



## Texture Mapping II

- Light maps
- Environment Maps
- Projective Textures
- Bump Maps
- Displacement Maps
- Solid Textures
- Mipmaps
- Shadows

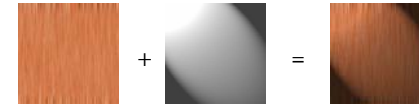
1

7. Texture Mapping



## Light Maps

- Simulates the effect of a local light source



- Can be pre-computed and dynamically adapted

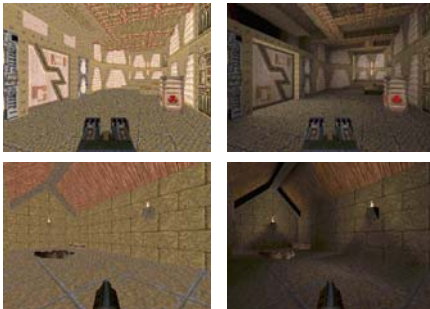
2

7. Texture Mapping



## Light Maps

- Texture mapping in Quake



textures only

textures and light maps

3

7. Texture Mapping



## Environment Map



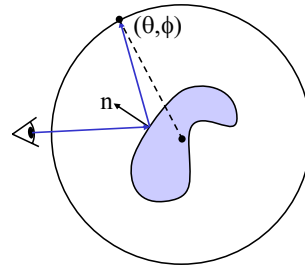
4

7. Texture Mapping



## Environment Map

- Method to render **reflective objects**
- Compute intersection of reflected ray with surrounding sphere
- Take parameter values of intersection as texture coordinates



5

7. Texture Mapping



## Examples – Environment Map



6

7. Texture Mapping



## Environment Map

- How to get an environment map of a real environment?



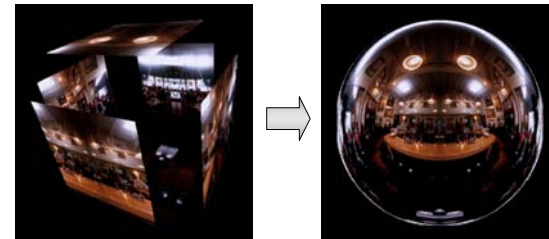
7

7. Texture Mapping



## Cube Mapping

- Sphere can be replaced by cube
- Simplify computations

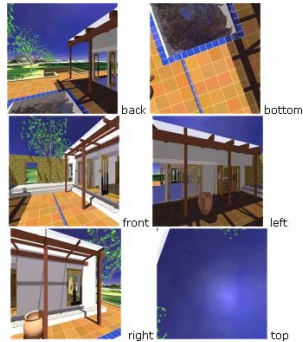


8

7. Texture Mapping



## Cube Map Demo



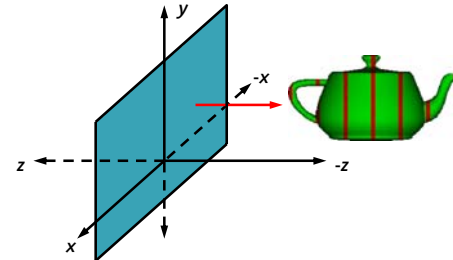
[http://developer.nvidia.com/object/cube\\_map\\_ogl\\_tutorial.html](http://developer.nvidia.com/object/cube_map_ogl_tutorial.html)

7. Texture Mapping 9



## Linear Mapping

- Uses object or eye coordinates
- (In)dependent of transforms
- Can be used to visualize distance from objects

