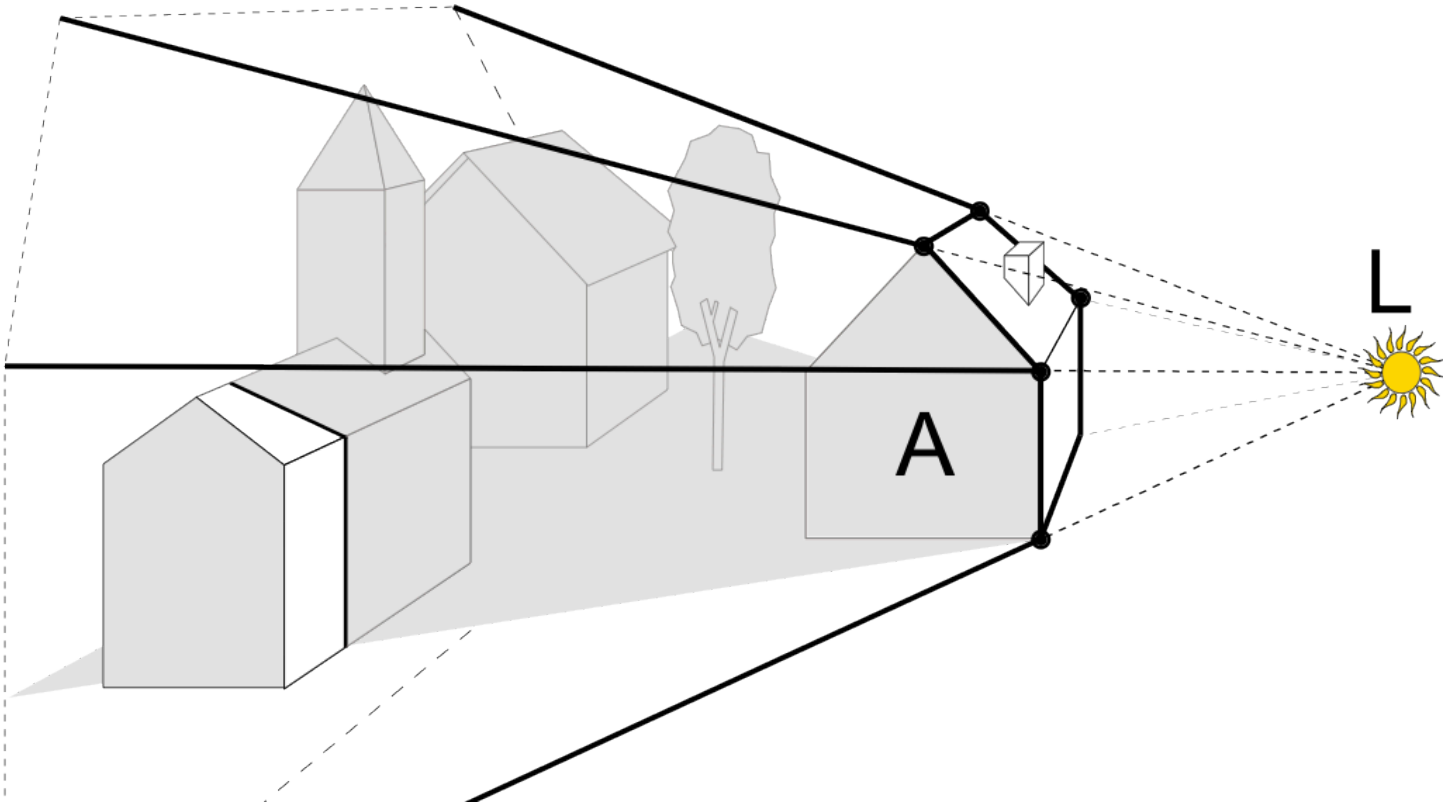
- Introduces problems with far clipping plane
- Solved by clamping the depth during clipping



Questions?

Optimizing Shadow Volumes

- Use silhouette edges only (edge where a back-facing & front-facing polygon meet)



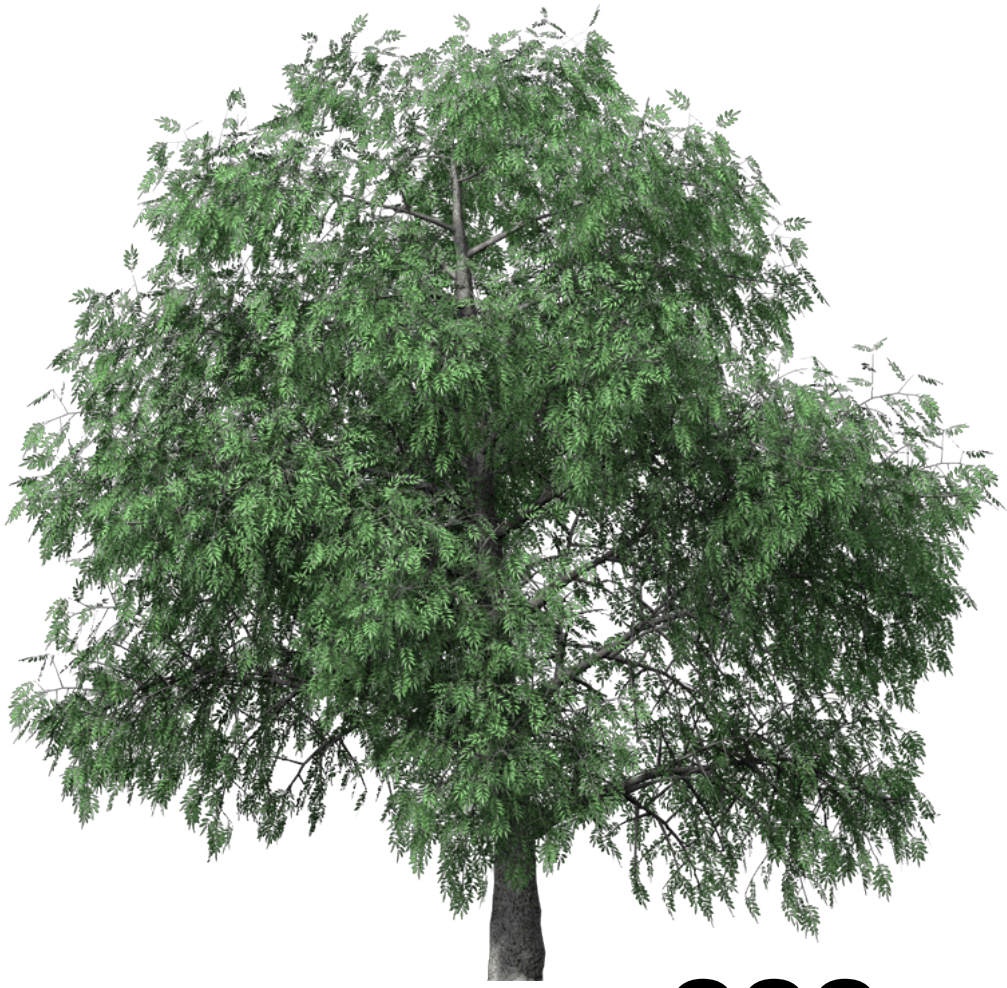
Shadow Volumes Are Sort of Passé

- It is possible to implement shadow volumes rather efficiently using the stencil buffer
  - John Carmack's Doom 3 engine did this very well
  - Shadow volumes had their 15 minutes of popularity around that time
  - NVIDIA even came up with hardware acceleration ("UltraShadow")
- Further info
  - Wikipedia
  - Aila & Akenine-Möller: Hierarchical Shadow Volumes, Proc. Graphics Hardware 2004

