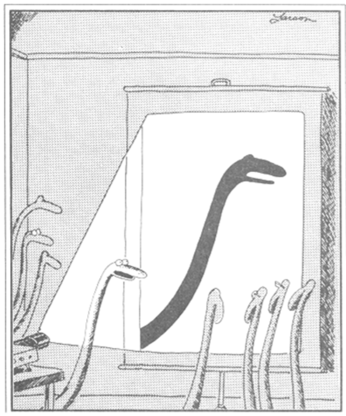
Shadows

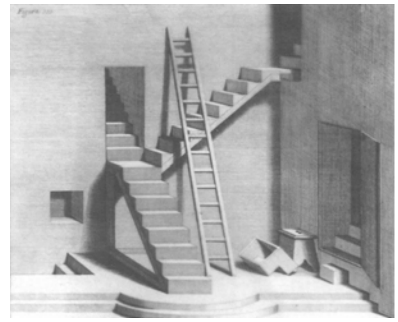
Thanks to:
Frédo Durand
and Seth Teller
MIT



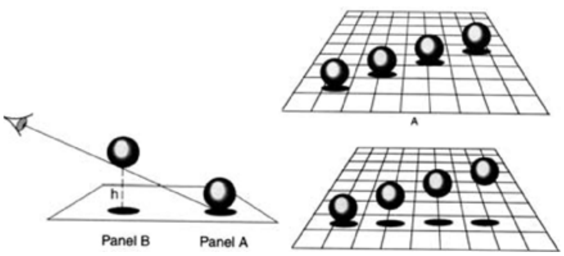
"Now this is…this is…well, I guess it's another snake."