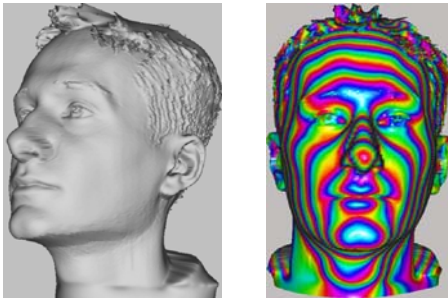


7. Texture Mapping 10



## An Example

- Mapping of distances from laser range data



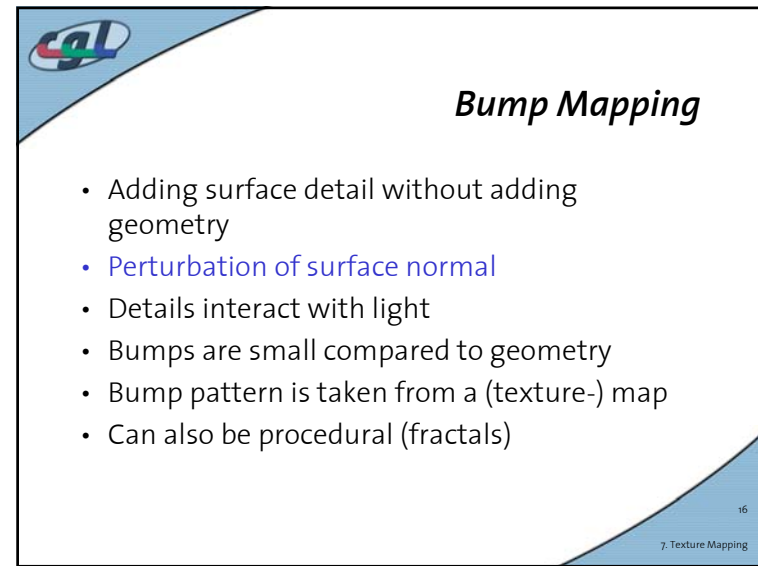
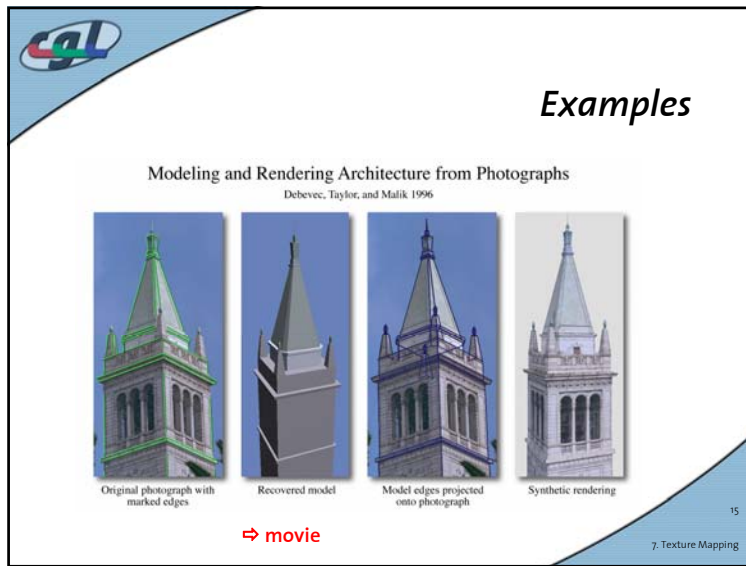
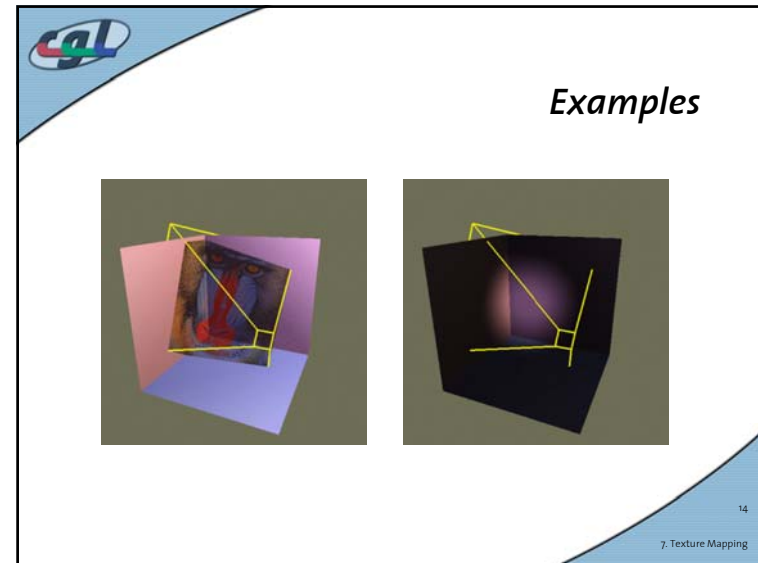
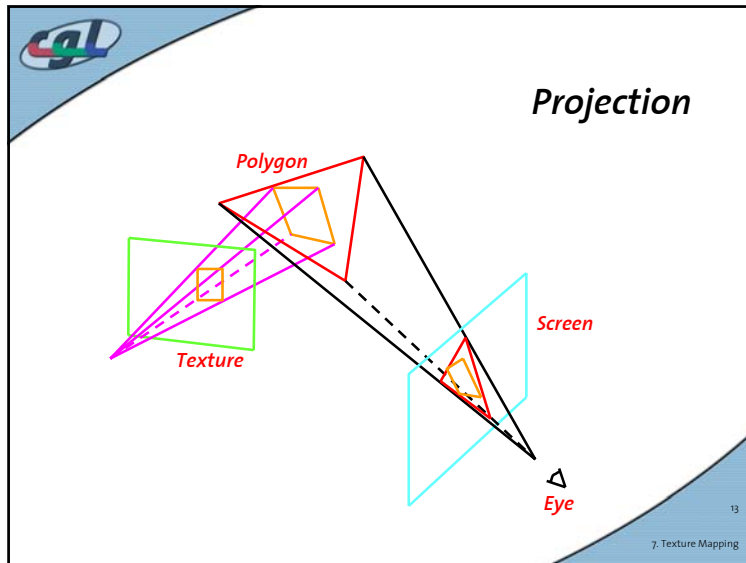
7. Texture Mapping 11