Id Software

Shadow Volumes Are Sort of Passé

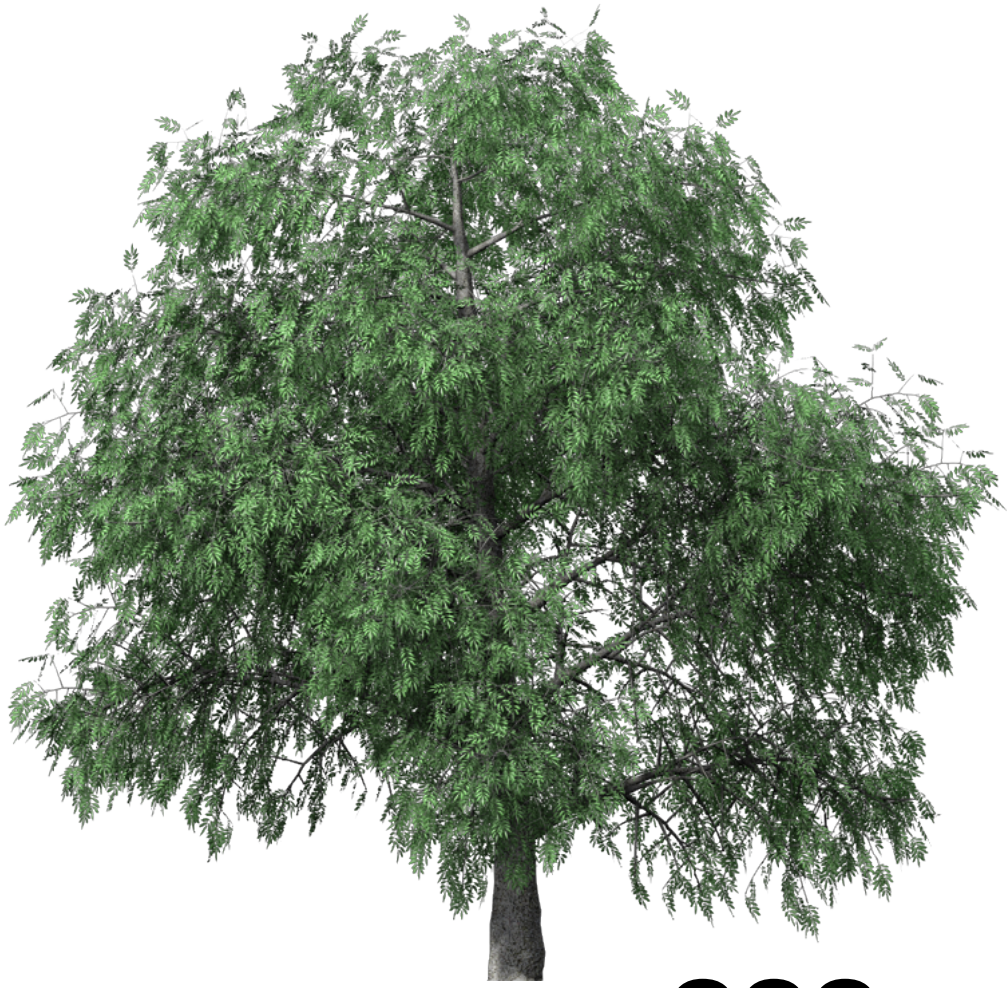
- The need to extract silhouette edges and rasterize the resulting shadow polygons is just too much work when scene complexities and image resolutions grow
- Still, such use of the stencil buffer can be useful for other tricks



???

Shadow Volumes Are Sort of Passé

- The need to extract silhouette edges and rasterize the resulting shadow polygons is just too much work when scene complexities and image resolutions grow
- Shadow maps are bad, but they're the best we've got!

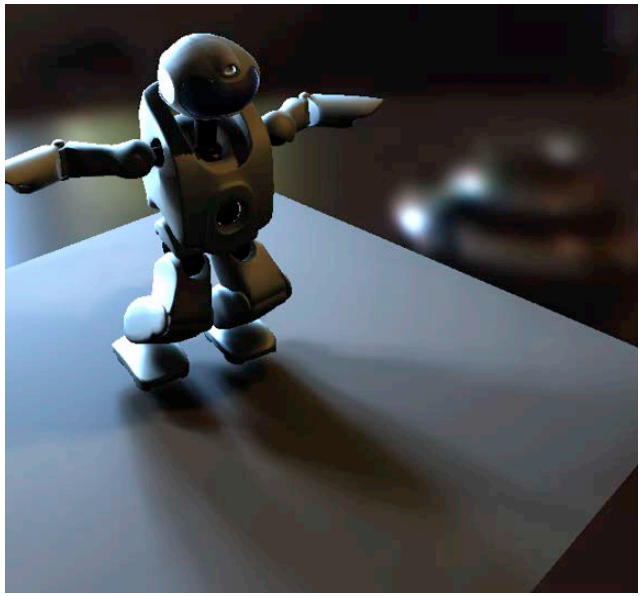


???

