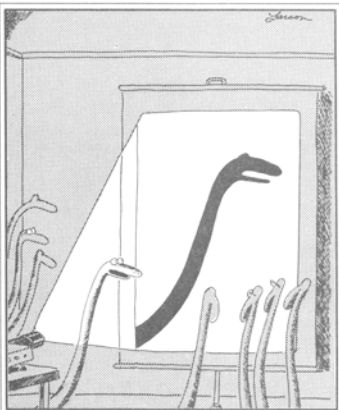


### Shadows

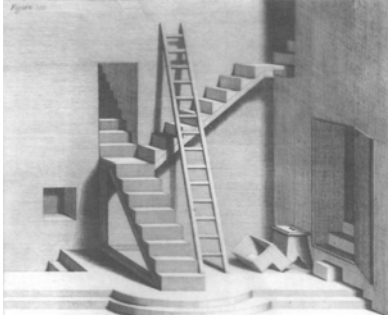
Thanks to:  
Frédéric Durand  
and Seth Teller  
MIT



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

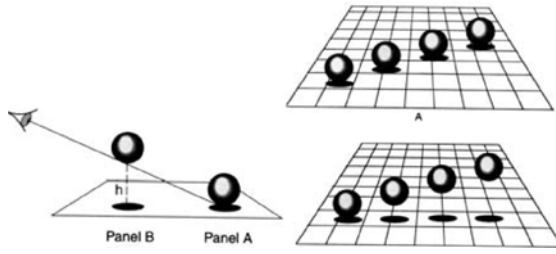
### Shadows

- Realism
- Depth cue



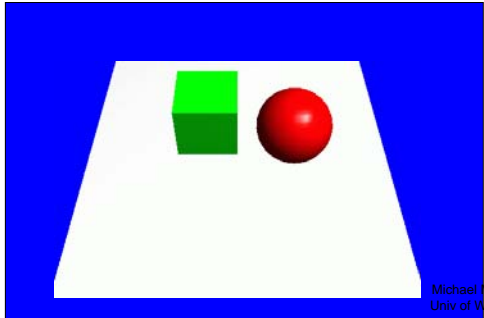
2

### Shadows as depth cue



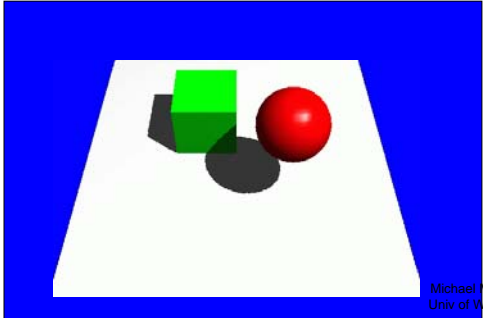
3

### Spatial relationship between objects



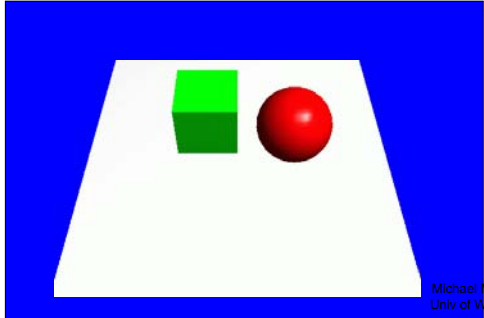
Michael McCool  
Univ of Waterloo

### Spatial relationship between objects



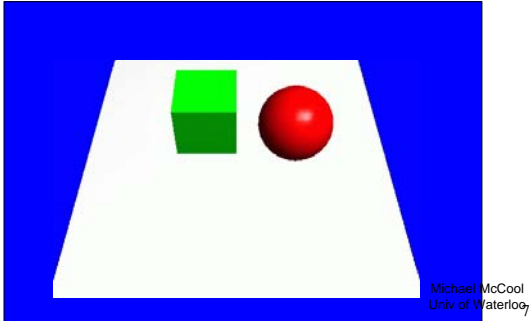
Michael McCool  
Univ of Waterloo

### Spatial relationship between objects



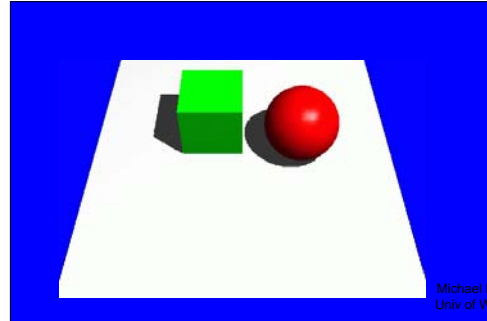
Michael McCool  
Univ of Waterloo

## Spatial relationship between objects



Michael McCool  
Univ of Waterloo

## Spatial relationship between objects



Michael McCool  
Univ of Waterloo

## Shadows add visual acuity



without



with

9

## Shadows and art

- Only in Western pictures (here Caravaggio)