Shadows

- Realism
- Depth cue

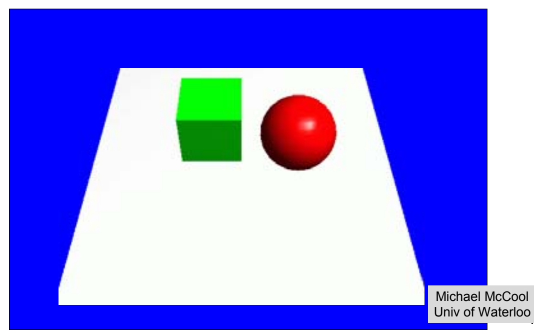


Shadows as depth cue

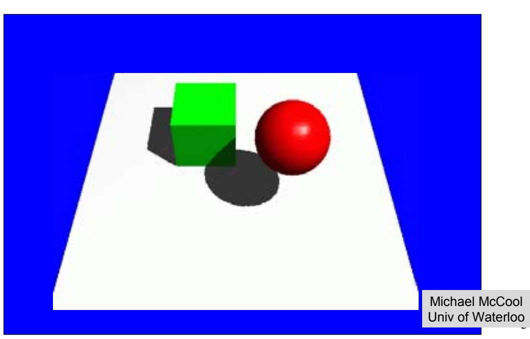


3

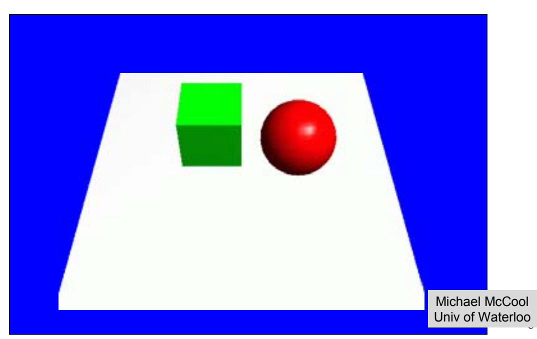
Spatial relationship between objects



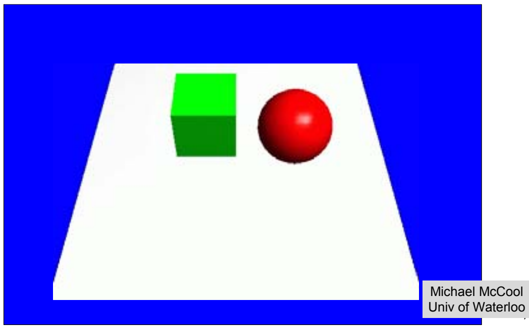
Spatial relationship between objects



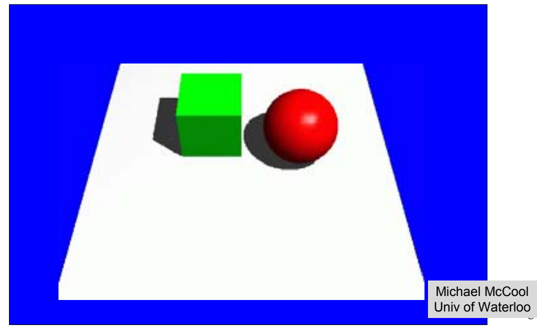
Spatial relationship between objects



Spatial relationship between objects



Spatial relationship between objects



Shadows and art

- Only in Western pictures (here Caravaggio)

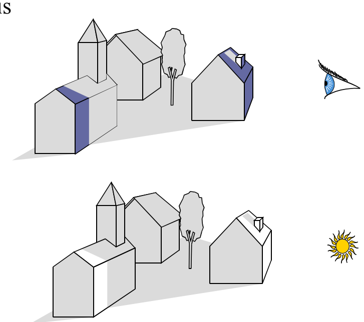
Shadows and art

- 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.

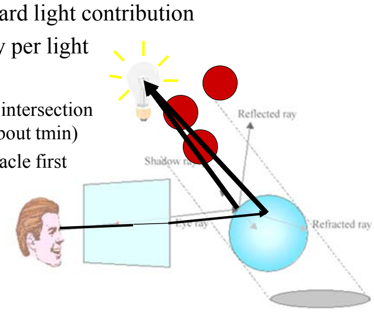
Duality shadow-view

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



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 tmin)
  - Test latest obstacle first



Ray-casting shadows

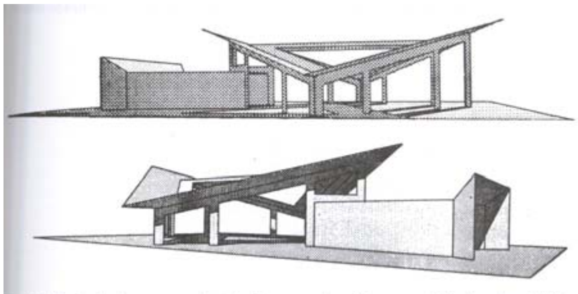


Fig. 16.52 Early pictures rendered with ray tracing. (Courtesy of Arthur Appel, IBM T.J. Watson Research Center.)

13

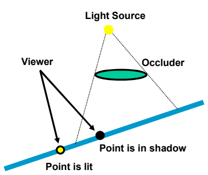
Local vs. Global Illumination

- 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**

14

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

15

Fake methods

- Still (not so) commonly used in games

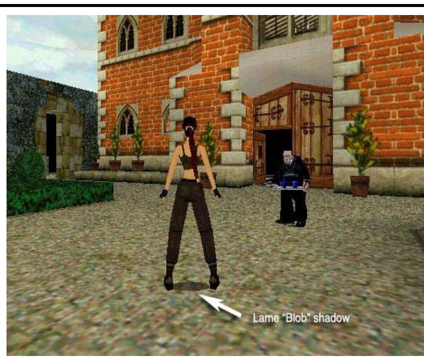
Images from TombRaider. ©Eidos Interactive.

- Shadows are simple, hand-drawn polygons
- No global effect

…but better than no shadow at all ☺

16

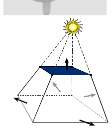
Shadow Quality: "Blobs"



17

Overview

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



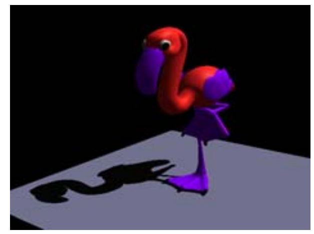
18

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

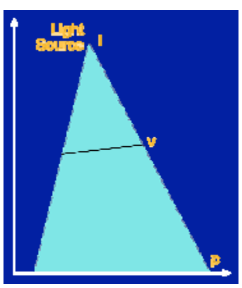
Projected Geometry

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



Projected Geometry

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



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

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

Projected Geometry

- Transformation as 4 by 4 matrix

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

Projected Geometry

- General case: receiver polygon in plane *E*

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

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

- 4x4 matrix

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

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

Planar Shadows

Shadow is projected into the plane of the floor.