Further Reading on Shadows

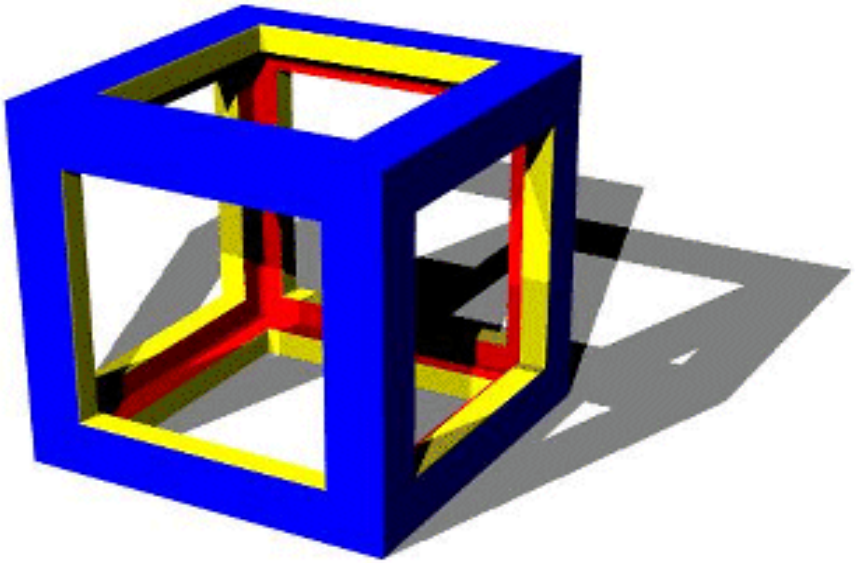
- Some recent techniques allow filtering the shadow map instead of just the tests (percentage closer)
  - Works pretty well in some situations, although not general
  - Variance Shadow Maps (Donnelly, Lauritzen I3D 2006)
  - Convolution Shadow Maps
    - Annen et al., SIGGRAPH 2008
    - Annen et al., EGSR 2007
- An interesting hybrid between shadow maps and ray tracing
  - Aila and Laine, Alias-Free Shadow Maps, EGSR 2004

Annen et al. 2008



Today

- Shadow Maps
- Shadow Volumes
  - (The Stencil Buffer)
- Deep Shadow Maps
- Alias-free Shadow Maps

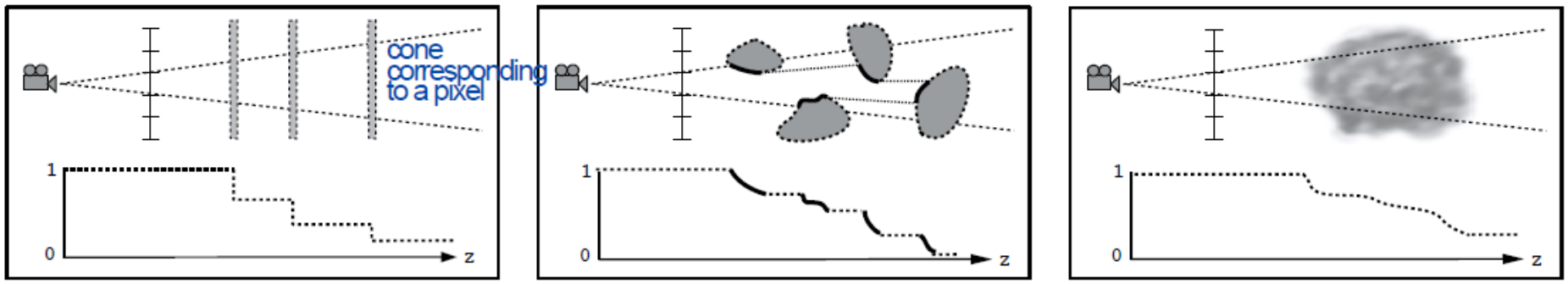


Deep shadow maps

- Lokovic & Veach, Pixar
- Shadows in participating media like smoke, inside hair, etc.
  - They represent not just depth of the first occluding surface, but the attenuation along the light rays
- Note: shadowing only, no scattering

Visibility function along depth

- Fraction of a pixel occluded, as a function of depth
- Due to
  - small occluders, semi-transparent objects, smoke & volumetric effects



(a) A stack of semitransparent objects *(b) Partial coverage by opaque blockers* *(c) Volume attenuation due to smoke*

Figure 3: *Visibility functions in flatland. Each diagram shows a beam of light that starts at the shadow camera origin (*i.e. *the light source) and passes through a single pixel of the deep shadow map, accompanied by that pixel's visibility function.* ***(a)*** *The beam's power is reduced as it passes through consecutive semitransparent surfaces.* ***(b)*** *The blockers are opaque, but each covers only part of the pixel's area; the emphasized segments of the function correspond to visible portions of the blockers.* ***(c)*** *Passage through smoke reduces the beam's power in a more continuous manner.*

Deep shadow maps

- Preprocess:
  Compute dense visibility function for each pixel of shadow map
  - send lots of rays or rasterize at a high resolution
  - compress visibility function at each pixel
    - to reduce memory cost
- At render time, shadow query
  - Transform visible point into light coordinates
    - just like shadow maps
  - Read visibility value from compressed function

Compression algorithm

- Approximation: Piecewise linear
- Set an error bound
- Decide which vertices to keep
- Greedy from zero do far