10

## Shadows and art

- Shadows as the origin of painting
- People painted by tracing the shadows onto the canvas/wall

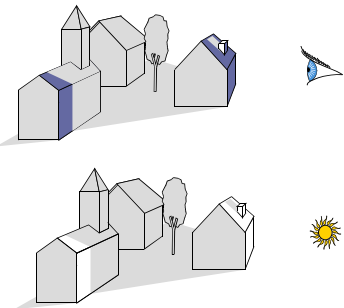


Plate 18  
David Allan,  
*The Origin of Painting*  
(*The Maid of Corinth*),  
1775. Oil on wood,  
38.7 x 31 cm. Edinburgh,  
National Gallery of  
Scotland.

11

## Duality shadow-view

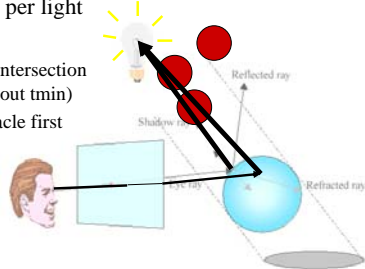
- A point is lit if it is visible from the light source
- Shadow computation very similar to view computation



12

## Shadow ray

- Ray from visible point to light source
- If blocked, discard light contribution
- One shadow ray per light
- Optimization?
  - Stop after first intersection (don't worry about  $t_{min}$ )
  - Test latest obstacle first



## Ray-casting shadows

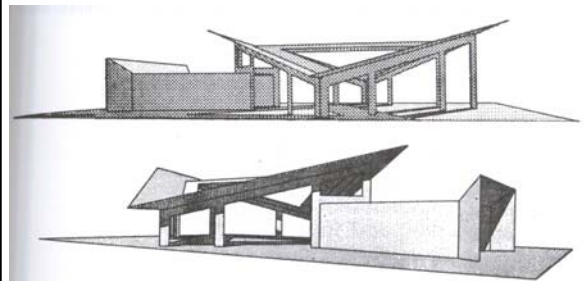


Fig. 16.52 Early pictures rendered with ray tracing. (Courtesy of Arthur Appel, IBM T.J. Watson Research Center.)

14

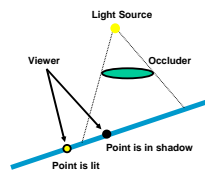
## Local vs. Global Illumination

- Core OpenGL uses local illumination model
  - Light sources used: distant, point, spot
  - Pros:
    - Fast calculation of pixel color using Phong, only needs
      - Position & direction of light source
      - Material properties (diffuse, specular, ambient coeff)
      - Light settings (intensity, fall-off, color)
      - ...but no info about other objects in the scene !
    - Each primitive can be processed *independently* of others
  - Cons:
    - Good looking images need global effects
      - Reflected light (e.g. environment mapping)
      - **Shadows**

15

## Global: Shadows

- A point is in shadow if the light got blocked between the light source and point

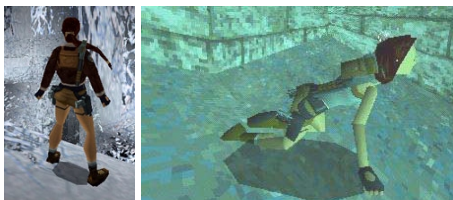


- Need mask that contains information about blocked / non blocked pixels

16

## Fake methods

- Still (not so) commonly used in games



Images from TombRaider. ©Eidos Interactive.

