Shadows

- Shadow Algorithms
  - Planar Shadows
  - Shadow Maps
  - Shadow Volumes
- Radiosity Shadows
- Ray Traced Shadows
- Light Maps
- Projective Texture Shadows

Shadows

- Regions of a scene not completely visible by the light sources
- One of the most important clues about the spatial relationship among objects in a scene

Shadows (Cont.)

- Depend only on the visibility from the light sources
- Can be computed as a pre-processing for static scenes
- Most common shadow algorithms are restricted to direct light and point or directional light sources
- Area light sources are usually approximated by several points lights

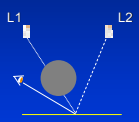
Illumination Function

$$I_\lambda = ambient + \sum_{l}^{lights} S_l f_{att\,l} I_{l\lambda}(diffuse + specular)$$

- Point Light Sources
  - S_l equals to 1 if the point is visible from the light source; 0, otherwise
- Area Light Sources
  - S_l is the fraction of the area light source to which the point is visible

Illumination Function (Cont.)

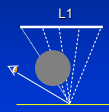
Multiple Point Light Sources



In shadow wrt L1, but lit by L2

$S_1 = 0, S_2 = 1$

Area Light Source

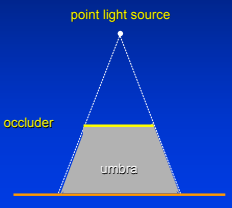


Partially in shadow

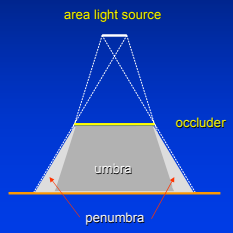
$S_1 = 0.4$

Shadow Regions

Point Light Sources
Usually Hard Shadows



Area Light Sources
Soft Shadows

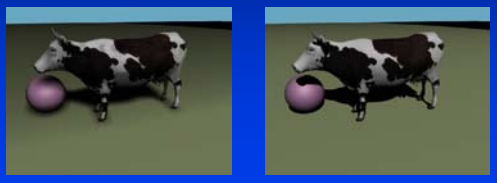


Soft x Hard Shadows

- Soft shadows
  - Produced by area light sources (umbra + penumbra)
  - Most common kind of shadow
  - More realistic effects (in general)

Soft x Hard Shadows (Cont.)

- Hard (Sharp) shadows
  - Produced by point light sources
  - More realistic effects for shadows cast by the sun or by a powerful light bulb placed far away from the object



Shadow Algorithms

- Object-based
  - Local illumination model
  - Area subdivision
  - Planar projection
  - Radiosity
  - Shadow-volumes
- Image-based
  - Shadow-maps
  - Layered attenuation Maps
  - Light maps
  - Projective texture shadows
- Hybrid
  - Ray-tracing

Local Illumination Model

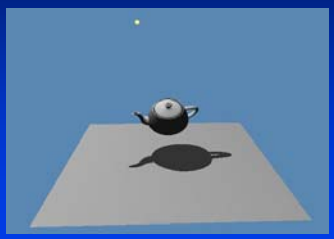
- Only accounts for some kinds of self-shadowing due to normals facing away from the light source
- Simplest to implement, it is supported in hardware

Planar Shadows

- Casting shadows of arbitrary objects onto arbitrary surfaces is non-trivial
- Approximate shadows can convey the right visual effect
- We can project shadows of polygonal models onto an arbitrary plane using a matrix
- Basically, the idea is to “project” (or smash) the polygonal models onto the plane

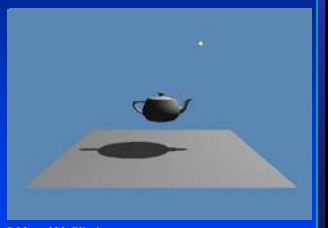
Planar Shadows (Cont.)