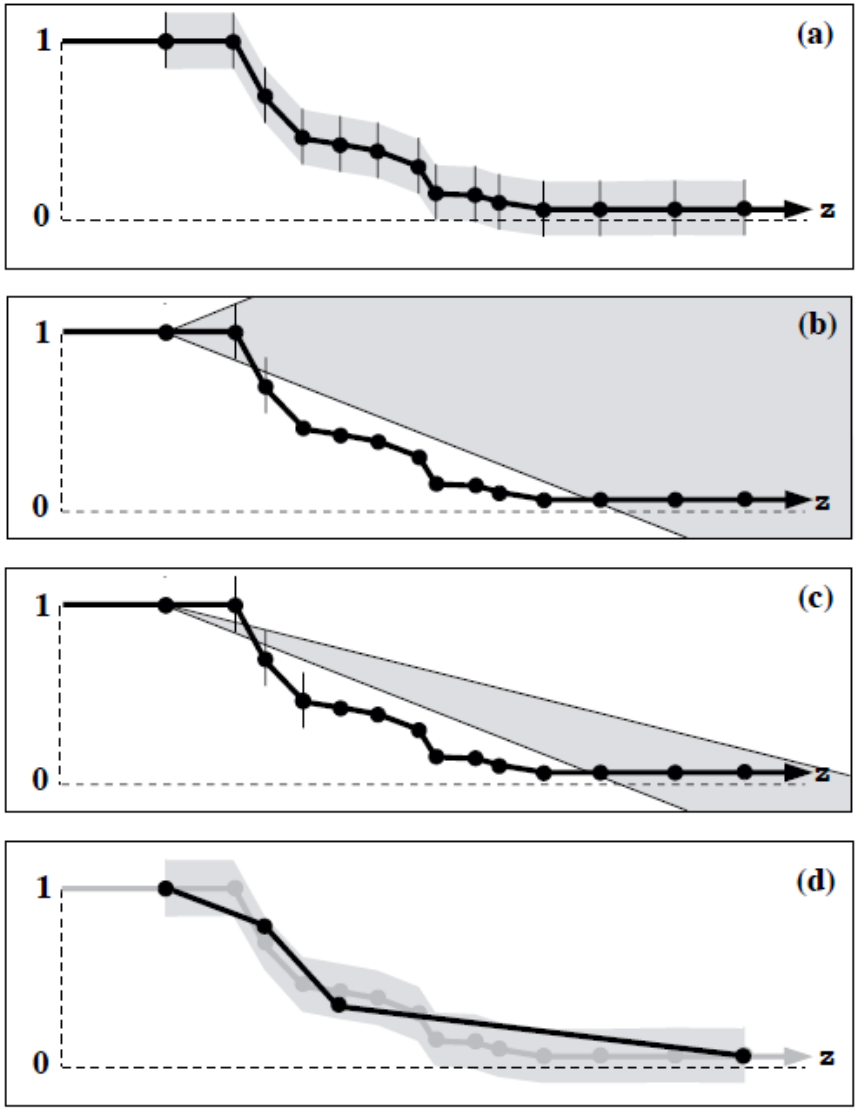


Figure 5: *Our compression algorithm.* ***(a)*** *A piecewise linear curve and an illustration of its error bound.* ***(b)*** *Each input vertex defines a* target window *that constrains the slope of the next output segment.* ***(c)*** *The current slope range is intersected with each target window until it would become empty.* ***(d)*** *The output segment is extended to the current* z *value with a slope equal to the midpoint of the current slope range, and this process is repeated.*

Deep shadow map results

Figure 11: *A cloud with pipes. Notice the shadows cast from surfaces onto volumetric objects and vice versa. A single deep shadow map contains the shadow information for the cloud as well as the pipes.*

Deep shadow map results

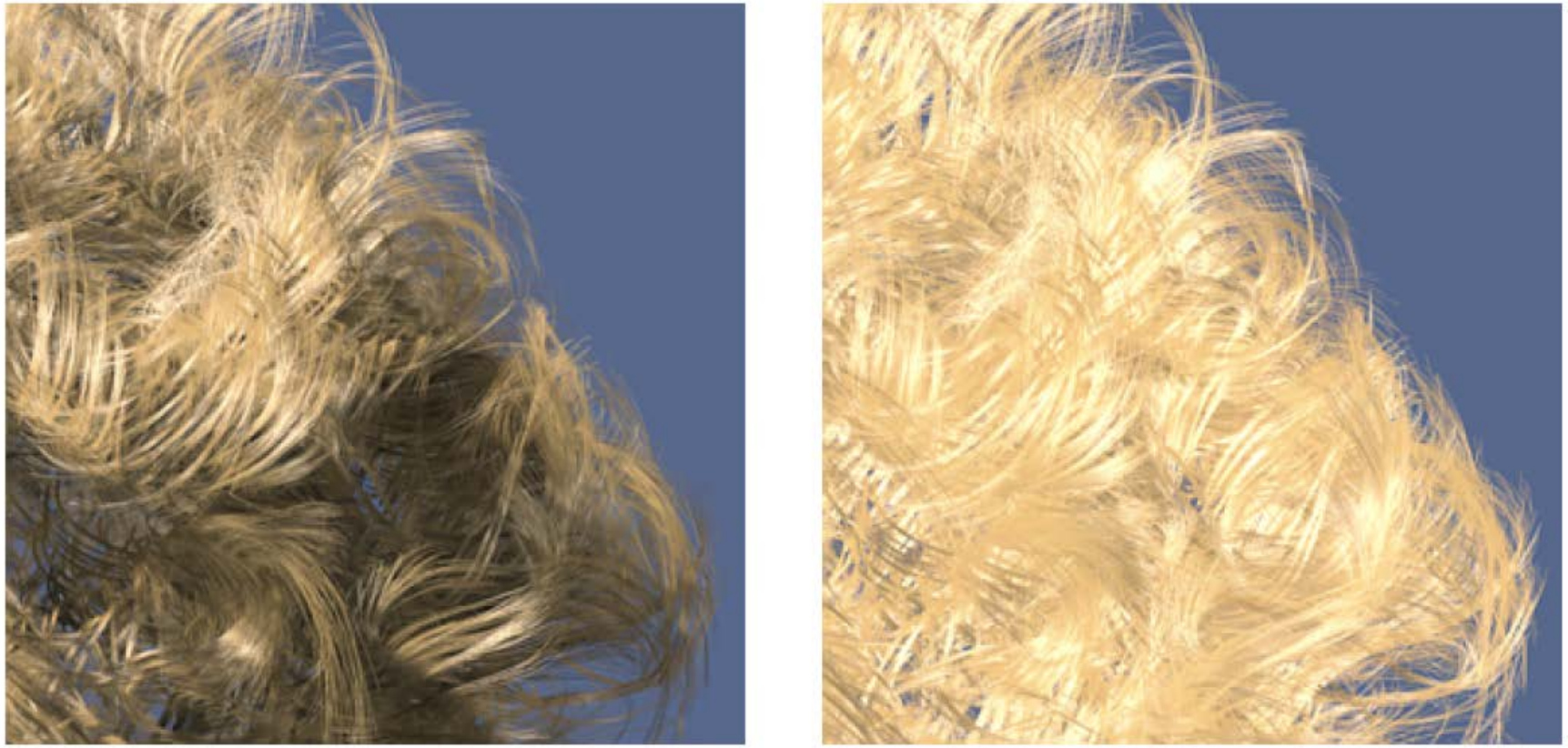


Figure 1: *Hair rendered with and without self-shadowing.*

Deep shadow map results

- Advantage of deep shadow map over higher-resolution normal shadow map:
  Pre-filtering for shadow antialiasing

