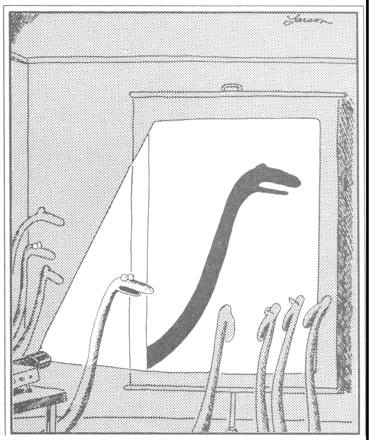
Real-Time Hard Shadows

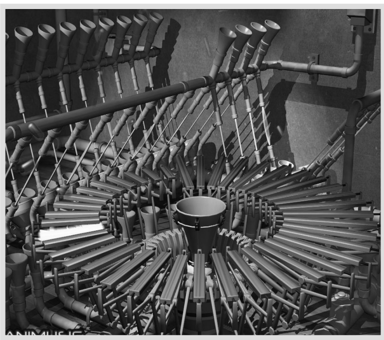


"Now this is…this is…well, I guess it's another snake."

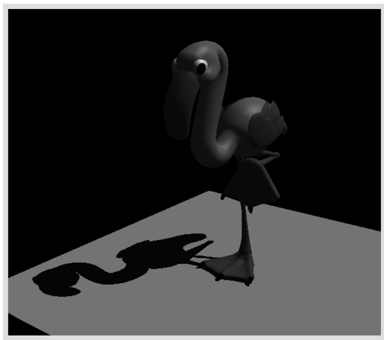
Collected by Ronen Gvili.
Most slides were taken from :
Fredu Durand
Stefan Brabec

Shadows

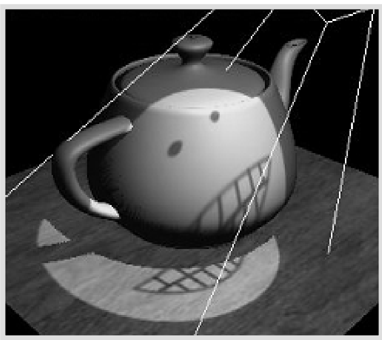
Hard Shadows



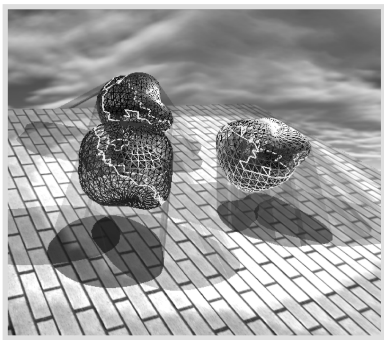
Planar (Projected) Shadows



Shadow Maps



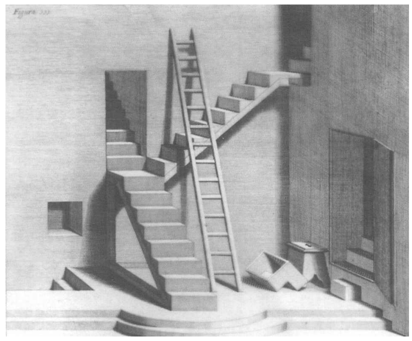
Volume (Stencil) Shadows



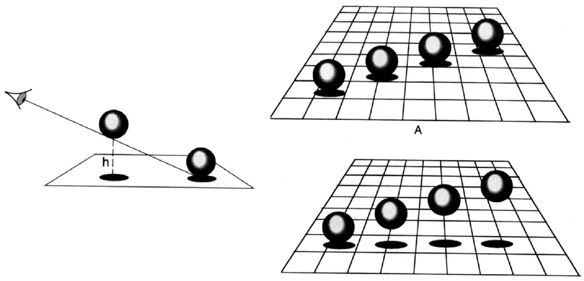
Soft Shadows

Why are Shadows Important?

- Depth cue
- Scene Lighting
- Realism
- Contact points



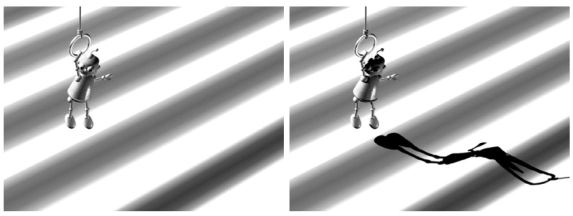
Shadows as a Depth Cue



Shadows as Geometry Cue

(a) Shadows provide information about the relative positions of objects. On the left-hand image, we cannot determine the position of the robot, whereas on the other three images we understand that it is more and more distant from the ground.

Shadows as Geometry Cue

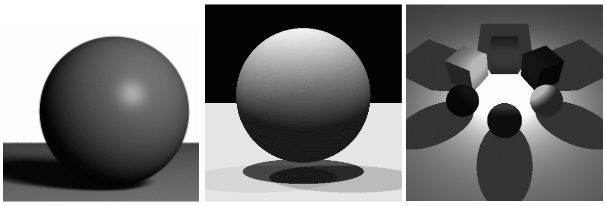


(b) Shadows provide information about the geometry of the receiver. Left: not enough cues about the ground. Right: shadow reveals ground geometry.

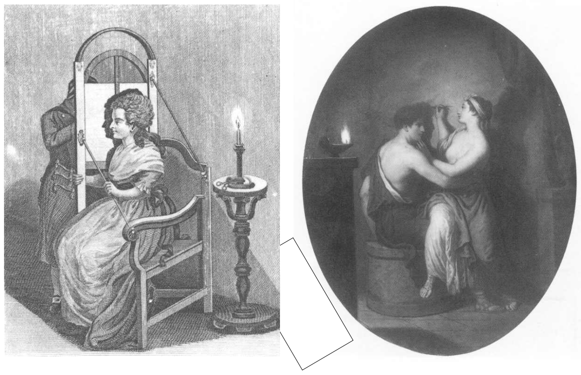
Shadows Provide Extra Information

For Intuition about Scene Lighting

- Position of the light (e.g. sundial)
- Hard shadows vs. soft shadows
- Colored lights
- Directional light vs. point light

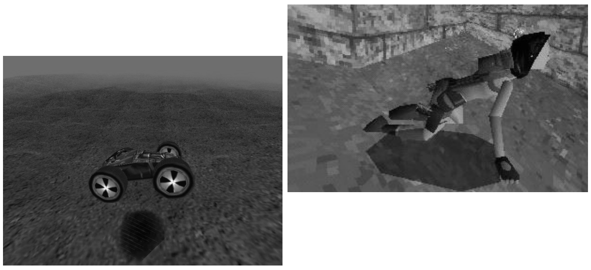


Shadows as the Origin of Painting



Approximated Shadows

- Hand-Drawn Geometry

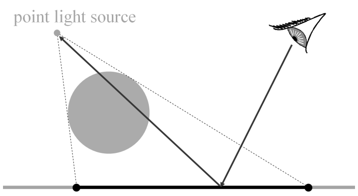


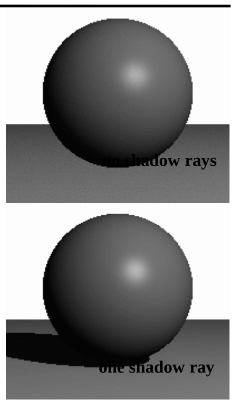
Approximated Shadows

- Polygons / Texture Maps:
  - Precomputed shape that moves with object
  - Rotation / Translation / Scale
  - Blurred (more realistic, soft)
- Pros:
  - Fast & simple: no global computation
- Cons:
  - Quality not very realistic