25

Constructing a Shadow Matrix

groundplane = N_x N_y N_z 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];
}
```

26

How to add shadows ?

- Can be done in two ways:
  - 1st 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 !

27

How to add shadows ?

- 2nd & 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

28

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

29

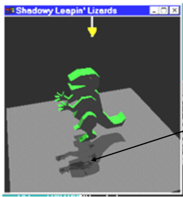
How to Render the Shadow

```
// 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);

PushMatrix(ModelView);    // save the state of the modelview matrix

// Project the shadow by pre-multiplying by the shadow matrix
lMultMatrixf((GLfloat *) floorShadow, ModelView);
drawDinosaur();
PopMatrix(ModelView);      // restore the modelview matrix
```

30

Not Quite So Easy (1)

Without stencil to avoid double blending of the shadow pixels:



Notice darks 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.

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.

Not Quite so Easy (3)

Good. Bad.
Notice right image's reflection falls off the floor!

Same problem with Shadows!

Planar Projection

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

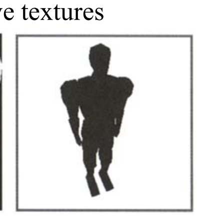
Projected Geometry

- 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)

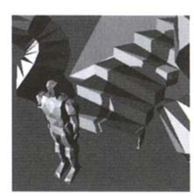
Fake shadows using textures

- 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"

Fake shadows using textures

- Limitations?

Image from light source BW image of obstacle Final image

Figure from Moller & haines "Real Time Rendering"

37

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.

38

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

39

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