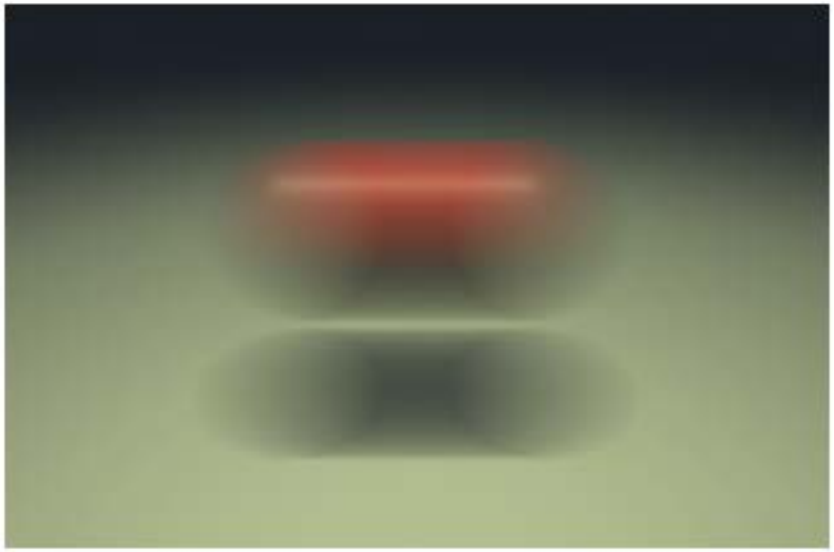
(a) Ball with 50,000 hairs

(b) 512×512 *Normal shadow map*

(c) $4k \times 4k$ *Normal shadow map*

(d) 512×512 *Deep shadow map*

Enables motion blur in shadows



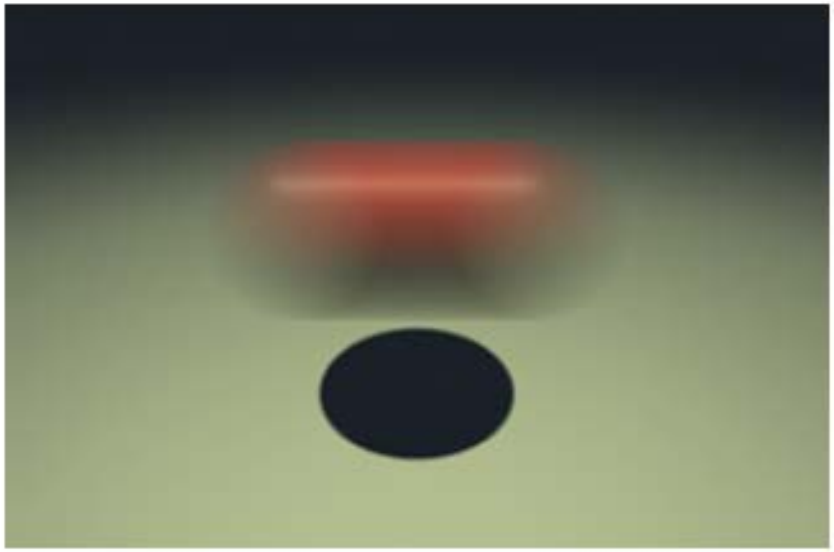
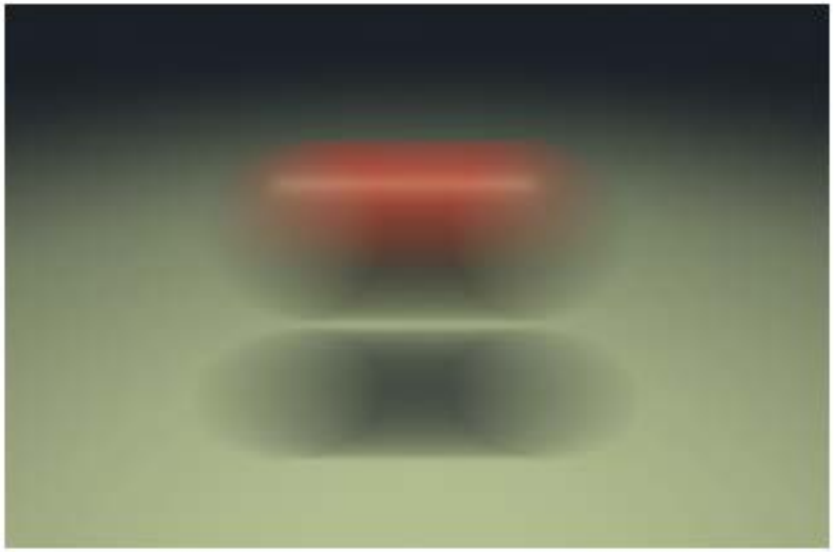


Figure 12: *Rapidly moving sphere with and without motion blur.*

Enables motion blur in shadows



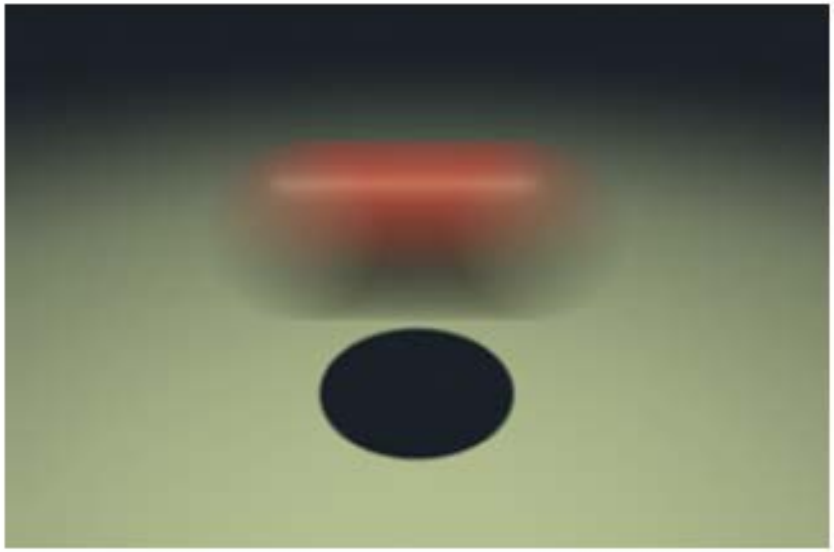
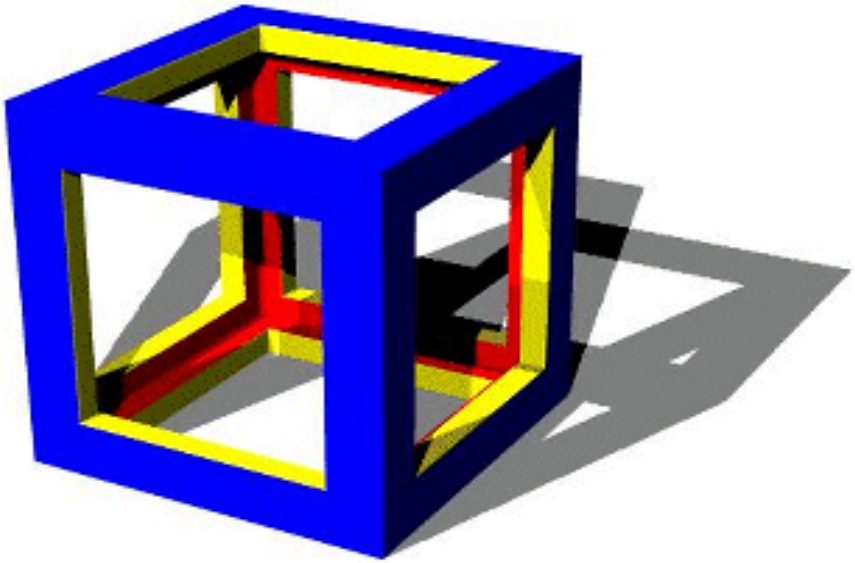


Figure 12: *Rapidly moving sphere with and without motion blur.*

Questions?