Shadows – Ray Tracing

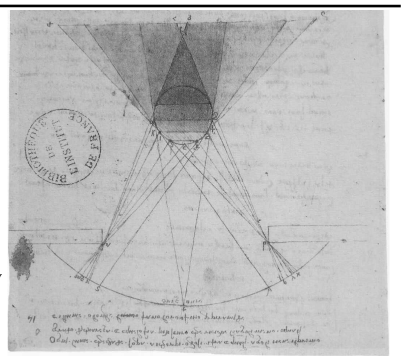
- One shadow ray per intersection per point light source





Soft Shadows

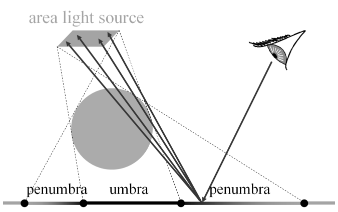
- Caused by extended light sources
- Umbra
  - source completely occluded
- Penumbra
  - Source partially occluded
- Fully lit

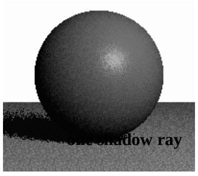


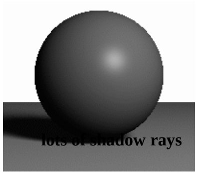
XVI. Léonard de Vinci (1452-1519). Lumière d'une fenêtre sur une sphère ombreuse avec (en partant du haut) ombre intermédiaire, primitive, dérivée et (sur la surface, en bas) portée. Plume et lavis sur pointe de métal sur papier, 24 x 38 cm. Paris, Bibliothèque de l'Institut de France (ms. 2185 ; B.N. 2038, f° 14 r°).

Soft Shadows – Ray Tracing

- Multiple shadow rays to sample area light source

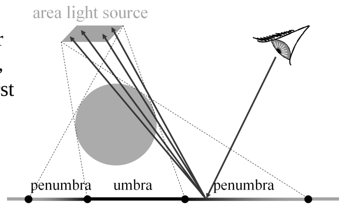




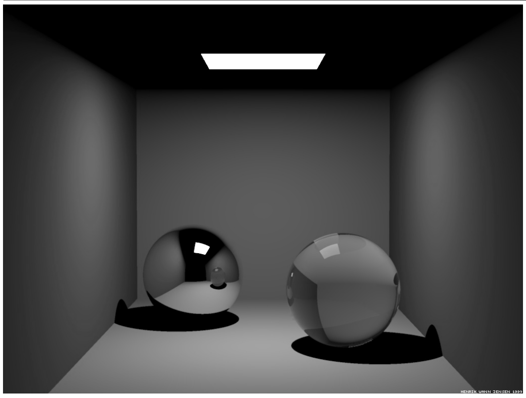


Shadows in Ray Tracing

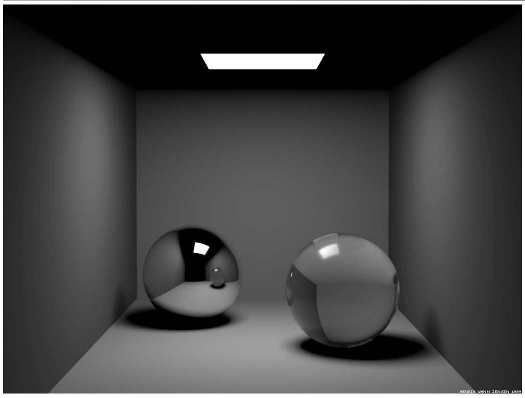
- Shoot ray from visible point to light source
- If blocked, discard light contribution
- Optimization?
  - Stop after first intersection
  - Coherence: remember the previous occluder, and test that object first



Traditional Ray Tracing

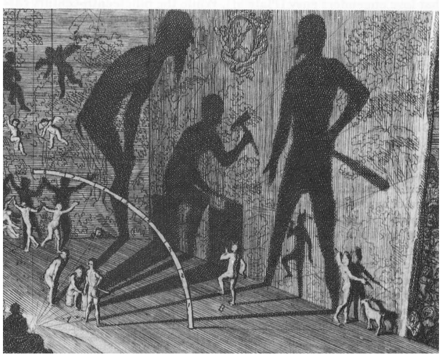


Ray Tracing + Soft Shadows

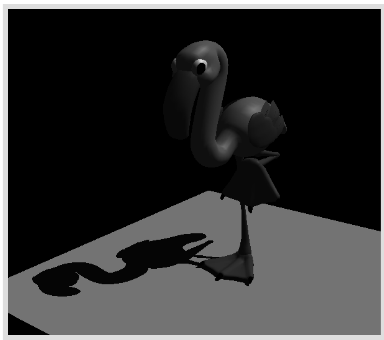


Questions?

Plate 50 Samuel van Hoogstraten, *Shadow Theatre*. From *Inleyding tot de hooghe schoole der schilderkonst* 1678.



Planar (Projected) Shadows



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"
- Used for real-time mirrors (& other reflective surfaces), shadows & more!
- A “scissoring” tool.

Stencil Buffer

- Can specify different rendering operations for each of the following stencil tests:
  - stencil test fails.
  - stencil test passes & depth test fails.
  - stencil test passes & depth test passes.

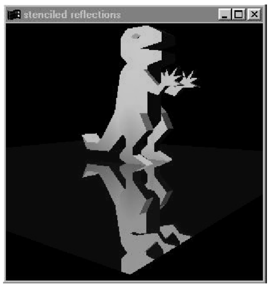
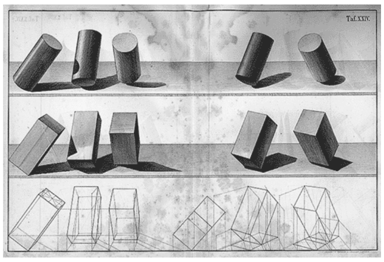


image from NVIDIA’s stencil buffer tutorial (http://developer.nvidia.com)

Planar Shadows

- [Blinn88] *Me and my fake shadow.*
  - Shadows for selected large receiver polygons
    - Ground plane
    - Walls

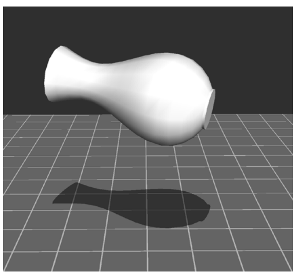


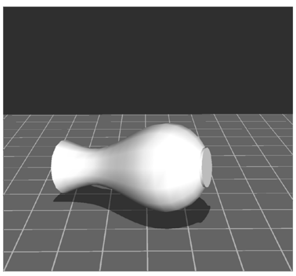
Planar Shadows

- Basic algorithm
  - Render scene (full lighting).
  - For each receiver polygon
    - Compute projection matrix *M*.
    - Mult with actual transformation (modelview).
    - Render selected (occluder) geometry.
      - Darken/Black.

Cast Shadows on Planar Surfaces

- Draw the object primitives a second time, projected to the ground plane.