## Projective Textures

- Generalize texture coordinates to a 4D homogeneous vector  $(u, v, r, q)$
- Texture matrix computes full  $4 \times 4$  transform to  $(u^P, v^P)$  used for texture lookup
- Texture image can be projected independently of viewing projection
- Applications:
  - Slide projector
  - Spotlight simulation

7. Texture Mapping 12





## Bump Mapping

- Given a surface  $\mathbf{p}(u,v)$  and a perturbation value  $b$  (Jim Blinn)

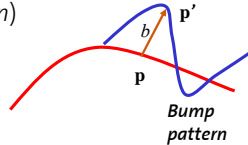
$$\mathbf{n} = \frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial v} = \mathbf{p}_u \times \mathbf{p}_v$$

- Point  $\mathbf{p}'$  on the bumpy surface

$$\mathbf{p}' = \mathbf{p} + \frac{b \mathbf{n}}{|\mathbf{n}|}$$

- Compute normal at Point  $\mathbf{p}'$

$$\mathbf{n}' = \frac{\partial \mathbf{p}'}{\partial u} \times \frac{\partial \mathbf{p}'}{\partial v}$$



17

7. Texture Mapping



## Bump Mapping

- Partial derivatives at point  $\mathbf{p}'$

$$\frac{\partial \mathbf{p}'}{\partial u} = \frac{\partial \mathbf{p}}{\partial u} + \frac{\partial}{\partial u} \left( \frac{b \mathbf{n}}{|\mathbf{n}|} \right)$$

- Perturbed normal approximated by (see Blinn)

$$\mathbf{n}' = \mathbf{n} + b_u (\mathbf{n} \times \mathbf{p}_u) + b_v (\mathbf{n} \times \mathbf{p}_v)$$

18

7. Texture Mapping

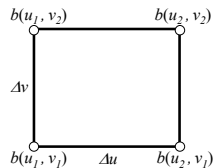


## Bump Mapping

- Discretization using Finite Differences

$$b_u = \frac{b(u_2, v_1) - b(u_1, v_1) + b(u_2, v_2) - b(u_1, v_2)}{2 \Delta u}$$

$$b_v = \frac{b(u_1, v_2) - b(u_1, v_1) + b(u_2, v_2) - b(u_2, v_1)}{2 \Delta v}$$