- Demo



Planar Shadows - Algorithm

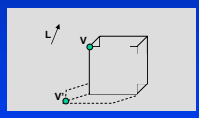
Compute the matrix *S* that projects the geometry onto the plane
Compute M' = S*M /* M is the ModelView matrix */
/* Render plane */
Set ModelView = M
Render the Plane
/* Render shadows */
Turn lighting & depth test off
Set shadow color
Set ModelView = M'
Render all objects
/* Render objects */
Turn lighting & depth test on
Set ModelView = M
For each object
  Set object color
  Render object



Shadow Matrix for Directional Light Source

- Directional light source and fixed plane, say, plane Y = 0
- A light source with direction L = [x_L, y_L, z_L] projects a vertex V = [x_V, y_V, z_V] on the plane Y = 0 at V' = [x_S, 0, z_S]
- The shadow ray starts at V and follows S = V - αL
- It hits the plane Y = 0 when

$$\alpha = \left(\frac{y_V}{y_L}\right) \Rightarrow \begin{cases} x_S = x_V - \alpha x_L \\ z_S = z_V - \alpha z_L \end{cases}$$



Shadow Matrix for Directional Light Source

- In matrix format
- $\begin{cases} x_S = x_V - \alpha x_L \\ y_S = 0 \\ z_S = z_V - \alpha z_L \end{cases}$ onde $\alpha = \left(\frac{y_V}{y_L}\right)$

$$\begin{bmatrix} x_s \\ 0 \\ z_s \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & (-x_L / y_L) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & (-z_L / y_L) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

Shadow Matrix for Perspective Shadows

- The shadow of a point V is cast in the direction V – L
  - Where L = [x_L, y_L, z_L] now represents the position of the light source in 3D
- A shadow point is given by S = V - α(V – L), for Y_S = 0
- Solving for α when Y_S = 0

$$\alpha = \left(\frac{y_V}{y_V - y_L}\right) \Rightarrow \begin{cases} x'_S = \frac{x_L y_V - x_V y_L}{y_V - y_L} \\ y'_S = 0 \\ z'_S = \frac{z_L y_V - z_V y_L}{y_V - y_L} \end{cases}$$

$$\begin{bmatrix} x_s \\ 0 \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} -y_L & x_L & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & z_L & -y_L & 0 \\ 0 & 1 & 0 & -y_L \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

This matrix produces negative *w* values if $y_L > y_V$, which is the typical case. This will lend to clipping after the perspective projection. Solution: multiply the matrix by -1

Shadow Matrix: General Case

- Let V and L be represented as homogeneous row vectors (in WCS)
- Let P be an arbitrary plane (in WCS), represented by a column vector
- A point along the line from V to L can be represented as αV + βL (I)
- The intersection between such a line and P is given by

$$\begin{aligned} (\alpha V + \beta L) \cdot P &= 0 \\ \alpha (V \cdot P) + \beta (L \cdot P) &= 0 \end{aligned}$$

- Which can be satisfied for

$$\begin{aligned} \alpha &= (L \cdot P) \\ \beta &= -(V \cdot P) \end{aligned}$$

Replacing α and β in (I)

- Thus, a shadow point is given by: $S = V(L \cdot P) - (V \cdot P)L$

Shadow Matrix: General Case

$$S = V(L \cdot P) - (V \cdot P)L$$

- We can reorder the terms of the in the shadow point equation as:

$$S = \underbrace{V}_{1\text{x}4}\underbrace{(L \cdot P)}_{1\text{x}1} - \underbrace{V}_{1\text{x}4} \cdot \underbrace{(PL)}_{4\text{x}4}$$

- Regrouping:

$$\begin{aligned} S &= V\left[(L \cdot P)I - (PL)\right] \\ S &= \left[(L \cdot P)I - \left(L^T P^T\right)\right] V^T \end{aligned}$$

Shadow Matrix: General Case

- From the expression below

$$S = \left[(L \cdot P)I - \left(L^T P^T\right)\right]V^T$$

- We can write a general Shadow Matrix as

$$S = \begin{bmatrix} L\cdot P & 0 & 0 & 0 \\ 0 & L\cdot P & 0 & 0 \\ 0 & 0 & L\cdot P & 0 \\ 0 & 0 & 0 & L\cdot P \end{bmatrix} - \begin{bmatrix} L_x \\ L_y \\ L_z \\ L_w \end{bmatrix} \begin{bmatrix} P_x & P_y & P_z & P_w \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ 1 \end{bmatrix}$$

$$S = \begin{bmatrix} L\cdot P - L_x P_x & -L_x P_y & -L_x P_z & -L_x P_w \\ -L_y P_x & L\cdot P - L_y P_y & -L_y P_z & -L_y P_w \\ -L_z P_x & -L_z P_y & L\cdot P - L_z P_z & -L_z P_w \\ -L_w P_x & -L_w P_y & -L_w P_z & L\cdot P - L_w P_w \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ 1 \end{bmatrix}$$

Shadow Mapping [Williams78]