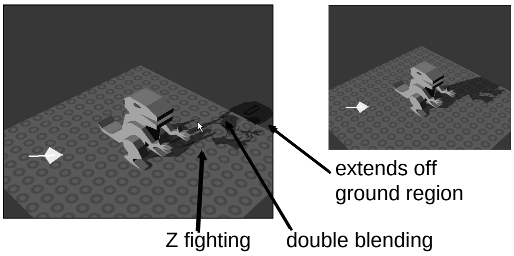




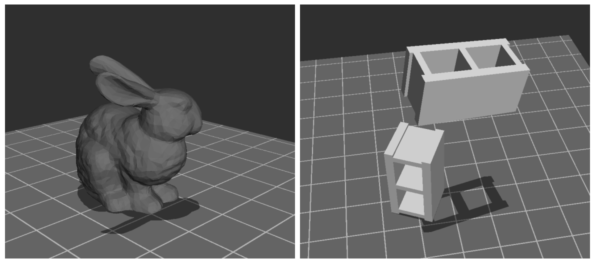
Planar Shadows

- Problems
  - Z-Fighting
    - Use bias when rendering shadow polygons.
    - Use stencil buffer (or disable depth test).
  - Bounded receiver polygon ?
    - Use stencil buffer (restrict drawing to receiver area).
  - Shadow polygon overlap ?
    - Use stencil count (only the first pixel gets through).
  - Does not produce self-shadows, shadows cast on other objects, shadows on curved surfaces, etc.

Planar Shadows

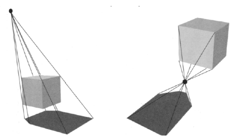


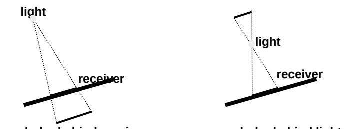
Planar Shadows



Planar Shadows

- Wrong Shadows & Anti-Shadows
  - Objects behind light source.
  - Objects behind receiver.
- Solution
  - Clipping
    - Use 3D-3D transformation e.g. [Heckbert97] for valid z coordinates (setting a clipping plane).





Fake Shadows using Projective Textures

- Separate obstacle and receiver
- Compute b/w image of obstacle from light
- Use image as projective texture for each receiver

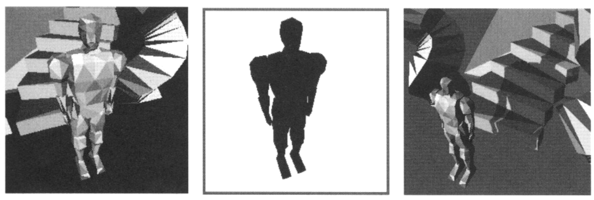


Image from light source BW image of obstacle Final image

Figure from Moller & Haines "Real Time Rendering"

Projected Geometry

- Summary
  - Only practical for very few, large receivers.
  - Easy to implement.
  - Use stencil buffer (z fighting, overlap, receiver).
  - Efficiency can be improved by rendering shadow polygons to texture maps.
    - Occluders and receiver 'static' for some time.

Questions?

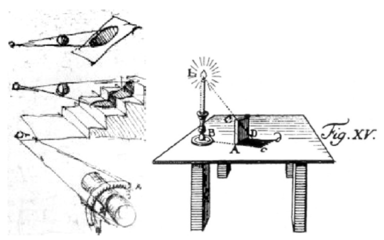
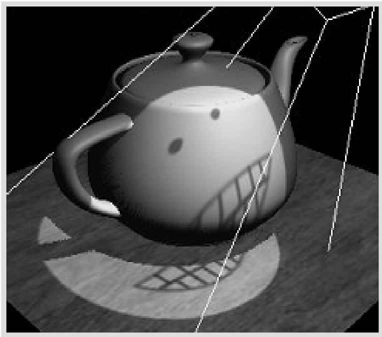


Figure 1: *Left: Study of shadows by Leonardo da Vinci*[48] — *Right: Shadow construction by Lambert*[35].

Shadow Maps

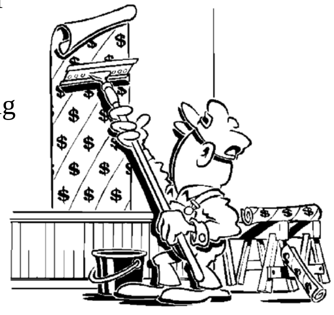


Texture Mapping

- Don't have to represent everything with geometry

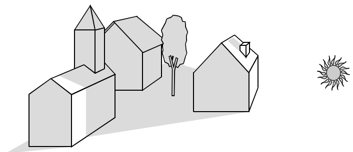
Texture Mapping

- Like wallpapering or gift-wrapping with stretchy paper
- Curved surfaces require extra stretching or cutting
- More on this in a couple weeks...

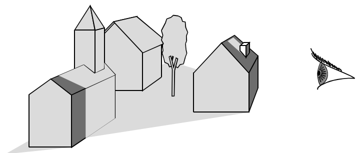


Shadow/View Duality

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

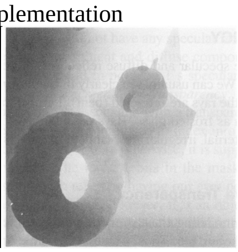


- Shadow computation similar to view computation



Shadow maps

- [Williams78] *Casting curved shadows on curved surfaces.*
  - Image-space algorithm
  - Well suited for hardware implementation



Shadow Mapping

- Texture mapping with depth information
- ≥ 2 passes through the pipeline
  - Compute shadow map (depth from light source)
  - Render final image (check shadow map to see if points are in shadow)

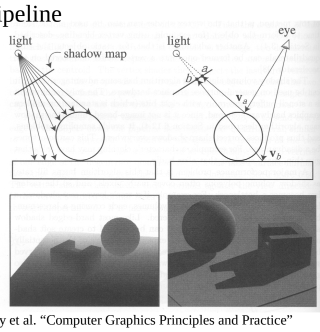
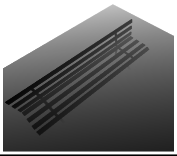
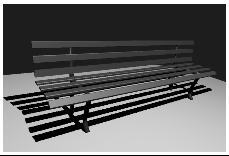


Figure from Foley et al. “Computer Graphics Principles and Practice”

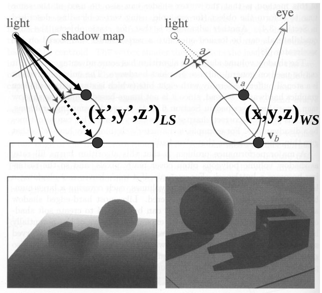
Shadow Maps

- Algorithm:
  - – Render scene as seen from light source.
  - – Save back depth buffer (2D shadow map).
  - – Render scene from viewer's position:
    - Transform pixel coordinates to light source space.
    - Compare z with z value stored in shadow map:
      - – Pixel is in shadow if z(light) < z(viewer).

Shadow Map Look Up

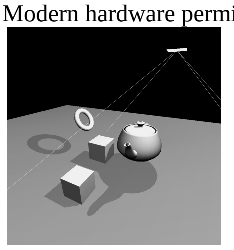
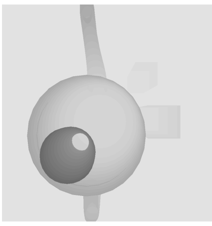
- We have a 3D point $(x,y,z)_{WS}$
- How do we look up the depth from the shadow map?
- Use the 4x4 perspective projection matrix from the light source to get $(x',y',z')_{LS}$
- ShadowMap(x',y') < z'?