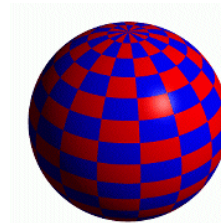


19

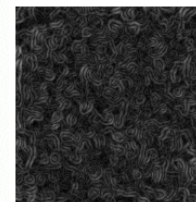
7. Texture Mapping



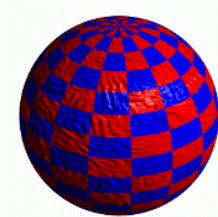
## Examples



Sphere w/Diffuse Texture




Swirly Bump Map




Sphere w/Diffuse Texture &amp; Bump Map

20

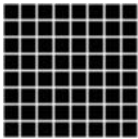
7. Texture Mapping




## Examples



Cylinder w/Diffuse Texture Map




Bump Map



Cylinder w/Texture Map & Bump Map

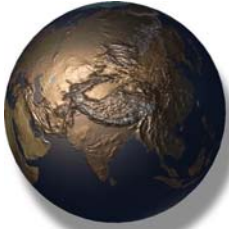
⇒ movie

21  
7. Texture Mapping




## Bump Mapping

- What's missing?
  - Bumps on silhouette
  - Self-occlusion
  - Self-shadowing

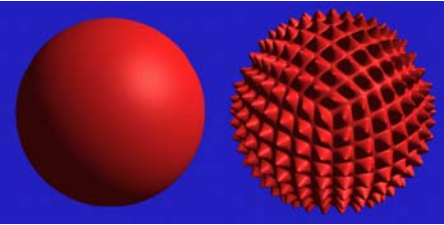


22  
7. Texture Mapping




## Displacement Mapping

- Use the texture map to displace the geometry



23  
7. Texture Mapping



## Displacement Mapping




Image from:

*Geometry Caching for Ray-Tracing Displacement Maps*  
by Matt Pharr and Pat Hanrahan.

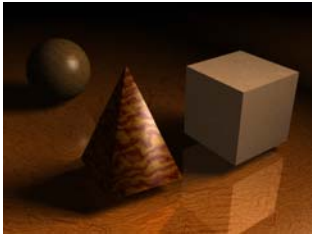
*note the detailed shadows cast by the stones*

24  
7. Texture Mapping



## Solid Textures

- 3D bitmaps
- Procedural textures

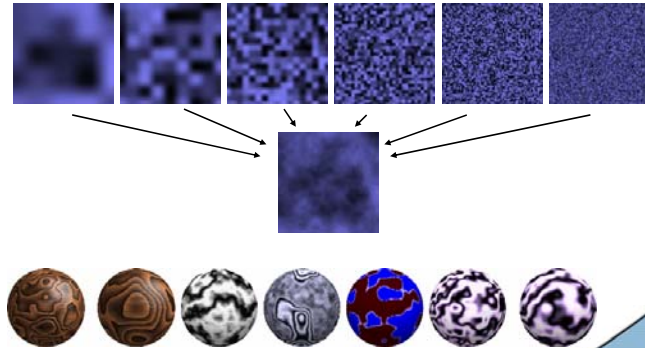


25

7. Texture Mapping



## Perlin Noise



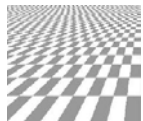
26

7. Texture Mapping

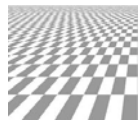


## Mip-Mapping

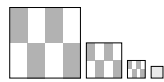
- **Minimized textures** produce aliasing effects
- Store texture at multiple levels-of-detail
- Use **smaller versions** when **far from camera**
- **MIP** comes from the Latin *multum in parvo*, meaning a multitude in a small space.