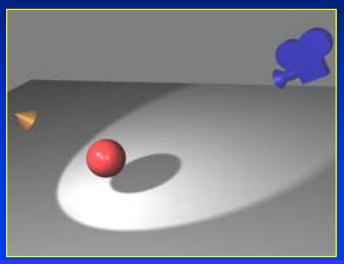
- Image-precision algorithm based on depth buffer values
- It can be applied to any surface that can be rasterized
- Easy to map to the graphics hardware using the texture sub-system

Shadow Mapping Algorithm

- Generate a depth map (shadow map) of the scene from the point of view of each light source
- For each pixel in the camera view
  - Project the 3-D coordinates of the surface visible through that pixel onto the "image plane" of each of the light sources
  - Use the (x,y) coordinates of the projection to index the light's shadow map
  - If the stored z_L (in the shadow map) is closer to the light source than the projected z value, the pixel is in shadow; otherwise, it is lit

Shadow Mapping

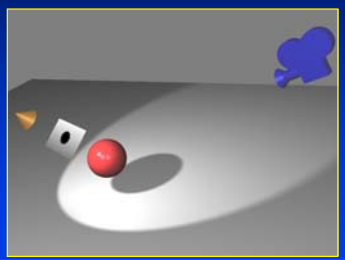
- Shadows are generated in two steps



Slide courtesy of Christian A. Pagot (PPGC - UFRGS 2005)

Shadow Mapping

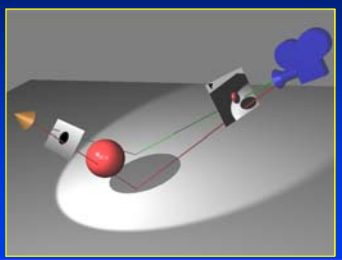
- 1° step: Generation of the Shadow Map



Slide courtesy of Christian A. Pagot (PPGC - UFRGS 2005)

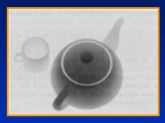
Shadow Mapping

- 2° step: Generation of the image with shadow



Slide courtesy of Christian A. Pagot (PPGC - UFRGS 2005)

Shadow Mappping - Example

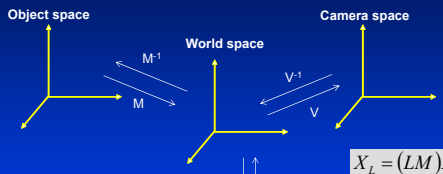


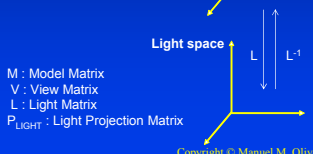
Shadow-map from light 1

Shadow-map from light 2

Camera view

Mapping to the Light CS





$$X_L = (LM)X_O$$

$$\begin{bmatrix} x_{sl} \\ y_{sl} \\ z_{sl} \\ w_{sl} \end{bmatrix} = (P_{Light}LM)X_O$$

M : Model Matrix
V : View Matrix
L : Light Matrix
P_{LIGHT} : Light Projection Matrix

Accessing the Shadow Map

- Shadow map is stored in a texture: coordinates are in [0,1] range
- The expression below will produce x, y and z values inside the canonical volume (after division by w) in the range [-1,1]

$$\begin{bmatrix} x_{sl} \\ y_{sl} \\ z_{sl} \\ w_{sl} \end{bmatrix} = (P_{Light}LM)X_O$$

- We adjust this performing a scale and a translation

$$\begin{bmatrix} x'_{sl} \\ y'_{sl} \\ z'_{sl} \\ w'_{sl} \end{bmatrix} = (SP_{Light}LM)X_O \quad \text{where} \quad S = \begin{bmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Accessing the Shadow Map

- The shadow map is then accessed using coordinates (in [01,]):

$$\left(\frac{x'_{sl}}{w'_{sl}}, \frac{y'_{sl}}{w'_{sl}}, \frac{z'_{sl}}{w'_{sl}} \right)$$

- Notice that the S matrix can applied either before or after the perspective division