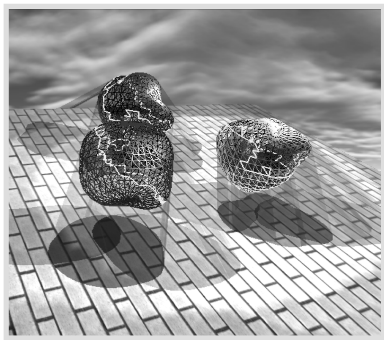
Foley et al. "Computer Graphics Principles and Practice"

Shadow Maps

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

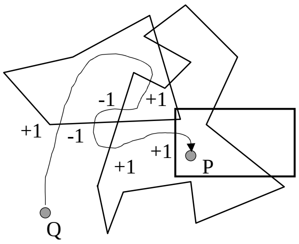



Volume (Stencil) Shadows



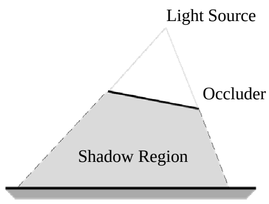
Shadow Volumes

- Six intersections: +1, +1, -1, -1, +1, +1
- Sum = 2: P is inside 2 polyhedra



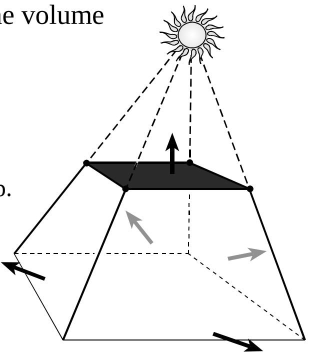
Shadow Volumes

- [Crow77] *Shadow algorithms for computer graphics.*
  - – Compute regions of shadow in 3D
    - Object-space algorithm.
    - Cast shadows onto arbitrary receiver geometry (polygons).



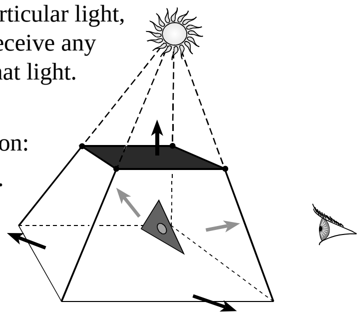
Shadow Volumes

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



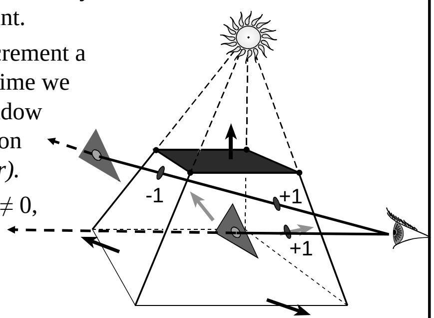
Shadow Volumes

- If a point is inside a shadow volume cast by a particular light, the point does not receive any illumination from that light.
- Naive implementation: #polygons * #lights.

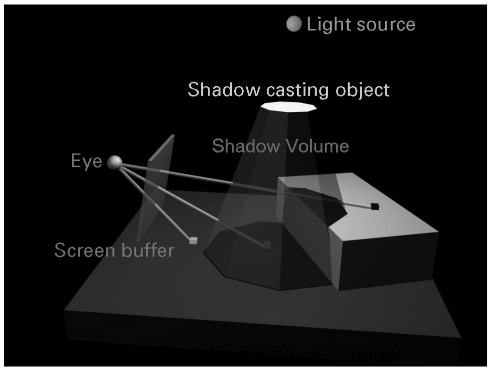


Shadow Volumes

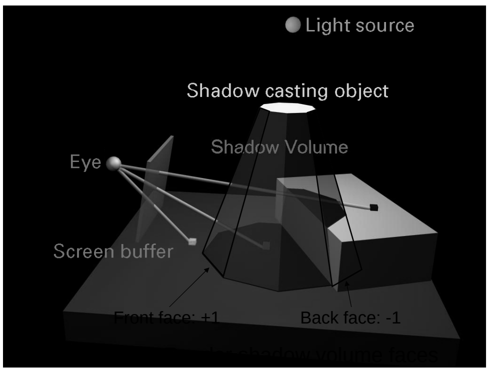
- Shoot a ray from the eye to the visible point.
- Increment/decrement a counter each time we intersect a shadow volume polygon *(check z buffer).*
- If the counter ≠ 0, the point is in shadow.