- Shadows are simple, hand-drawn polygons
  - No global effect
- ...but better than no shadow at all ☺

17

## Shadow Quality: "Blobs"

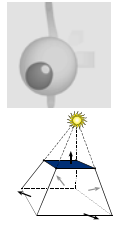


Lame "Blob" shadow

18

## Overview

- Projected Shadows
- Shadow map
  - Image-precision, texture mapping
- Shadow volume
  - Object space
- Soft shadows



19

## Planar Projection

- Render a ground-plane
- Render an object
- Then render the object again, but this time
  - Projected onto the plane
  - Without light, so that the shadow is black
  - Half transparent (using blending), to avoid completely dark shadows
  - Avoid multiple “darkening” on one spot by using ordinary z-buffer checks

20

## Projected Geometry

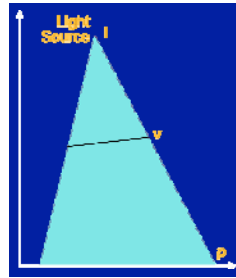
- [Blinn88] *Me and my fake shadow*
  - Shadows for selected large receiver polygons
    - Ground plane
    - Walls



21

## Projected Geometry

- Example:  $xz$  plane at  $y=0$



$$p_x = \frac{l_y v_x - l_x v_y}{l_y - v_y}$$

$$p_z = \frac{l_y v_z - l_z v_y}{l_y - v_y}$$

22

## Projected Geometry

- Transformation as 4 by 4 matrix

$$\bar{p} = \begin{pmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix}$$

23

## Projected Geometry

- General case: receiver polygon in plane  $E$

$$E: \vec{n} \cdot \vec{x} + d = 0$$

$$\bar{p} = \bar{l} - \frac{d + \vec{n} \cdot \bar{l}}{\vec{n} \cdot (\bar{v} - \bar{l})} (\bar{v} - \bar{l})$$

- 4x4 matrix

$$M = \begin{pmatrix} \mathbf{n} \cdot \mathbf{1} + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & \mathbf{n} \cdot \mathbf{1} + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & \mathbf{n} \cdot \mathbf{1} + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & \mathbf{n} \cdot \mathbf{1} \end{pmatrix}$$

24

## Projected Geometry

- 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

25

## Planar Shadows



Shadow is projected into the plane of the floor.

26

## Constructing a Shadow Matrix

```

groundplane = N, N, N, D
void shadowMatrix(GLfloat shadowMat[4][4], GLfloat groundplane[4], GLfloat lightpos[4])
{
    GLfloat dot;
    /* Find dot product between light position vector and ground plane normal. */
    dot = groundplane[X] * lightpos[X] + groundplane[Y] * lightpos[Y] +
        groundplane[Z] * lightpos[Z] + groundplane[W] * lightpos[W];

    shadowMat[0][0] = dot - lightpos[X] * groundplane[X];
    shadowMat[1][0] = 0.f - lightpos[X] * groundplane[Y];
    shadowMat[2][0] = 0.f - lightpos[X] * groundplane[Z];
    shadowMat[3][0] = 0.f - lightpos[X] * groundplane[D];
    shadowMat[0][1] = 0.f - lightpos[Y] * groundplane[X];
    shadowMat[1][1] = dot - lightpos[Y] * groundplane[Y];
    shadowMat[2][1] = 0.f - lightpos[Y] * groundplane[Z];
    shadowMat[3][1] = 0.f - lightpos[Y] * groundplane[D];
    shadowMat[0][2] = 0.f - lightpos[Z] * groundplane[X];
    shadowMat[1][2] = 0.f - lightpos[Z] * groundplane[Y];
    shadowMat[2][2] = dot - lightpos[Z] * groundplane[Z];
    shadowMat[3][2] = 0.f - lightpos[Z] * groundplane[D];
    shadowMat[0][3] = 0.f - lightpos[W] * groundplane[X];
    shadowMat[1][3] = 0.f - lightpos[W] * groundplane[Y];
    shadowMat[2][3] = 0.f - lightpos[W] * groundplane[Z];
    shadowMat[3][3] = dot - lightpos[W] * groundplane[D];
}
    
```

27

## How to add shadows ?

- Can be done in two ways:
  - 1<sup>st</sup> method: Full illumination + darkening

$FB = DiffuseTex0 * ( Light0 + Light1 + Light2... )$   
*if pixel is in shadow (with respect to Light0)*  
 $FB = FB * 0.5$

This is wrong since the contribution of Light1,2 etc. is also affected !