40

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

41

The Shadow Mapping Concept (3)

- The Shadow Map Comparison
  - Two values
    - A = Z value of fragment's XYZ light position
    - B = Z value from depth map at fragment's light XY position
    - A = Z value of fragment's XYZ light position
  - If A is less than B, 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

42

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

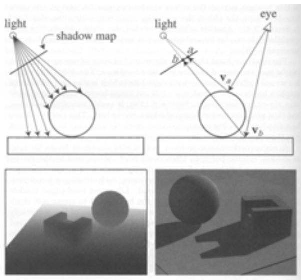


Figure from Foley et al. "Computer Graphics Principles and Practice"

Shadow map look up

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

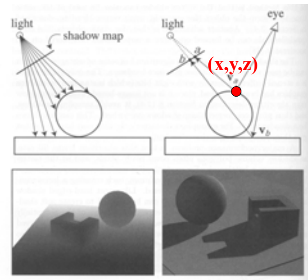


Figure from Foley et al. "Computer Graphics Principles and Practice"

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: ShadowMap(x',y')<z'

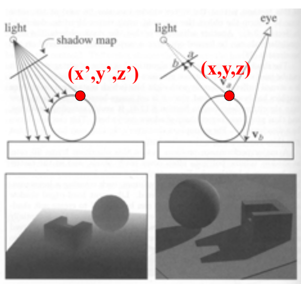


Figure from Foley et al. "Computer Graphics Principles and Practice"

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: ShadowMap(x',y')<z'

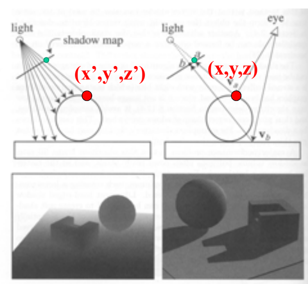
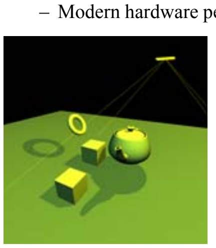
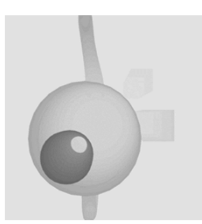


Figure from Foley et al. "Computer Graphics Principles and Practice"

Shadow maps

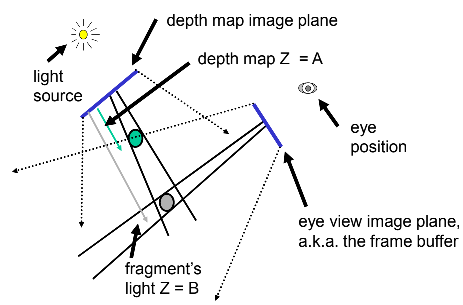
- Can be done in hardware
- Using hardware texture mapping
  - Texture coordinates u,v,w generated using 4x4 matrix
  - Modern hardware permits tests on texture values





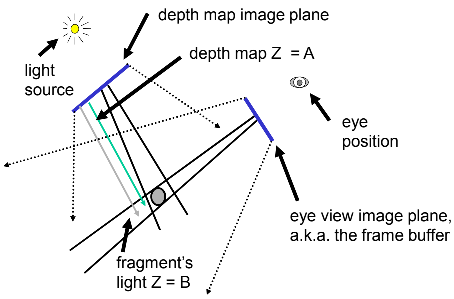
Shadow Mapping with a Picture in 2D (1)

The A < B shadowed fragment case



Shadow Mapping with a Picture in 2D (2)

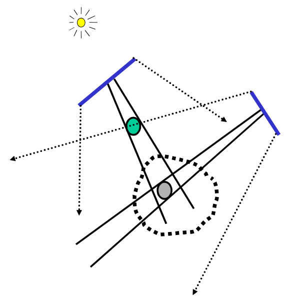
The A ≅ B unshadowed fragment case



49

Shadow Mapping with a Picture in 2D (3)

Note image precision mismatch!