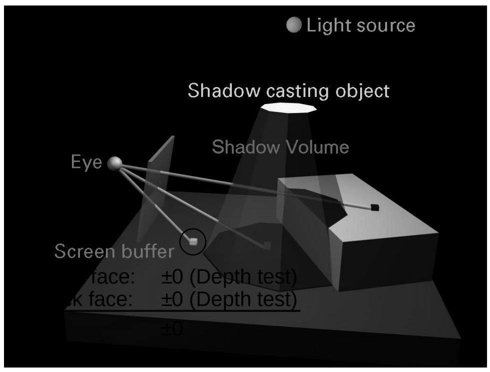
Shadow Volumes



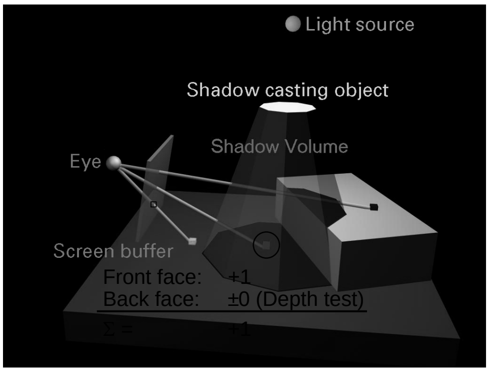
Shadow Volumes



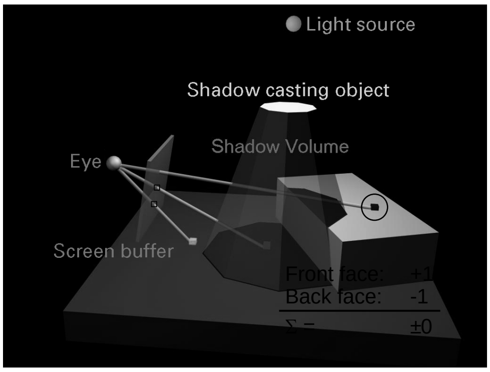
Shadow Volumes



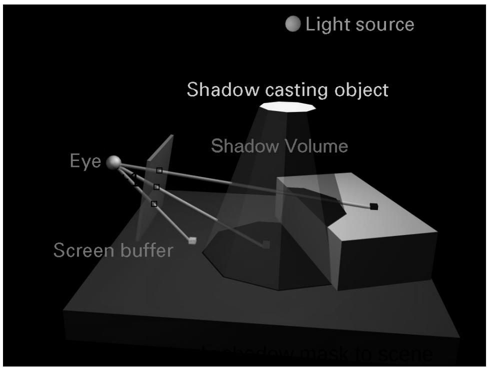
Shadow Volumes



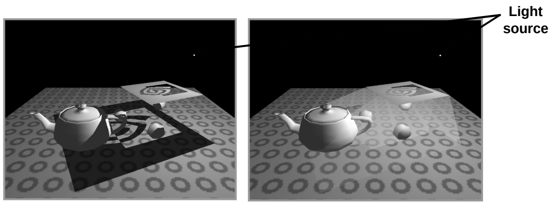
Shadow Volumes



Shadow Volumes



Shadow Volumes



Shadow Volumes

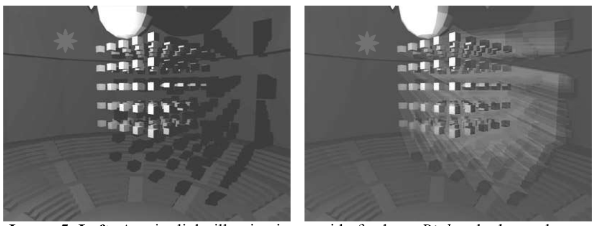
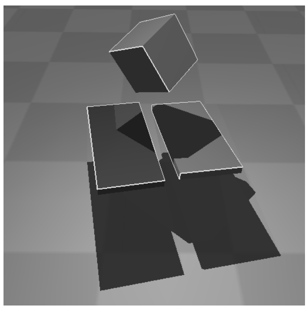


Image 5. Left: A point light illuminating a grid of cubes. *Right:* shadow volumes

Shadow Volumes w/ the Stencil Buffer

- [Heidmann 91] *Real shadows real time.*



Shadow Volumes w/ the Stencil Buffer

Initialize stencil buffer to 0

Draw scene with ambient light only

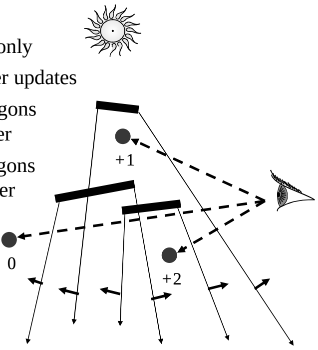
Turn off frame buffer & z-buffer updates

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

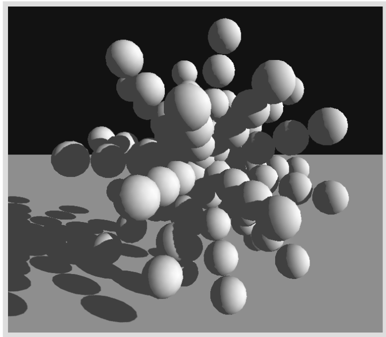
Draw back-facing shadow polygons
    If z-pass → decrement counter

Turn on frame buffer updates

Turn on lighting and
    redraw pixels with
    counter = 0

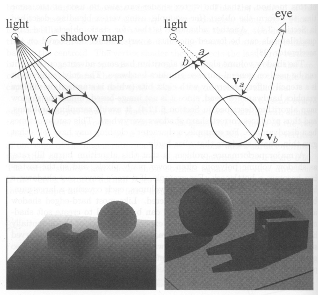


Shadow Maps



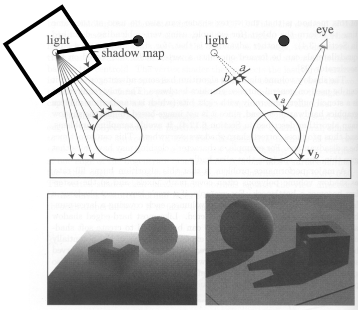
Limitations of Shadow Maps

1. Field of View
2. Resolution in Z coordinates.
3. Aliasing – Resolution in XY coordinates.



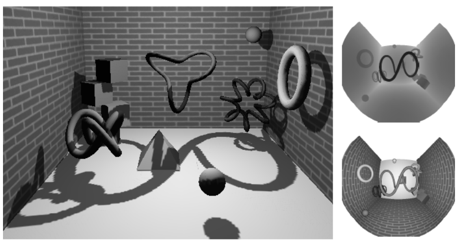
Field of View Problem

- What if point to shadow is outside field of view of shadow map?
  - Use cubical shadow map
  - Use only spot lights!

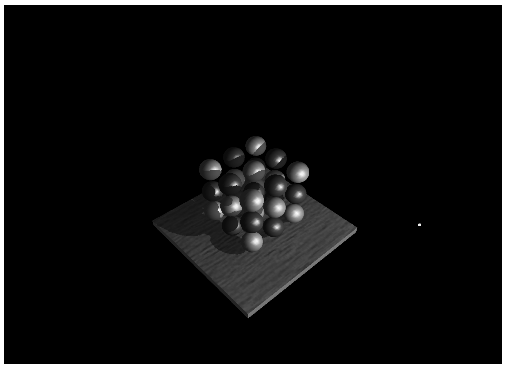


Field of View Problem

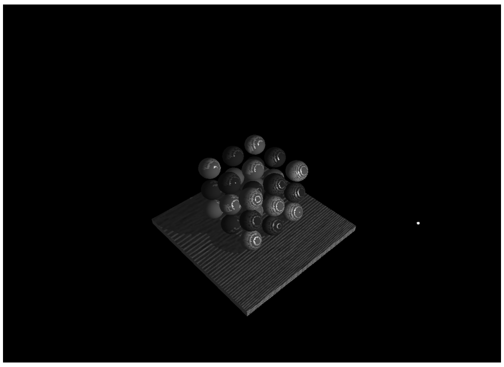
- Cubical Shadow Maps



Aliasing In Z Coordinates.

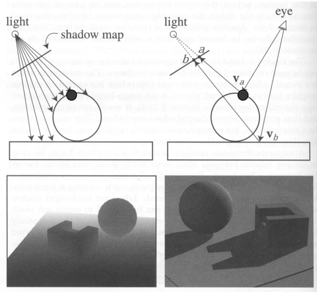


Aliasing In Z Coordinates.



Aliasing In Z Coordinates.

- For a point visible from the light source
  ShadowMap(x',y') ≈ z'
- This happens due to finite resolution in the Z-Buffer (8-bit) and the sampling (number of pixels of the Z-Buffer).



Aliasing In Z Coordinates.

• The difference in the depths of the samples is based on the slope of the polygon in light space.

Recall : Pixel is in shadow if z(light) < z(viewer).

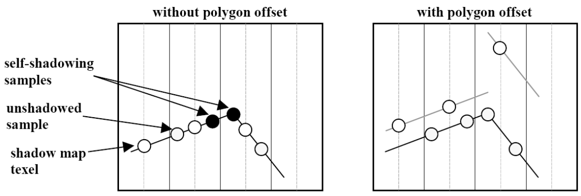
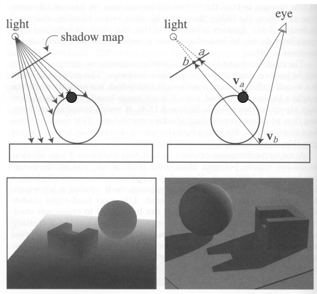


Figure 3. These figures illustrate the need for polygon offsetting to eliminate self-shadowing artifacts. The variable sampling location necessitates the use of z slope-based offset.

Aliasing In Z Coordinates.

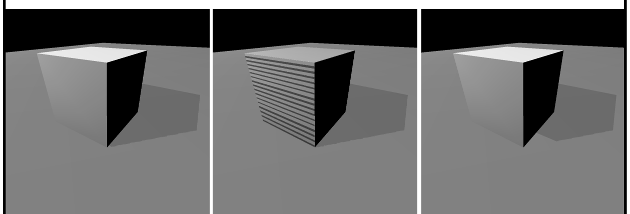
- For a point visible from the light source
  ShadowMap(x',y') ≈ z'
- How can we avoid erroneous self-shadowing?
  - Narrow the light frustum.
  - Add bias (epsilon).