without mipmap



with mipmap



mipmap

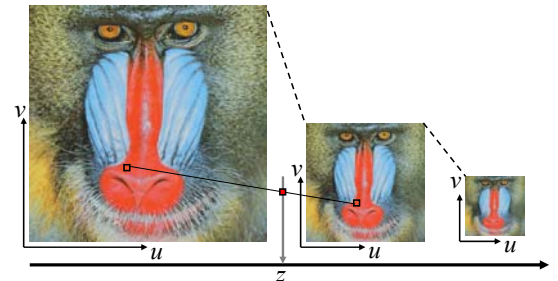
27

7. Texture Mapping



## Texture Interpolation

- Compute texture value  $(R,G,B)$  as function of  $(u,v,z)$
- Tri-linear interpolation

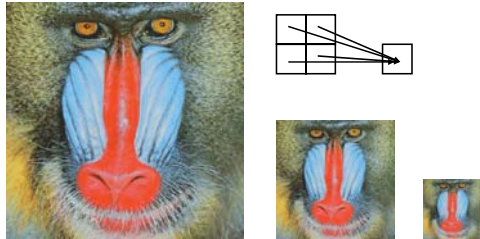


28

7. Texture Mapping



## Computation of the Mip Map



- Color = weighted average of nearby pixels (filter)
- See `gluBuild2DMipMaps()`

⇒ demo

29

7. Texture Mapping



## Shadows

- Why are shadows important?
  - Depth cue
  - Scene lighting
  - Realism
  - Contact points

Plate 10 Samuel van Hoogstraten, *Shadows Thrown From Following Air de Angles schouw der architectuur* 1678.



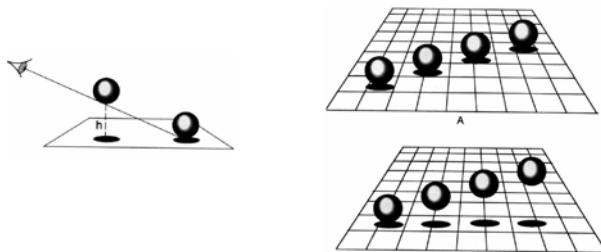
30

from Fredo Durand's graphics class...

7. Texture Mapping



## Shadows as a Depth Cue



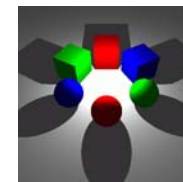
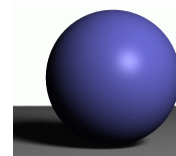
31

7. Texture Mapping



## For Intuition about Scene Lighting

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



32

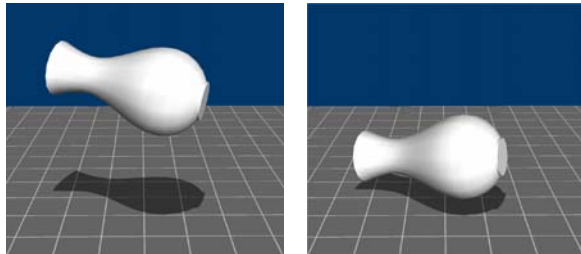
7. Texture Mapping





## Cast Shadows on Planar Surfaces

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



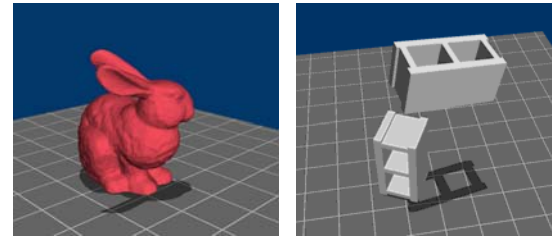
33

7. Texture Mapping



## Limitations of Planar Shadows

- Does not produce self-shadows, shadows cast on other objects, shadows on curved surfaces, etc.