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

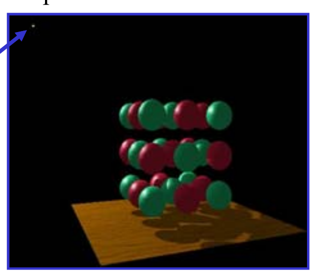
This mismatch can lead to artifacts

50

Visualizing the Shadow Mapping Technique (1)

- A fairly complex scene with shadows

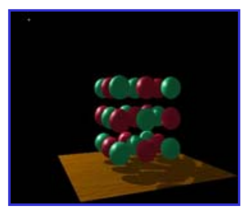
the point light source



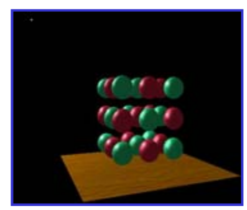
51

Visualizing the Shadow Mapping Technique (2)

- Compare with and without shadows



with shadows

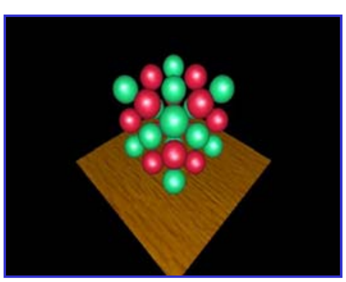


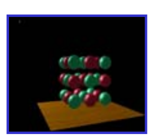
without shadows

52

Visualizing the Shadow Mapping Technique (3)

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



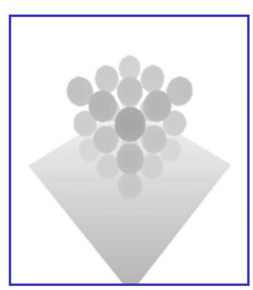


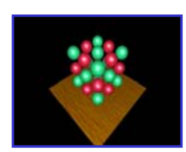
FYI: from the eye's point-of-view again

53

Visualizing the Shadow Mapping Technique (4)

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



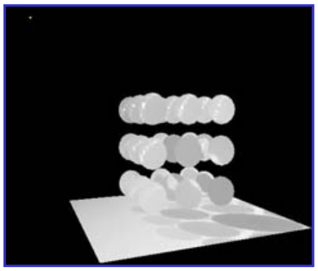


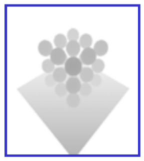
FYI: from the light's point-of-view again

54

Visualizing the Shadow Mapping Technique (5)

- Projecting the depth map onto the eye's view



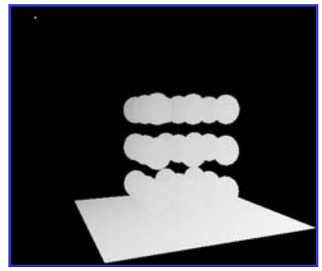


FYI: depth map for light's point-of-view again

55

Visualizing the Shadow Mapping Technique (6)

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



56

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 approximatel y equal

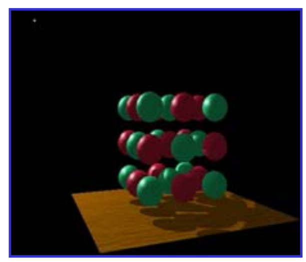
Non-green is where shadows should be

57

Visualizing the Shadow Mapping Technique (7)

- Scene with shadows

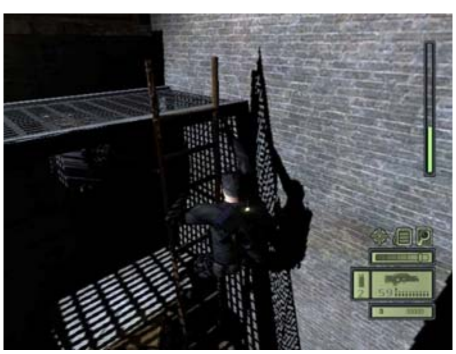
Notice how specular highlights never appear in shadows



Notice how curved surfaces cast shadows on each other

58

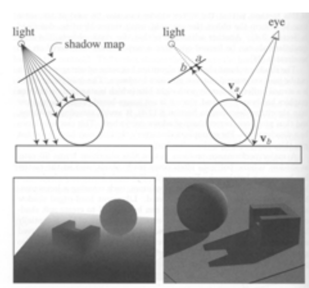
Shadow Quality: Shadow Maps



59