28

## How to add shadows ?

- 2<sup>nd</sup> & correct method: Use mask for each light

$FB = DiffuseTex0 * ( Light0 * Mask0 + Light1 * Mask1 + Light2 * Mask2... )$

Mask values

- 0 if pixel is in shadow (with respect to Light X)
- 1 if pixel is lit by Light X
- 0...1 for pixels on shadow edge (soft shadow edge)

Accumulation of  $(Light0 * Mask0) + ...$  can be done using additive blending

29

## How to add shadows ?

- Algorithm
  - Render scene with ambient illumination only
  - For each light source
    - Render scene with illumination from this light only
    - Scale illumination by shadow mask
    - Add up contribution to frame buffer
- Expensive but nearly correct !
- Speed-Up
  - Use more lights & masks in one pass
    - Masks stored as textures
    - Apply masks & sum up using fragment shaders

30

## How to Render the Shadow

```

/* Render 50% black shadow color on top of whatever
the floor appearance is. */
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,
            GL_ONE_MINUS_SRC_ALPHA);
glDisable(GL_LIGHTING); /* Force the 50% black. */
glColor4f(0.0, 0.0, 0.0, 0.5);

glPushMatrix();

/* Project the shadow. */
glMultMatrixf((GLfloat *) floorShadow);
drawDinosaur();
glPopMatrix();
    
```

31

## Not Quite So Easy (1)

Without stencil to avoid double blending of the shadow pixels:



Notice dark spots on the planar shadow.

Solution: Clear stencil to zero. Draw floor with stencil of one. Draw shadow if stencil is one. If shadow's stencil test passes, set stencil to two. No double blending.

32

## Not Quite So Easy (2)

There's still another problem even if using stencil to avoid double blending.

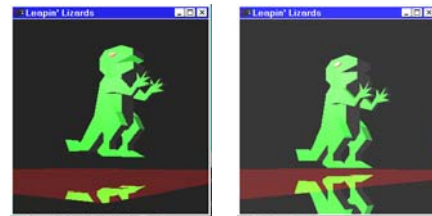


depth buffer Z fighting artifacts

Shadow fights with depth values from the floor plane. Use polygon offset to raise shadow polygons slightly in Z.

33

## Not Quite so Easy (3)



Good. Notice right image's reflection falls off the floor! Same problem with Shadows!

34

## Planar Projection

- Fast
  - Can be done with a matrix operation
- Easy
  - Just use the Modelview transform
- Very unrealistic
  - Just planar shadows

35

## Projected Geometry

- Problems
  - Z-Fighting
    - Use bias when rendering shadow polygons
    - Use stencil buffer (no 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)

36

## Fake shadows using textures

- Separate occluder and receiver
- Compute b/w image of obstacle from light
- Use projective textures



Image from light source BW image of obstacle Final image

Figure from Moller & haines "Real Time Rendering"  
37

## Fake shadows using textures

- Limitations?



Image from light source BW image of obstacle Final image

Figure from Moller & haines "Real Time Rendering"  
38

## Shadow maps

- Use texture mapping but using depth
- 2 passes (at least)
  - Compute shadow map from light source
    - Store depth buffer (shadow map)
  - Compute final image
    - Look up the shadow map to know if points are in shadow

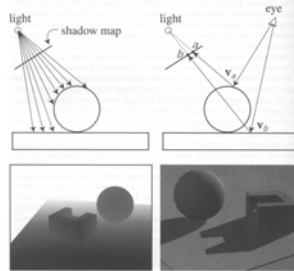


Figure from Foley et al. "Computer Graphics Principles and Practice"  
39

## Shadow map look up

- We have a 3D point  $x, y, z$
- How do we look up the shadow map?

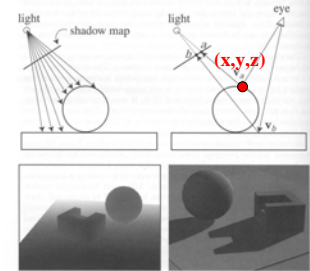


Figure from Foley et al. "Computer Graphics Principles and Practice"  
40

## Shadow map look up

- We have a 3D point  $x, y, z$
- How do we look up the shadow map?
- Use the 4x4 camera matrix from the light source
- We get  $(x', y', z')$
- Test:  $\text{ShadowMap}(x', y') < z'$