Today

- Shadow Maps
- Shadow Volumes
  - (The Stencil Buffer)
- Deep Shadow Maps
- Alias-free Shadow Maps

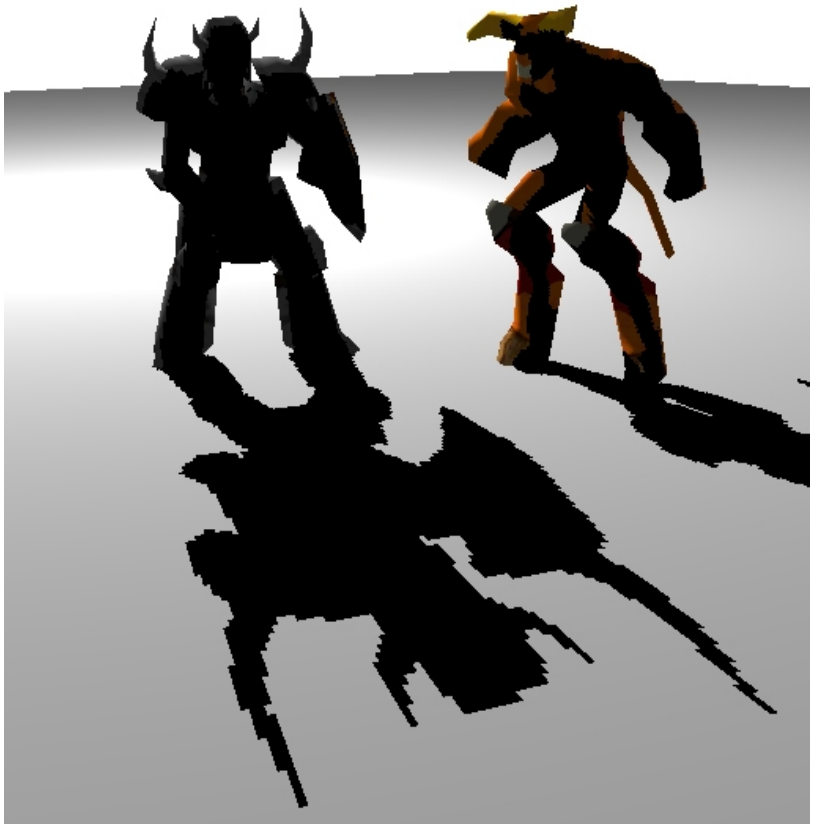


Alias-free shadow maps

- Aila and Laine, http://www.tml.tkk.fi/~timo/
- aka Irregular z-buffer by Johnson et al. http://pl887.pairlitesite.com/papers/tog05-izb/

- Following slides by Aila and Laine

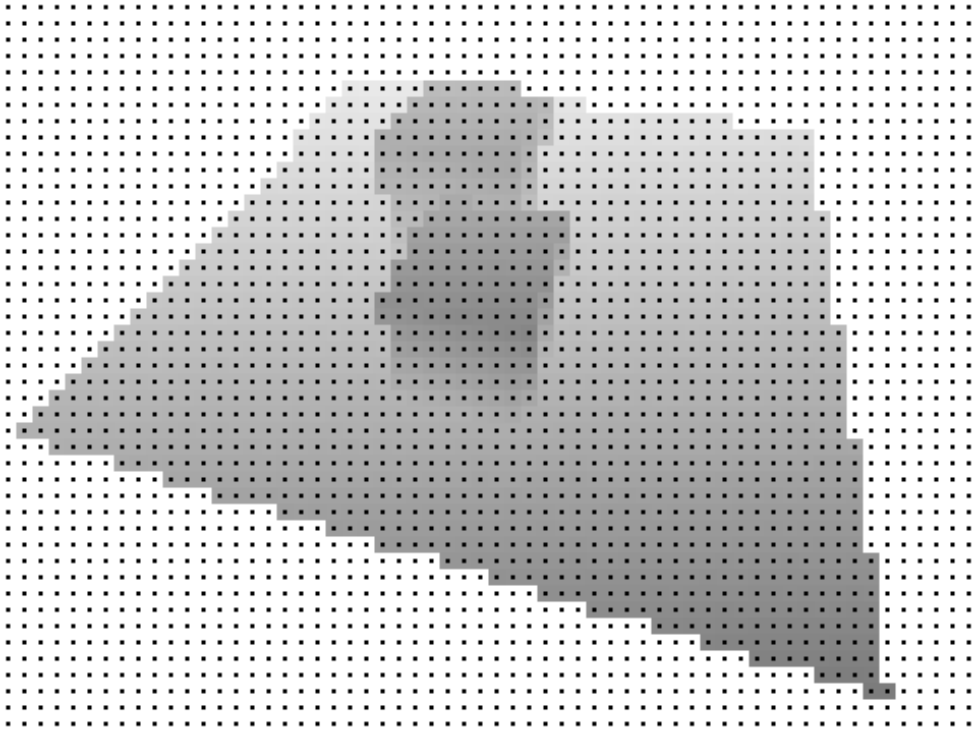
Aliasing: resolution issues



[Sen et al. 2003]