Bias (Epsilon) for Shadow Maps

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

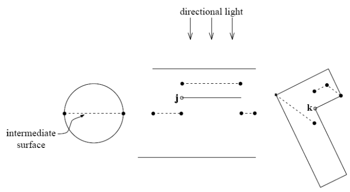
Choosing a good bias value can be very tricky.



Correct image | Not enough bias | Way too much bias

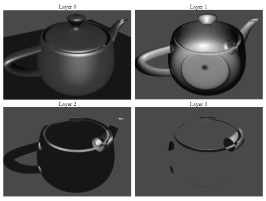
Bypassing the Bias - Midpoint

- Using an intermediate surface [Woo 92] : instead of keeping the closest depth value , the 2 closest values are kept in 2 buffers.
- The 2 buffers are averaged into one – which is used as the *Shadow Map*.



Bypassing the Bias - Midpoint

- A method to generate those buffers is *Depth Peeling* [Everitt 01].
  - Requires additional pass and extra memory..
  - Requires closed surfaces..



Bypassing the Bias - Midpoint

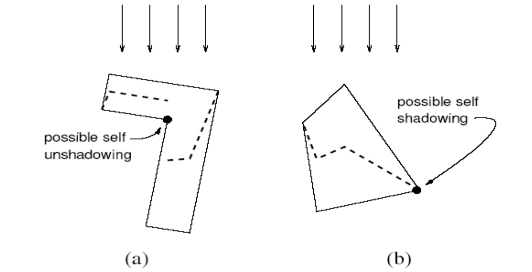
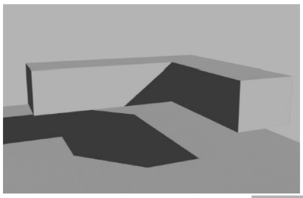
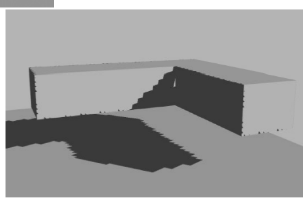


Figure 2: *Self-unshadowing (a) and self-shadowing (b) problems for midpoint shadow maps.*

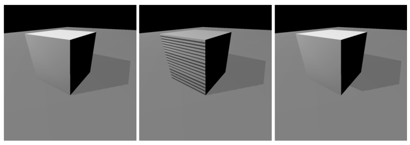
Bypassing the Bias - Midpoint





Bypassing the Bias – DD Layers

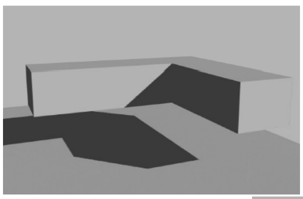
- [Weiskopf 03] *Shadow Mapping Based On Dual Depth Layers.*
  - Define the bias as: $Z_{bias} = \min(\frac{z_2 - z_1}{2}, Z_{offset})$

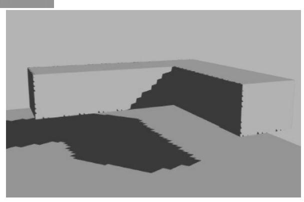


Correct image | Not enough bias | Way too much bias

- Z_{offset} prevents self-unshadowing in case $Z_2 << Z_1$.
- In case $Z_2 \sim Z_1$ the bias is determined by the midpoint preventing self-shadowing.

Bypassing the Bias – DD Layers



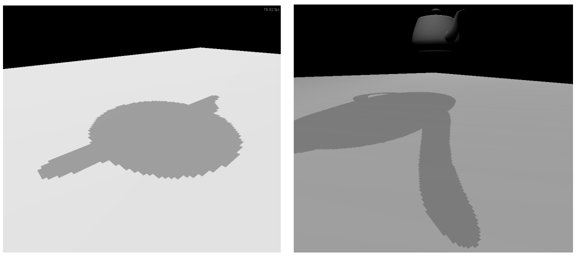


Bypassing the Bias

- Using *Priority Buffers* [Hourcade 85] : storing IDs instead of storing depth. Each polygon is given a different ID , rendered into the color buffer (from the light's pov). The Z-buffer resolves the ordering differences.
  - No hardware support..
- Using ID per object and not polygon.
  - No self shadows.
- Split the objects into low roughly convex pieces [valchos 01].

Aliasing In XY Coordinates.

- Under-sampling of the shadow map
- Reprojection aliasing – especially bad when the camera & light are pointing towards each other



Shadow Map Filtering

- [Reeves 87] *Percentage closer filtering*.
  - – Filtering depth values makes no sense
  - – Perform shadow test before filtering

113	112	112
113	23	23
113	24	24

compare
< 54

0	0	0
0	1	1
0	1	1

filter
4x one, 5x zero

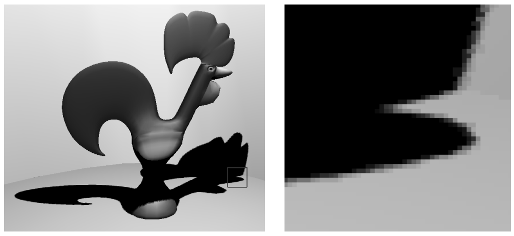
44 %
shadowed

Percentage Closer Filtering

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

Percentage Closer Filtering - Hardware

- [Brabec 01] *Hardware-accelerated Rendering of Antialiased Shadows With Shadow Maps*.

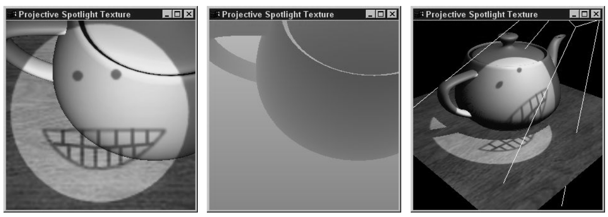


fast PCF (filter size 2x2)

Hardware-based PCF

- Multi-channel shadow map
  - – Use RGBA instead of alpha channel only
    - 4 values to sample a 2x2 region
    - Increases effective shadow map resolution by a factor of 2 in each dimension
  - – Shadow map generation:
    - Render scene four times where in each pass
      - – One channel (R,G,B or A) is selected
      - – Image-plane is jittered (stratified sampling)
    - Copy RGBA image to texture

Projective Texturing + Shadow Map



Light's View | Depth/Shadow Map | Eye's View

Images from Cass Everitt et al., "Hardware Shadow Mapping" NVIDIA SDK White Paper

Shadows in Production

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

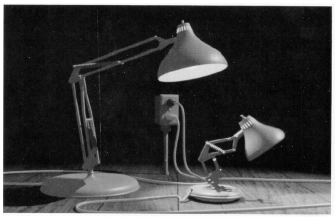


Figure 12. Frame from *Luxo Jr.*

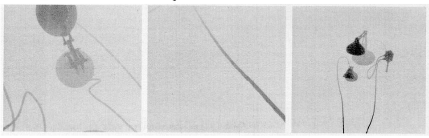


Figure 13. Shadow maps from *Luxo Jr.*

Shadow Maps - Pros

- Simplicity - simple to implement.
- Performance - can achieve (almost) real-time performance without gpu.
- Flexibility - data representation independent.
- Can be simply implemented in the GPU as a hardware texture.
- High quality variation made it usable in films .
- Extendable to produce soft shadow.
- Extended to handle non-opaque object shadowing.