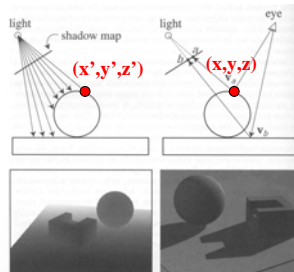


Figure from Foley et al. "Computer Graphics Principles and Practice"  
41

## Shadow maps

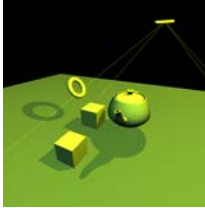
- In Renderman
  - (High-end production software)



42

## Shadow maps

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



43

## Introducing Another Technique: Shadow Mapping

- Image-space shadow determination
  - Lance Williams published the basic idea in 1978
    - By coincidence, same year Jim Blinn invented bump mapping (a great vintage year for graphics)
  - Completely image-space algorithm
    - means no knowledge of scene's geometry is required
    - must deal with aliasing artifacts
  - Well known software rendering technique
    - Pixar's RenderMan uses the algorithm
    - Basic shadowing technique for Toy Story, etc.

44

## Shadow Mapping References

- Important SIGGRAPH papers
  - Lance Williams, "Casting Curved Shadows on Curved Surfaces," SIGGRAPH 78
  - William Reeves, David Salesin, and Robert Cook (Pixar), "Rendering antialiased shadows with depth maps," SIGGRAPH 87
  - Mark Segal, et. al. (SGI), "Fast Shadows and Lighting Effects Using Texture Mapping," SIGGRAPH 92

45

## The Shadow Mapping Concept (1)

- Depth testing from the light's point-of-view
  - Two pass algorithm
  - First, render depth buffer from the light's point-of-view
    - the result is a "depth map" or "shadow map"
    - essentially a 2D function indicating the depth of the closest pixels to the light
  - This depth map is used in the second pass

46

## The Shadow Mapping Concept (2)

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

47

## The Shadow Mapping Concept (3)

- The Shadow Map Comparison
  - Two values
    - $A = Z$  value from depth map at fragment's light XY position
    - $B = Z$  value of fragment's XYZ light position
  - If B is greater than A, then there must be something closer to the light than the fragment
    - then the fragment is shadowed
  - If A and B are approximately equal, the fragment is lit

48



### Shadow Mapping with a Picture in 2D (1)

The  $A < B$  shadowed fragment case

light source

depth map image plane

depth map  $Z = A$

eye position

eye view image plane, a.k.a. the frame buffer

fragment's light  $Z = B$

49

### Shadow Mapping with a Picture in 2D (2)

The  $A \geq B$  unshadowed fragment case

light source

depth map image plane

depth map  $Z = A$

eye position

eye view image plane, a.k.a. the frame buffer

fragment's light  $Z = B$

50

### Shadow Mapping with a Picture in 2D (3)

Note image precision mismatch!

The depth map could be at a different resolution from the framebuffer

This mismatch can lead to artifacts

51

### Visualizing the Shadow Mapping Technique (1)

- A fairly complex scene with shadows

the point light source

52

### Visualizing the Shadow Mapping Technique (2)

- Compare with and without shadows

with shadows

without shadows

53

### Visualizing the Shadow Mapping Technique (3)

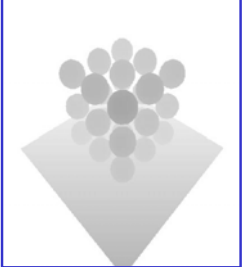
- The scene from the light's point-of-view