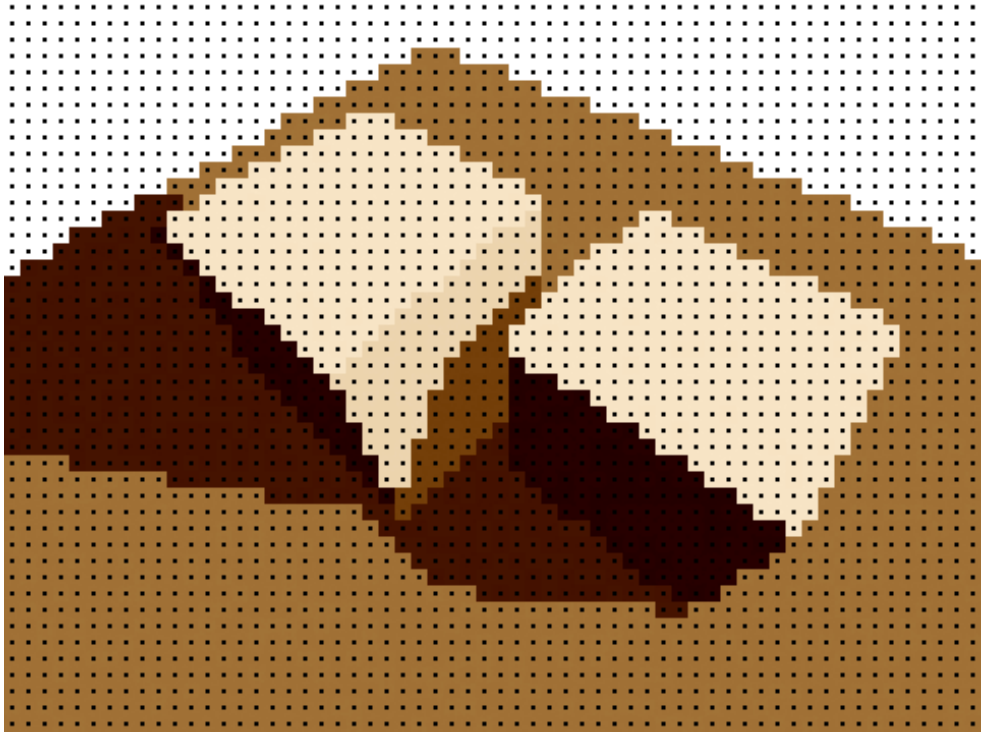
Traditional shadow maps

- Step 1. From light source:
  - Compute a shadow map (z-buffer)
  - Dots are sampling points
  - Depth known only at these points



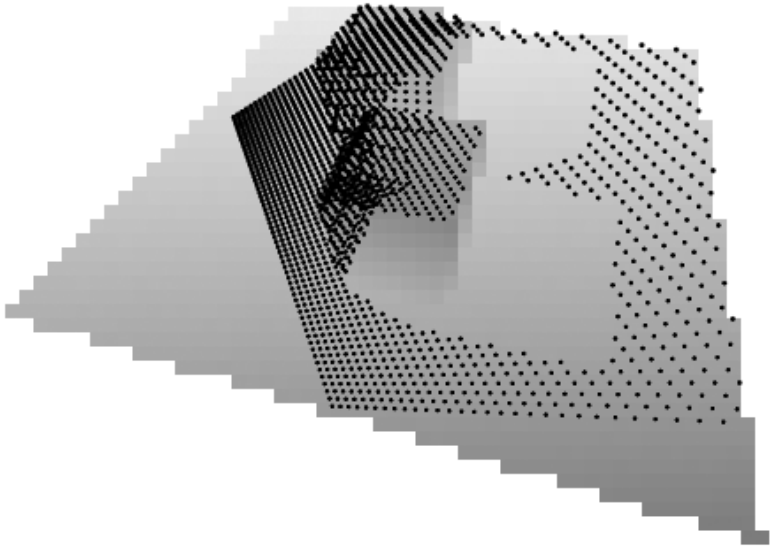
Traditional shadow maps

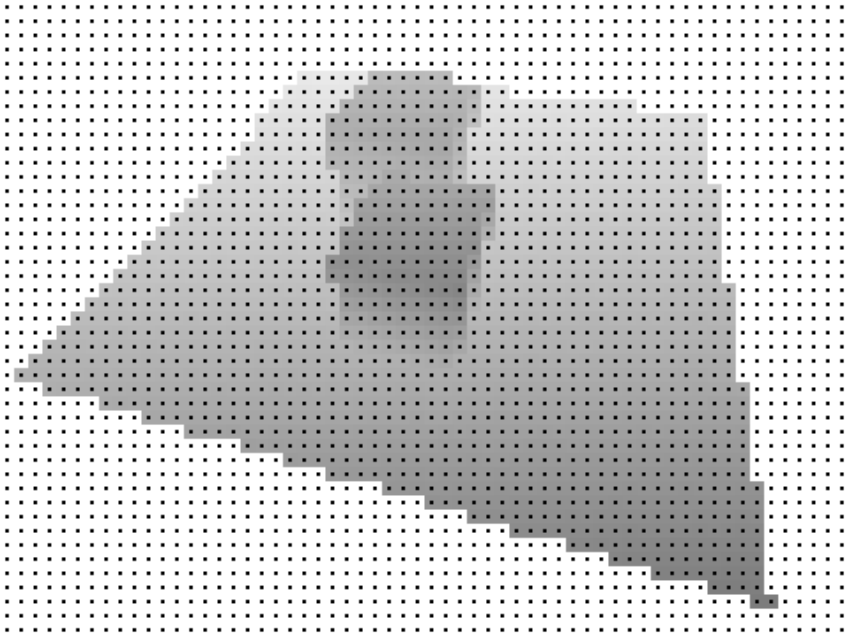
- Step 2. From camera:
  - Samples vs. pixels
  - Project each visible sample to light source
  - Determine shadow term using shadow map



Traditional shadow maps

- Shadow map sampling points





Points where the depth is queried

$\neq$

Points where the depth is computed/known

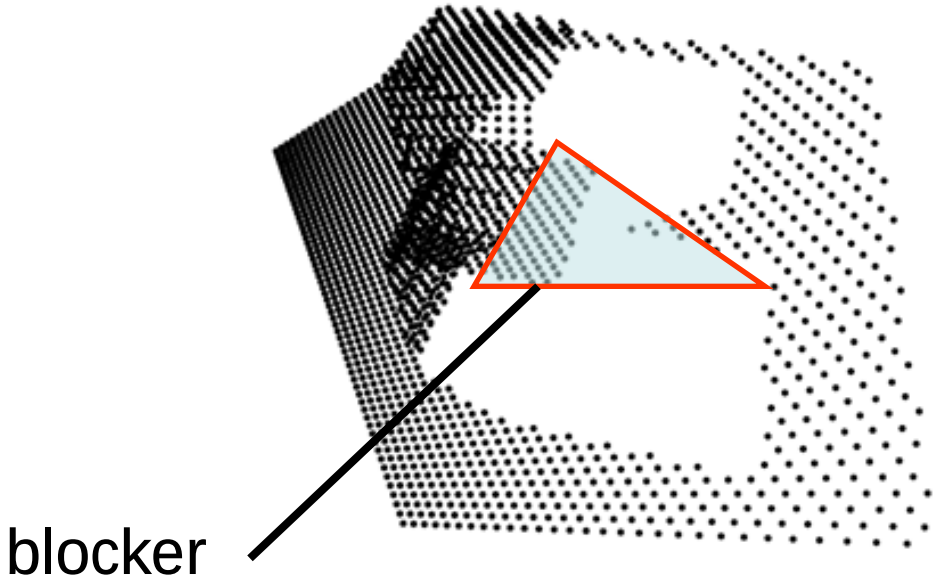
Idea: why not do it right?