Shadow Maps - Cons

- Quality – aliasing and self shadowing.
- No association information between occluder and receiver.
- More than a single shadow map is required per single point light (as so true for spotlights with large angle of view).
- Low rendering in cases the view region and the shadow map are poorly overlap.
- Changes in the shadow coverage can result in changes in the rendering quality (animation).

Shadows Maps In games

Blade Of Darkness

Shadows Maps In games

Half Life 2

Shadows Maps In games

Half Life 2

Shadows Maps In games

Splinter Cell

Shadows Maps In games

Silent Hill 3

Shadows Maps In games

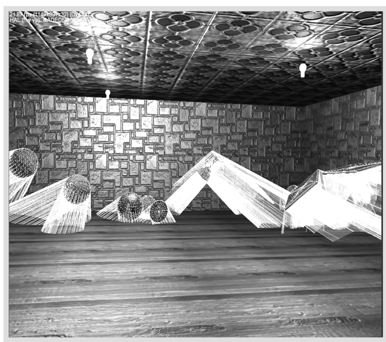
Deus Ex 2

Shadows Maps In games

Deus Ex 2

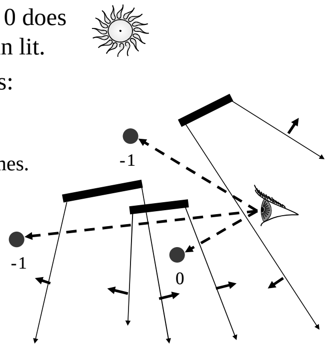
Questions?

Volume (Stencil) Shadows



If the Eye is in Shadow...

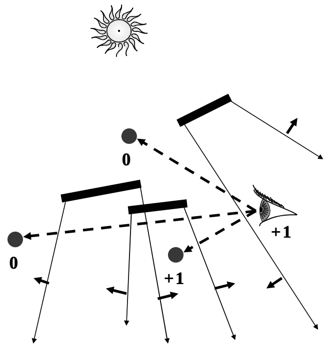
- ... then a counter of 0 does not necessarily mean lit.
- 3 Possible Solutions:
  1. Explicitly test eye point with respect to all shadow volumes.
  2. Clip the shadow volumes to the view frustum.
  3. "Z-Fail" shadow volumes.



1. Test Eye with Respect to Volumes

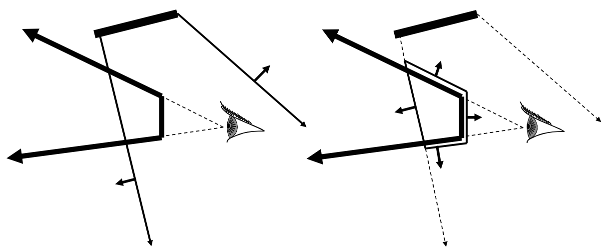
- Adjust initial counter value

Expensive..



2. Clip the Shadow Volumes

- Clip the shadow volumes to the view frustum and include these new polygons
- Messy CSG (Constructive Solid Geometry).



3. "Z-Fail" Shadow Volumes

- [Carmack 01] *"Carmack's Reverse"*

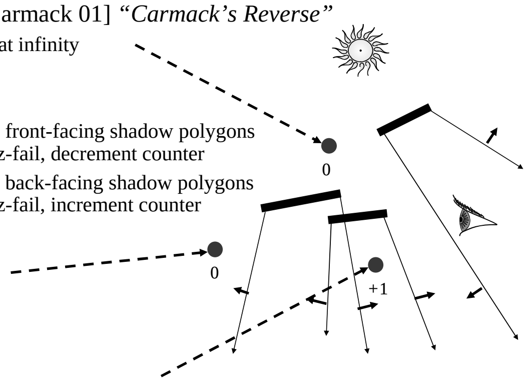
Start at infinity

...

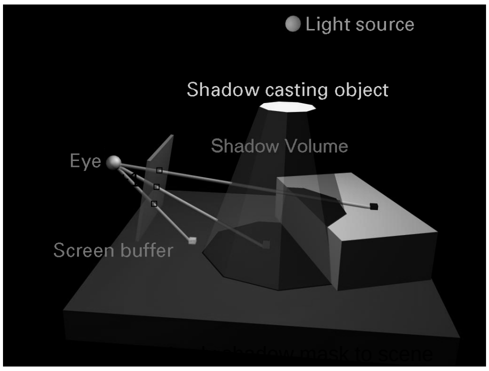
Draw front-facing shadow polygons
  If z-fail, decrement counter

Draw back-facing shadow polygons
  If z-fail, increment counter

...

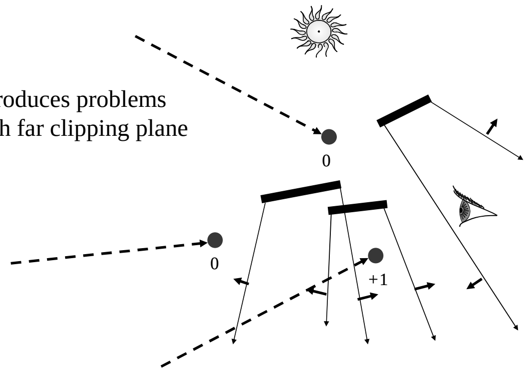


3. "Z-Fail" Shadow Volumes



3. "Z-Fail" Shadow Volumes

- Introduces problems with far clipping plane



Z-Fail versus Z-Pass

- When stencil increment/decrements occur:
  - Z-Pass: on depth test pass.
  - Z-Fail: on depth test fail.
- Increment on:
  - Z-Pass: front faces.
  - Z-Fail: back faces.
- Decrement on:
  - Z-Pass: front faces.
  - Z-Fail: back faces.

Z-Fail versus Z-Pass

- Both cases order passes based stencil operation:
  - First, render increment pass.
  - Second, render decrement pass.
- Which clip plane creates a problem:
  - Z-Pass: near clip plane.
  - Z-Fail: far clip plane.

Z-Fail versus Z-Pass

- If we could avoid either near plane or far plane view frustum clipping, shadow volume rendering could be robust.
- Avoiding near plane clipping:
  - Not really possible.
  - Objects can always be behind you.
  - Moreover, depth precision in a perspective view goes to hell when the near plane is too near the eye.
- Avoiding far plane clipping:
  - Perspective make it possible to render at infinity.
  - Depth precision is terrible at infinity, but we just care about avoiding clipping.

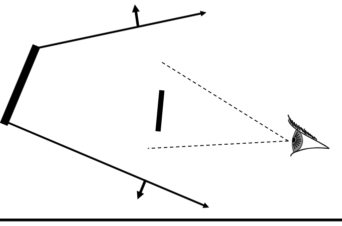
Capping The Volumes ..

- The light point is facing the viewer, yet is partially occluded.

Capping the volumes ..

- Incorrect shadows.