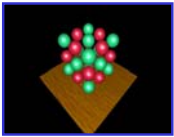
FYI: from the eye's point-of-view again

54

### Visualizing the Shadow Mapping Technique (4)

- The depth buffer from the light's point-of-view



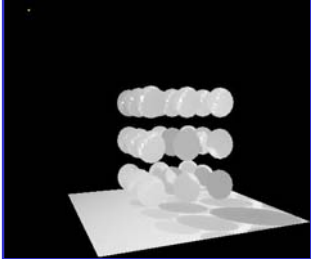



*FYI: from the light's point-of-view again*

55

### Visualizing the Shadow Mapping Technique (5)

- Projecting the depth map onto the eye's view



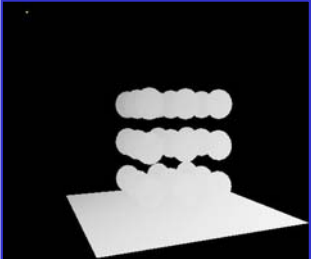


*FYI: depth map for light's point-of-view again*

56

### Visualizing the Shadow Mapping Technique (6)

- Projecting light's planar distance onto eye's view

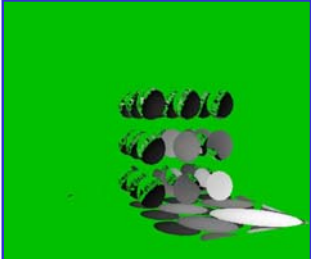


57

### Visualizing the Shadow Mapping Technique (6)

- Comparing light distance to light depth map

*Green is where the light planar distance and the light depth map are approximately equal*



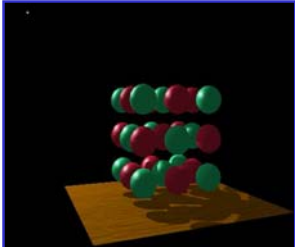
*Non-green is where shadows should be*

58

### Visualizing the Shadow Mapping Technique (7)

- Scene with shadows


*Notice how specular highlights never appear in shadows*



*Notice how curved surfaces cast shadows on each other*

59

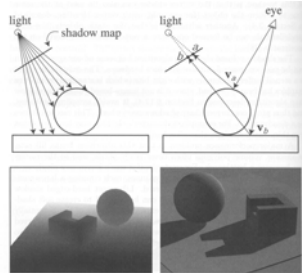
### Shadow Quality: Shadow Maps



60

## Problems with shadow maps?

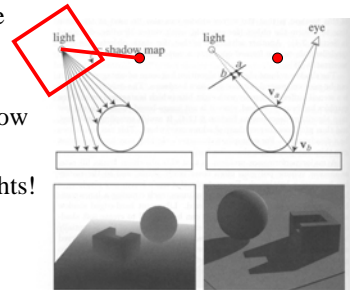
- Field of view
- Bias
- Aliasing



61

## Field of view problem

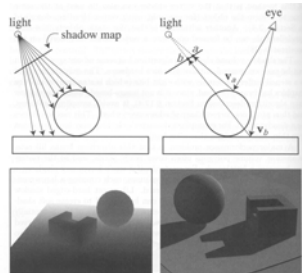
- What if point to shadow is outside field of view of shadow map?
- Use cubical shadow map
- Use only spot lights!



62

## Problems with shadow maps?

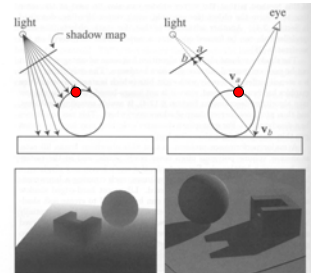
- Field of view
- Bias
- Aliasing



63

## The bias nightmare

- For a point visible from the light source
- $\text{ShadowMap}(x', y') \approx z'$
- Avoid erroneous self shadowing

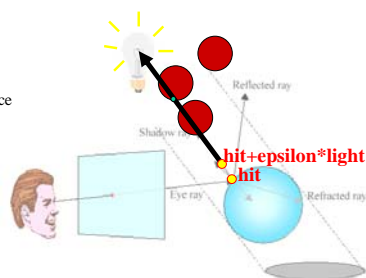


64

## The bias nightmare

- Shadow ray casting

- Start the ray at  $\text{hit} + \text{light} * \text{epsilon}$
- Add bias to avoid degeneracy
- Yet another instance of geometric robustness

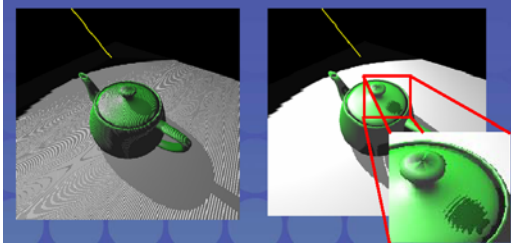


## Bias for shadow maps

$$\text{ShadowMap}(x', y') + \text{bias} < z'$$

Choosing the good bias value can be very tricky

- bias too small  $\rightarrow$  surface acne
- bias too large  $\rightarrow$  shadow leaks



66

## Construct Light View Depth Map

- Realizing the theory in practice
  - Constructing the depth map
    - use existing hardware depth buffer
    - **use `glPolygonOffset` to bias depth value**
    - read back the depth buffer contents (bind to a texture)
  - Depth map used as a 2D texture