1. Project screen-space samples to image plane of light source
2. Rasterize blocker geometry using them as sampling points

- Depth known at correct positions ☺
  - same result as shadow rays

Rasterization

- Q: How to rasterize using irregular sampling points?



Rasterization + depth test

1. Test if sampling point covered
   - we use edge functions [Pineda88]
2. Depth test

Let’s make it practical!

- hierarchical processing of sampling points
- we use axis-aligned 2D BSP

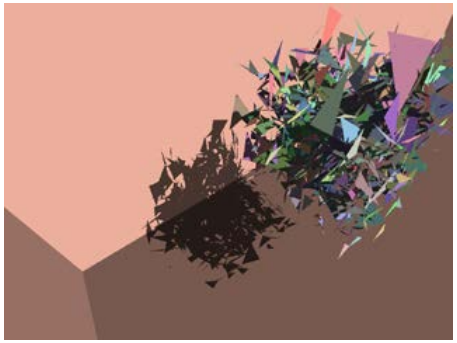
Properties

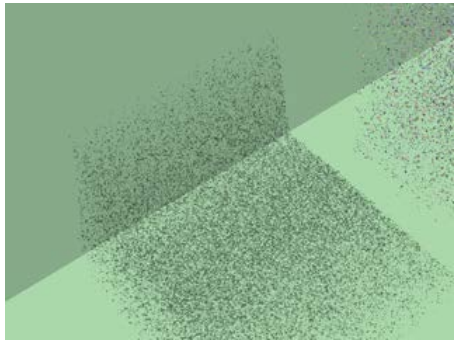
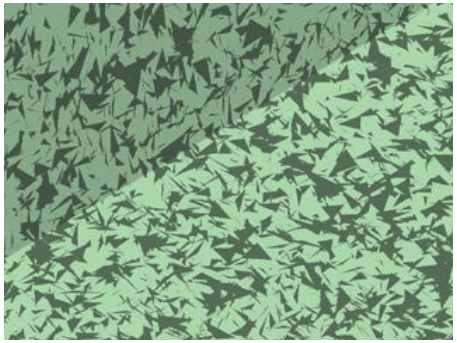
- Resolution issues disappear
- Bias term independent of scene
- Semi-transparent shadow receivers
  - simply transform multiple samples per pixel
- Semi-transparent shadow casters
  - can modulate the RGB color of shadows

Performance (1/2)

- Scalability: random triangles @ 1024x768:
  - transformation of samples: ~130ms
  - BSP construction: ~400ms
  - rasterization:

1K	10K	100K	1M
145ms	381ms	1568ms	8102ms

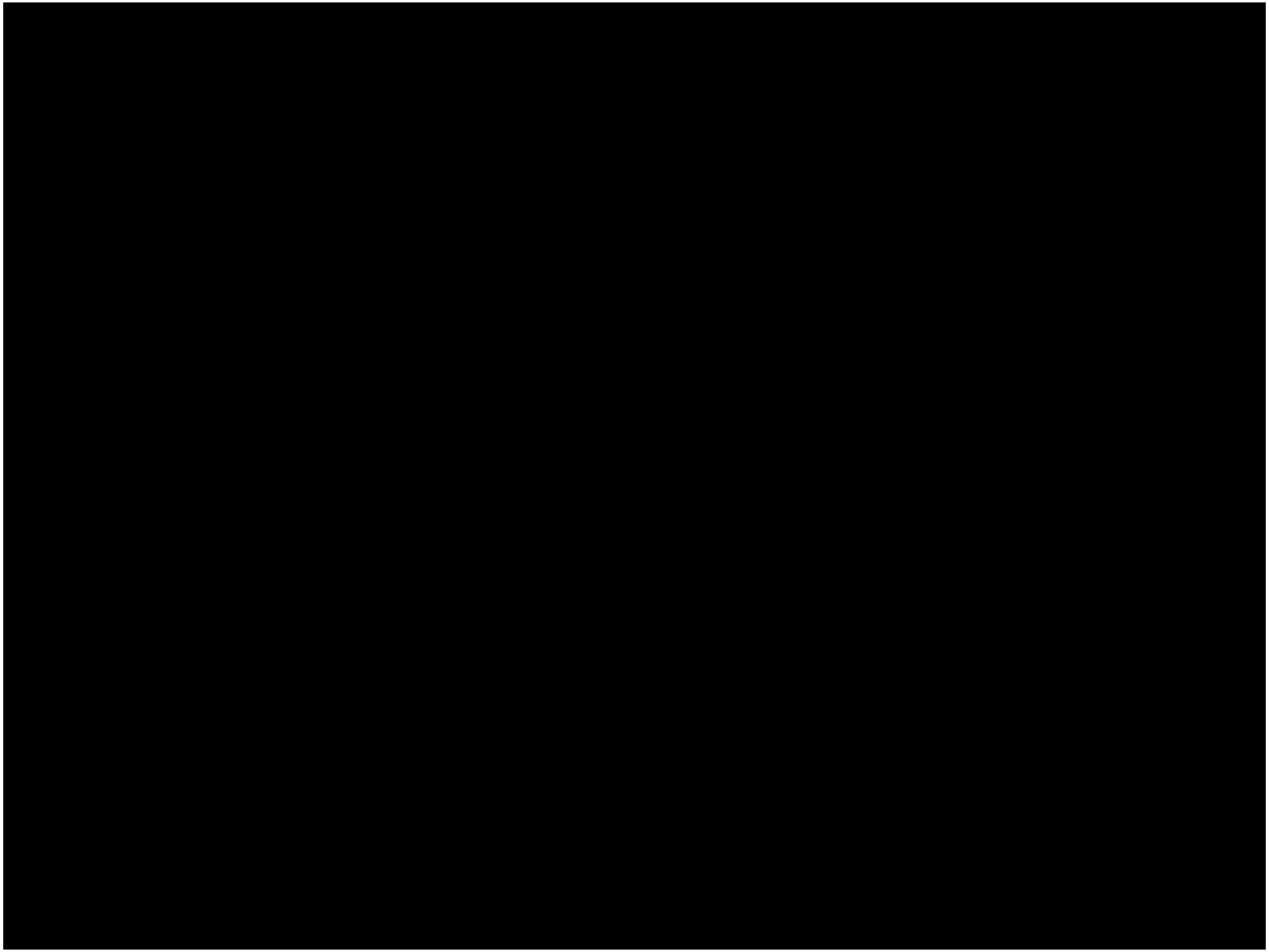


Performance (2/2)

- 2.9M semi-transparent shadow casting tris @ 1536x1088: 12.6s/frame
- Plenty of room for optimizations

Irregular z buffer results



That’s All For Today!

Image: Laine and Aila, EG 2005