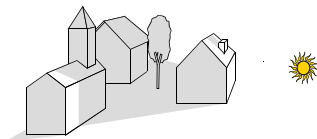
34

7. Texture Mapping

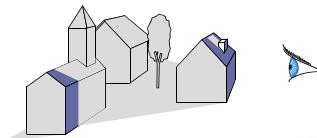


## Shadow/View Duality

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



- Shadow computation similar to view computation



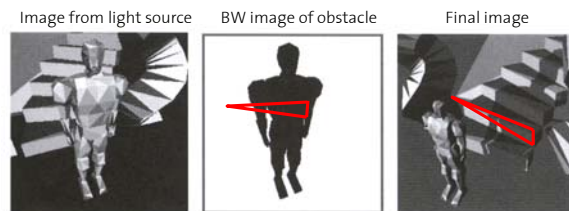
35

7. Texture Mapping



## 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



36

7. Texture Mapping

Figure from Moller & Haines "Real Time Rendering"



## Projective Texture Shadow Limitations

- Must specify occluder & receiver
- No self-shadows
- Resolution

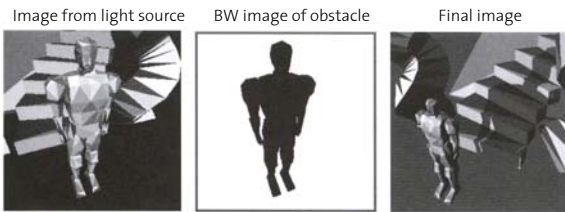


Figure from Moller & Haines "Real Time Rendering"

37

7. Texture Mapping



## Shadow Maps

- In Renderman (High-end production software)
- In Games (GPUs)



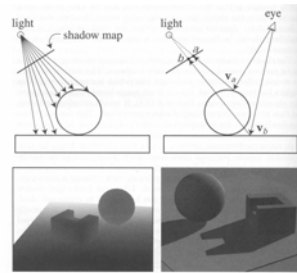
38

7. Texture Mapping



## Shadow Mapping

- Texture mapping with depth information
- Requires 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*



Foley et al. "Computer Graphics Principles and Practice"

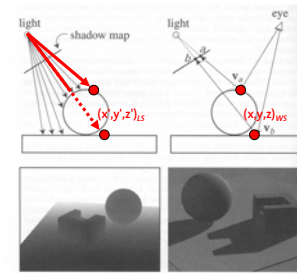
39

7. Texture Mapping



## 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  $4 \times 4$  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"

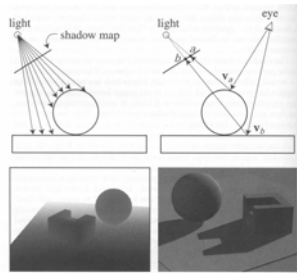
40

7. Texture Mapping



## Limitations of Shadow Maps

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



Foley et al. "Computer Graphics Principles and Practice"

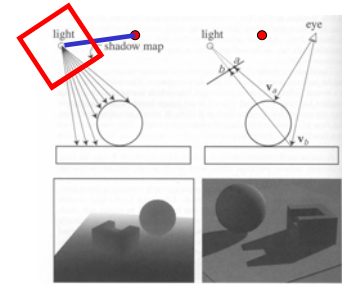
41

7. Texture Mapping



## 1. 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!



Foley et al. "Computer Graphics Principles and Practice"

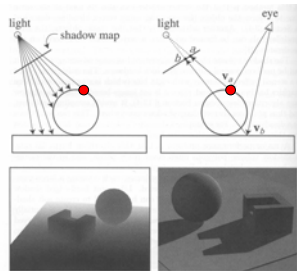
42

7. Texture Mapping



## 2. The Bias (Epsilon) Nightmare

- For a point visible from the light source  
 $\text{ShadowMap}(x',y') \approx z'$
- How can we avoid erroneous self-shadowing?
  - Add bias (epsilon)



Foley et al. "Computer Graphics Principles and Practice"

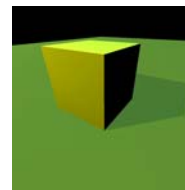
43

7. Texture Mapping

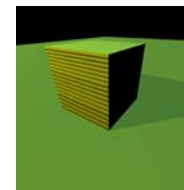


## 2. Bias (Epsilon) for Shadow Maps

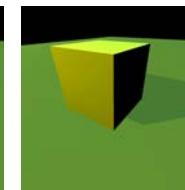
- $\text{ShadowMap}(x',y') + \text{bias} < z'$
- Choosing a good bias value can be very tricky