67

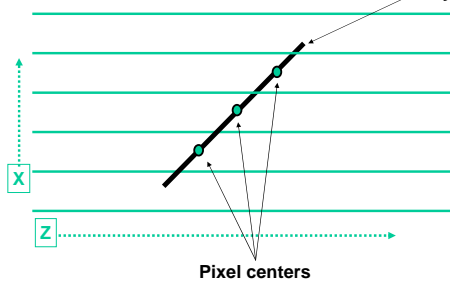
## Justification for `glPolygonOffset` When Constructing Shadow Maps

- Depth buffer contains “window space” depth values
  - Post-perspective divide means non-linear distribution
  - `glPolygonOffset` is guaranteed to be a window space offset
- Doing a “clip space” `glTranslatef` is not sufficient
  - Common shadow mapping implementation mistake
  - Actual bias in depth buffer units will vary over the frustum
  - No way to account for slope of polygon

68

## Sampling a Polygon’s Depth at Pixel Centers (1)

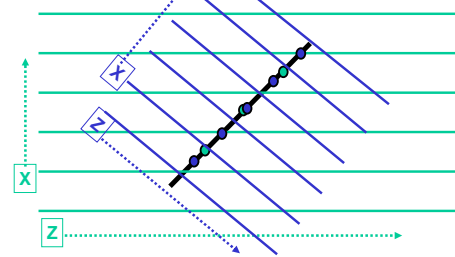
- Consider a polygon covering pixels in 2D



69

## Sampling a Polygon’s Depth at Pixel Centers (2)

- Consider a 2<sup>nd</sup> grid for the polygon covering pixels in 2D



70

## Sampling a Polygon’s Depth at Pixel Centers (3)

- How Z changes with respect to X



71

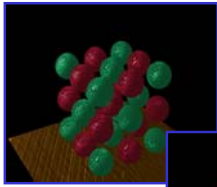
## Why You Need `glPolygonOffset`’s Slope

- Say pixel center is re-sampled to another grid
  - For example, the shadow map texture’s grid!
- The re-sampled depth could be off by  $\pm 0.5 \partial z / \partial x$  and  $\pm 0.5 \partial z / \partial y$
- The maximum absolute error would be  $|0.5 \partial z / \partial x| + |0.5 \partial z / \partial y| \approx \max(|\partial z / \partial x|, |\partial z / \partial y|)$ 
  - This assumes the two grids have pixel footprint area ratios of 1.0
  - Otherwise, we might need to scale by the ratio
- Exactly what polygon offset’s “slope” depth bias does

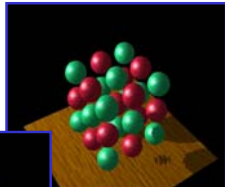
72

## Depth Map Bias Issues

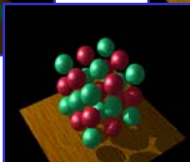
- How much polygon offset bias depends



*Too little bias, everything begins to shadow*



*Too much bias, shadow starts too far back*



*Just right*

73

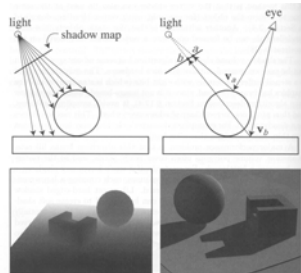
## Selecting the Depth Map Bias

- Not that hard
  - Usually the following works well
    - `glPolygonOffset(scale = 1.1, bias = 4.0)`
  - Usually better to error on the side of too much bias
    - adjust to suit the shadow issues in your scene
  - Depends somewhat on shadow map precision
    - more precision requires less of a bias
  - When the shadow map is being magnified, a larger scale is often required

74

## Problems with shadow maps?

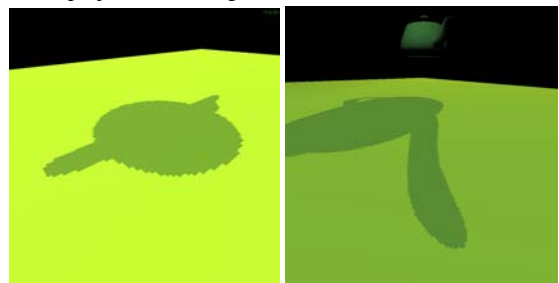
- Field of view
- Bias
- Aliasing



75

## Shadow map aliasing

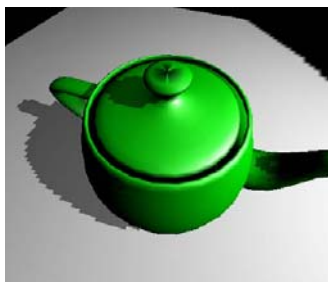
- Undersampling of shadow map
- Reprojection aliasing