Problems with shadow maps?

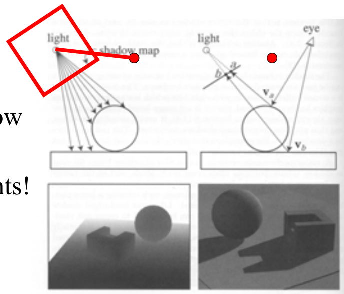
- Field of view
- Bias
- Aliasing



60

Field of view problem

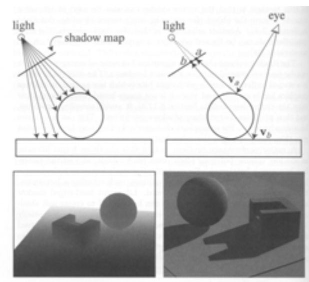
- What if point to shadow is outside field of view of shadow map?
- Use cubical shadow map
- Use only spot lights!



61

Problems with shadow maps?

- Field of view
- Bias
- Aliasing



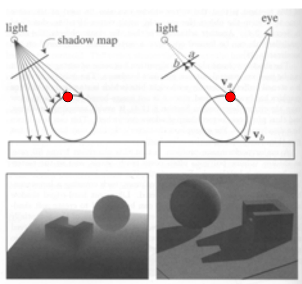
62

The bias nightmare

- For a point visible from the light source

ShadowMap(x’,y’)≈z’

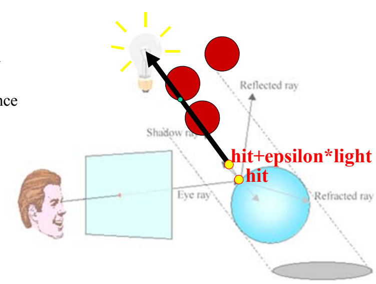
- Avoid erroneous self shadowing



63

The bias nightmare

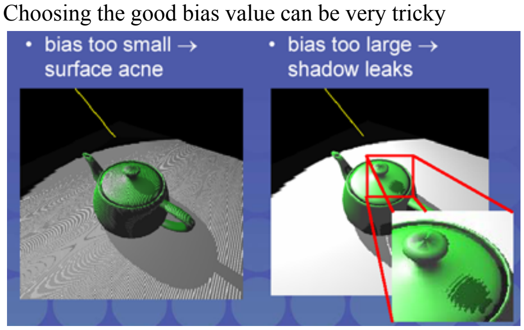
- Shadow ray casting
  - Start the ray at hit+light*epsilon
  - Add bias to avoid degeneracy
  - Yet another instance of geometric robustness



Bias for shadow maps

ShadowMap(x’,y’)+bias < z’

Choosing the good bias value can be very tricky



65

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

66

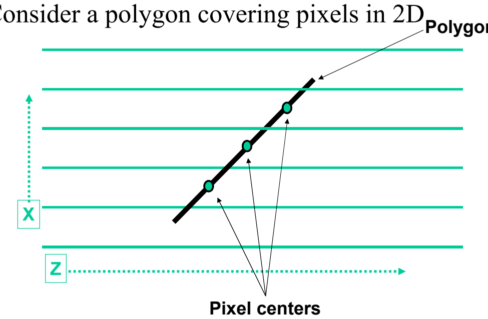
Justification for glPolygonOffset When Constructing Shadow Maps

- 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" translate 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

67

Sampling a Polygon's Depth at Pixel Centers (1)

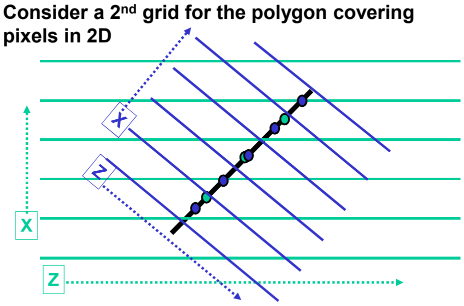
- Consider a polygon covering pixels in 2D



68

Sampling a Polygon's Depth at Pixel Centers (2)

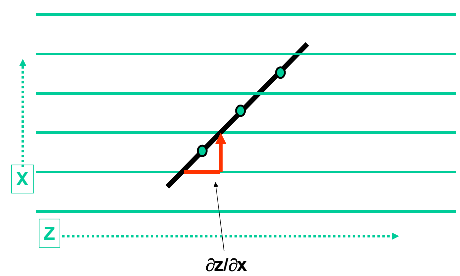
- **Consider a 2nd grid for the polygon covering pixels in 2D**



69

Sampling a Polygon's Depth at Pixel Centers (3)

- **How Z changes with respect to X**



70

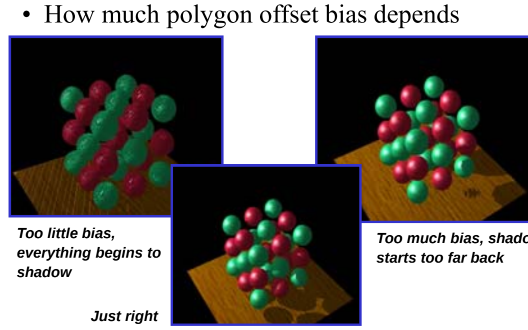
Why You Need glPolygonOffset's Slope

- 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
  $+/-0.5\ \partial z/\partial x$ and $+/-0.5\ \partial z/\partial y$