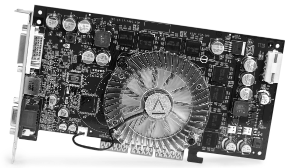
- A shadow volumes must be bounded not only by its sides but in its top and bottom ..
  - The occluder polygon can be used as the top .
  - A polygon connecting the volume edges in the infinity can be used as the bottom .



Avoiding far plane clipping - Hardware

Using NV_depth_clamp :

- All objects that normally clipped by the far plane are instead drawn on the far plane with maximum z-depth.
  - Hardware dependent (not supported in ATI cards).
  - Filling more pixels.. (might be slower than z-pass).



Avoiding far plane clipping - Software

- [Everitt 2002] *Robust Stenciled Shadow Volumes.*

Replace the far plane with Infinity.

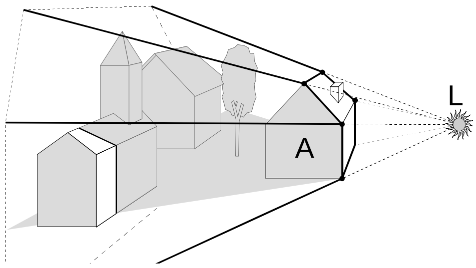
$$\mathbf{P} = \begin{bmatrix} \frac{2\times Near}{Right - Left} & 0 & \frac{Right + Left}{Right - Left} & 0 \\ 0 & \frac{2\times Near}{Top - Bottom} & \frac{Top + Bottom}{Top - Bottom} & 0 \\ 0 & 0 & -\frac{Far + Near}{Far - Near} & -\frac{2\times Far \times Near}{Far - Near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Shadow Volumes

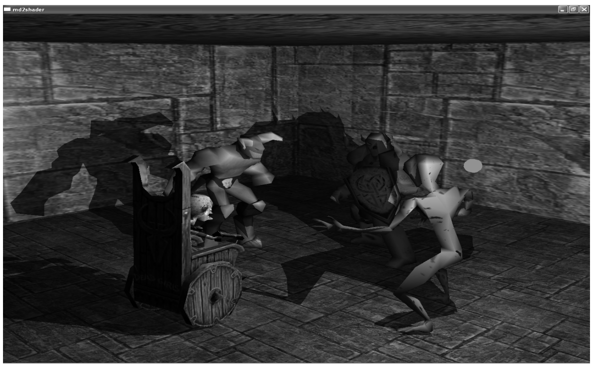
- Counting problems
  - Stencil depth
    - 8 bits for intersecting volumes
      - Stencil wrap mode
        - » Missing shadows for ‘counter mod 2^n == 0’
      - Stencil clamp
        - » Missing shadows (missing some ‘enter’ events).
    - 1 bit enough for non-intersecting volumes
      - Generate volumes from silhouette.
      - Toggle stencil bit.

Optimizing Shadow Volumes

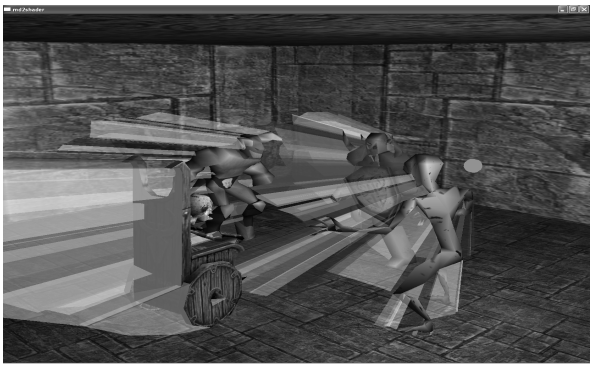
- Use silhouette edges only (edge where a back-facing & front-facing polygon meet)



Optimizing Shadow Volumes



Optimizing Shadow Volumes



Optimizing Shadow Volumes

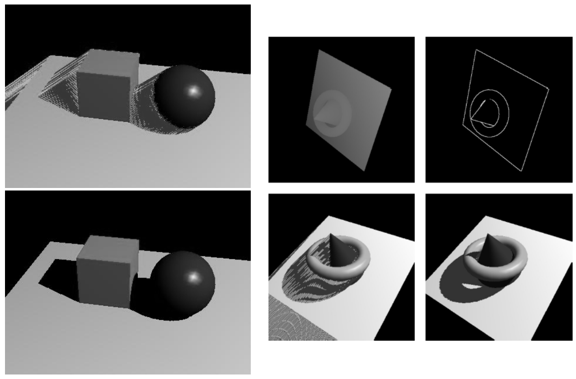
Optimizing Shadow Volumes - 2

- [McCool00] *Shadow volume reconstruction from depth maps* .
  - Combine the pros from shadow maps and shadow volumes:
    - Shadow volumes generated from depth maps.
    - Reduced number of shadow volumes for very complex scenes.
    - Does not need special hardware features (standard shadow texture using stencil buffer)

Optimizing Shadow Volumes - 2

- Render the scene from light source.
- Read back Z-Buffer.
- Reconstruct shadow volumes:
  - Canny edge detection.
  - Surface reconstruction.
- Render shadow volumes with stencil operation.
- Render final scene.

Optimizing Shadow Volumes - 2



Optimizing Shadow Volumes - 2

- Summary
  - Better than normal shadow volumes for very complex scenes:
    - Volume for silhouette.
    - Only one stencil bit (in-out toggle).
  - Needs CPU and memory transfer
    - Use CPU's special instruction set
    - OpenGL imaging extensions (convolution)

Optimizing Shadow Volumes - 3

Ultra Shadows:

- Using EXT_depth_bounds_test the programmer can cull the shadow pixels by setting bounds for the light/shadow region.

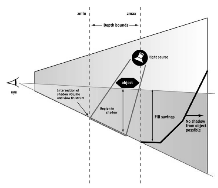


Figure 2. Programmers can define a subset of the scene (within z-min and z-max) to limit lighting/shadow calculations to the appropriate area for each light source.

  - Hardware dependent (not supported in ATI cards).
  - Requires scene preprocess .

Shadow Volumes - Cons

- Representation dependent (polygonal).
- Introduces a lot of new geometry.
- Limited precision of stencil buffer (counters).
- For optimization purposes (silhouette detection) requires adjacency information.
- Objects must be watertight to use silhouette trick.
- High fill rate – many long shadow polygons need to be scan converted.
- The soft shadow extension is non-trivial.
- Aliasing errors in the shadow counts due to scan conversion of very narrow shadow polygons.
- Handling transparent object can not be easily implanted in the GPU.

Shadow Volumes - Pros

- Precision – computed in object space, omni-directional.
- GPU support - stencil buffer (alpha buffer).
- Real time variations required no GPU development.
- Extendable to produce soft shadows.
- Advanced variations can deal with non-polygonal objects.

Shadows Volumes In games

Doom 3

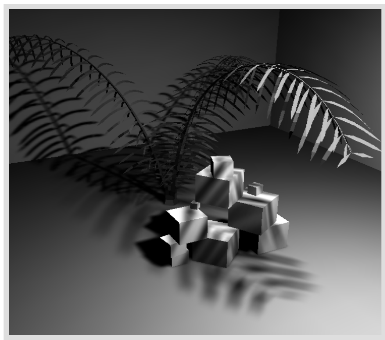
Shadows Volumes In games

Neverwinter Nights

Questions?

Plate 52 Grandville, *The Shadows (The French Cabinet)* from *La Caricature*, 1830.

Next Time : Soft Shadows



The End…