76

## Aliasing

- Finite shadow map resolution
- Result: pixelized shadows



77

## Shadow maps

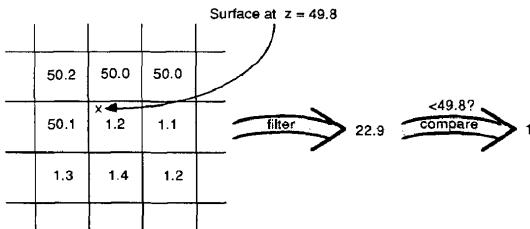
- In Renderman
  - (High-end production software)



78

## Shadow map filtering (PCF)

- Does not work!
- Filtering depth is not meaningful

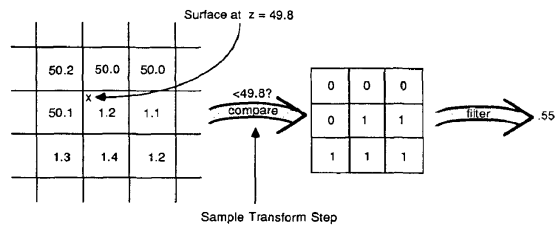


a) Ordinary texture map filtering. Does not work for depth maps.

79

## Percentage closer filtering

- Filter the result of the test
- But makes the bias issue more tricky



80

## Percentage closer filtering

- 5x5 samples
- Nice antialiased shadow
- Using a bigger filter produces fake soft shadows
- But makes the bias issue more tricky



81

## Shadows in production

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



Figure 12. Render from Larr 01.



Figure 13. Shadow maps from Larr 01.

## Movie Time!

83

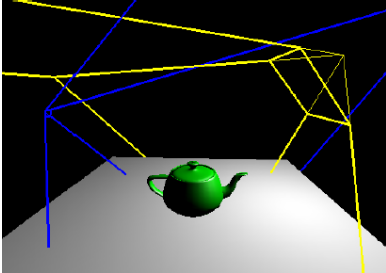
## Alaising

- Bad aliasing cases:
  - Large Scenes
    - High resolution shadow map required
  - Close-ups to shadow boundaries
    - Zoom in
  - Shadow stretches along the receiver

84

## Aliasing

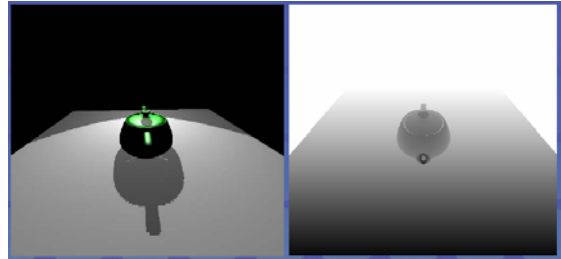
- Duelling frusta
  - Light shines opposite to viewing direction



85

## Aliasing

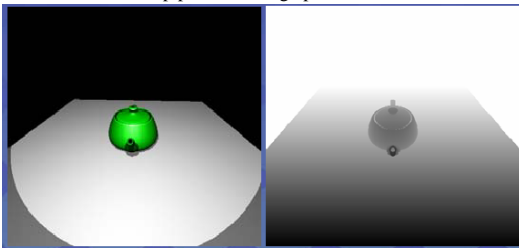
- Duelling frusta
  - Resolution mismatch



86

## Aliasing

- Miner's headlamp
  - Similar frusta
  - Similar sampling
  - One shadowmap pixel for image pixel



87

## Pros and Cons

- + general
  - everything that can be rendered can cast and receive a shadow
  - works together with vertex programs
- + fast
  - full hardware support
  - (almost) no overhead for static scenes
  - two passes needed for dynamic scenes
- + robust
- + easy to implement
- aliasing

88

## OpenGL Shadow Mapping

- Switch to other PPT slides

89

## Questions?



90