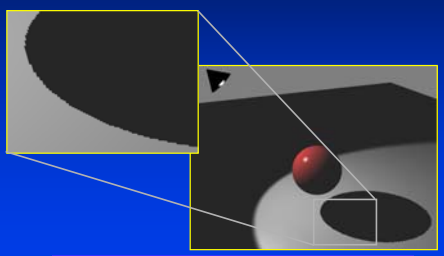
$$\begin{bmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{sl} \\ y_{sl} \\ z_{sl} \\ w_{sl} \end{bmatrix} = \begin{bmatrix} 0.5x_{sl} + 0.5w_{sl} \\ 0.5y_{sl} + 0.5w_{sl} \\ 0.5z_{sl} + 0.5w_{sl} \\ w_{sl} \end{bmatrix} \doteq \begin{bmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{sl}/w_{sl} \\ y_{sl}/w_{sl} \\ z_{sl/w_{sl}} \\ 1 \end{bmatrix}$$

Shadow Mapping: Problems

- Prone to aliasing
  - During the construction (point sampling)
  - During the access to the shadow map
    - A pixel from the camera's view might cover a large surface area, which in turn can map to several pixels in the shadow map
    - Addressed by [Reeves et al. 87]: The fraction of the projected area of the fragment in the shadow map corresponding to shadow is used to attenuate the light intensity
- Prone to self-shadowing (due to lack of precision)
  - Add some bias to the transformed z before testing

Shadow Mapping

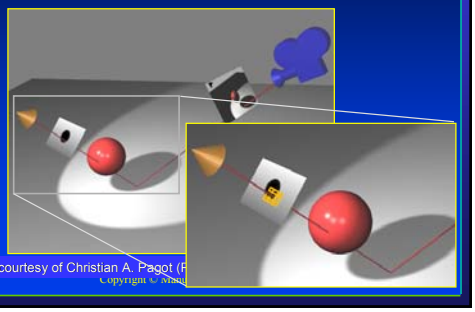
- Aliasing



Slide courtesy of Christian A. Pagot (PPGC - UFRGS 2005)

Percentage Closer Filtering [Reeves, 1987]

- Shadow test performed against an area of the shadow map



Images courtesy of Christian A. Pagot (P

The PCF Algorithm

Shadow Map

0.1	0.2	0.3
0.2	0.3	0.8
0.3	0.8	0.9

Shadow Test

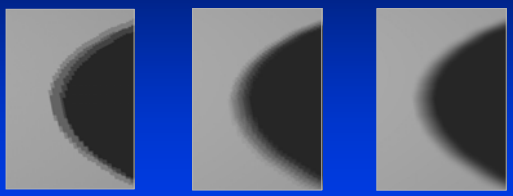
1	1	1
1	1	0
1	0	0

PCF $\frac{6}{9} = 0.667$

Slide courtesy of Christian A. Pagot (PPGC - UFRGS 2005)

Percentage Closer Filtering

- The quality of the antialiasing varies with the size of the PCF kernel (shape and number of samples)



PCF with 4 samples | PCF with 9 samples | PCF with 16 samples

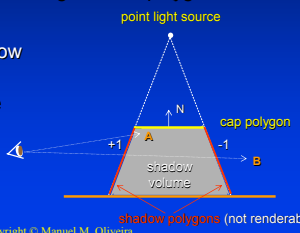
Slide courtesy of Christian A. Pagot (PPGC - UFRGS 2005)

Volumetric Shadows

- Set counter = # of shadow volumes containing the eye
- Trace rays from the eye and add +1 for each front facing and -1 for each back facing shadow polygon crossed

Counter = 0 → lit
Counter > 0 → shadow

Extension to multiple light sources is straightforward



Radiosity

- Shadows are determined by the form factors among the elements of the scene

Ray Tracing

- Trace rays from the surface point to each (point) light sources and check if such rays intersect any objects

Shadow cast by 1 out of 2 light sources

Shadow cast by a single light source

Light Map

- Used in game programming

Images courtesy of Alan Watt and Fabio Policarpo (3D Games: Real-time Rendering and Software Technology)

Projective Texture Mapping [Segal92]

- Imagine a light source as a projector

Projective Texture Shadows

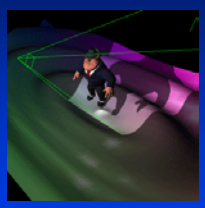
Light's point-of-view

Shadow projective texture (modulation image or light-map)

Eye's point-of-view, projective texture applied to ground-plane (self-shadowing is from another algorithm)

Reproduced from Kenny Hoff's lecture slides on Shadows (http://www.cs.unc.edu/~hoff/projects/comp236_ta/shadow_presentation/shadows.ppt)

Projective Texture Shadows



Reproduced from Kenny Hoff's lecture slides on Shadows (http://www.cs.unc.edu/~hoff/projects/comp236_ta/shadow_presentation/shadows.ppt)