Correct image



Not enough bias



Way too much bias

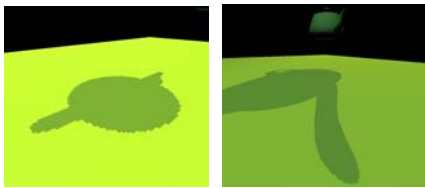
44

7. Texture Mapping



### 3. Shadow Map Aliasing

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

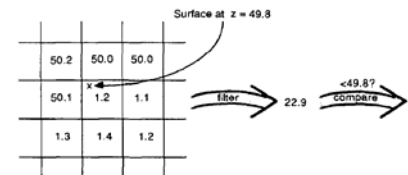


45



### 3. Shadow Map Filtering

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



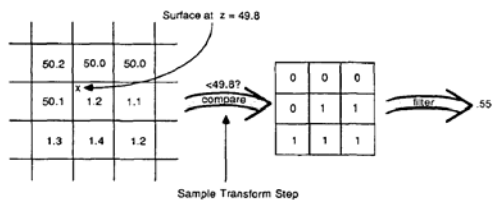
a) Ordinary texture map filtering. Does not work for depth maps.

46



### 3. Percentage Closer Filtering

- Instead filter the result of the test  
(weighted average of comparison results)
- But makes the bias issue more tricky



47



### 3. Percentage Closer Filtering

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



48



## 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

49

7. Texture Mapping



## Shadows in Production

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



Figure 12. Point Area Light [1]

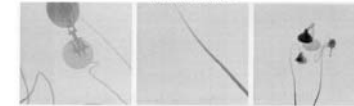


Figure 13. Shadow Map From Area [1]

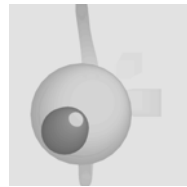
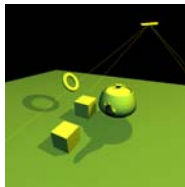
50

7. Texture Mapping



## Hardware Shadow Maps

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



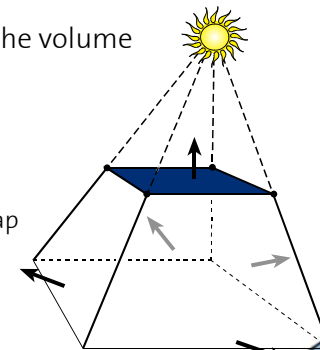
51

7. Texture Mapping



## 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



52

7. Texture Mapping

**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:  
 $\#polygons * \#lights$

53  
7. Texture Mapping

**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  $\neq 0$ , the point is in shadow

54  
7. Texture Mapping

**Optimizing Shadow Volumes**

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

55  
7. Texture Mapping

**Limitations of Shadow Volumes**

- Introduces a lot of new geometry
- Expensive to rasterize long skinny triangles
- Objects must be watertight to use silhouette trick
- Rasterization of polygons sharing an edge must not overlap & must not have gap

56  
7. Texture Mapping

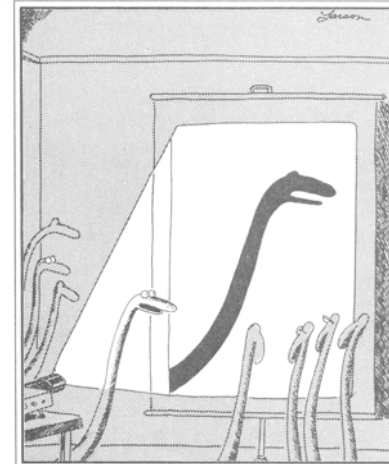


## Homework

Features / Limitations	Planar Fake Shadows	Projective Texture Shadows	Shadow Maps	Shadow Volumes
Allows objects to cast shadows on themselves (self shadowing)				
Permits shadows on arbitrary surfaces (i.e. curved)				
Renders geometry from the viewpoint of the light				
Generates extra geometric primitives				
Limited resolution of intermediate representation can result in jaggy shadow artifacts				

57

7. Texture Mapping



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

58

7. Texture Mapping