- 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

71

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

72

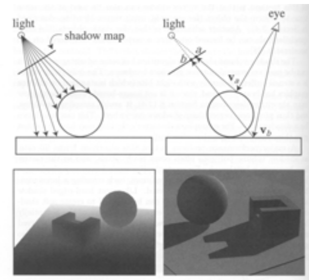
Selecting the Depth Map Bias

- 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

73

Problems with shadow maps?

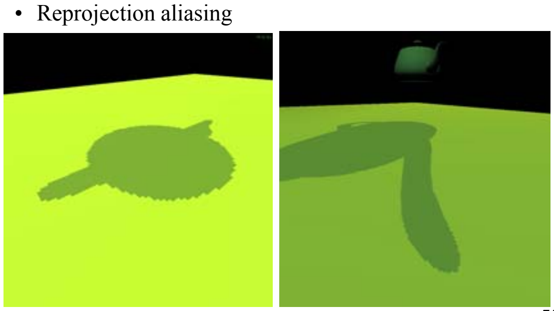
- Field of view
- Bias
- Aliasing



74

Shadow map aliasing

- Undersampling of shadow map
- Reprojection aliasing



75

Alaising

- Finite shadow map resolution
- Result: pixelized shadows

76

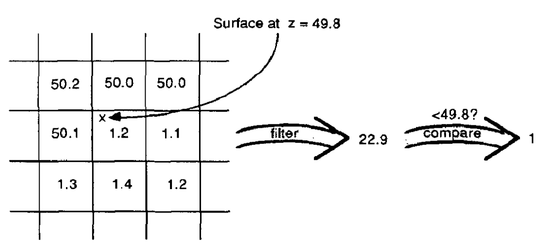
Shadow maps

- In Renderman
  - (High-end production software)

77

Shadow map filtering (PCF)

- Does not work!
- Filtering depth (tri-linear) is not meaningful

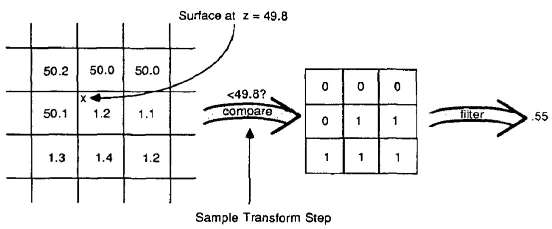


a) Ordinary texture map filtering. Does not work for depth maps.

78

Percentage closer filtering

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



79

Percentage closer filtering

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

80

Shadows in production

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

Figure 12. Frame from Luxo Jr.

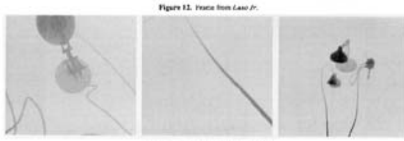


Figure 13. Shadow maps from Luxo Jr.

Movie Time!

82

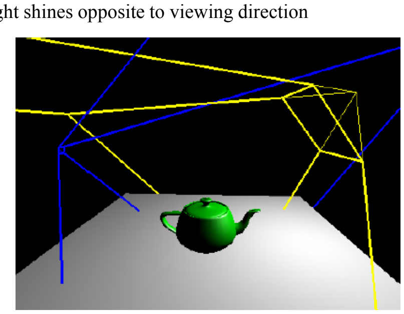
Aliasing

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

83

Aliasing

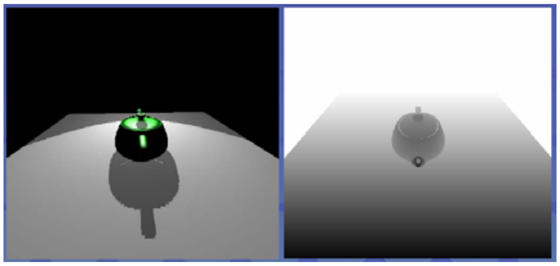
- Duelling frusta
  - Light shines opposite to viewing direction



84

Aliasing

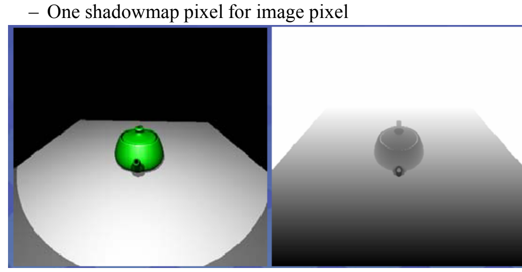
- Duelling frusta
  - Resolution mismatch



85

Aliasing

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



86

Pros and Cons

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

87

OpenGL Shadow Mapping

- Switch to other PPT slides

88

Questions?

89