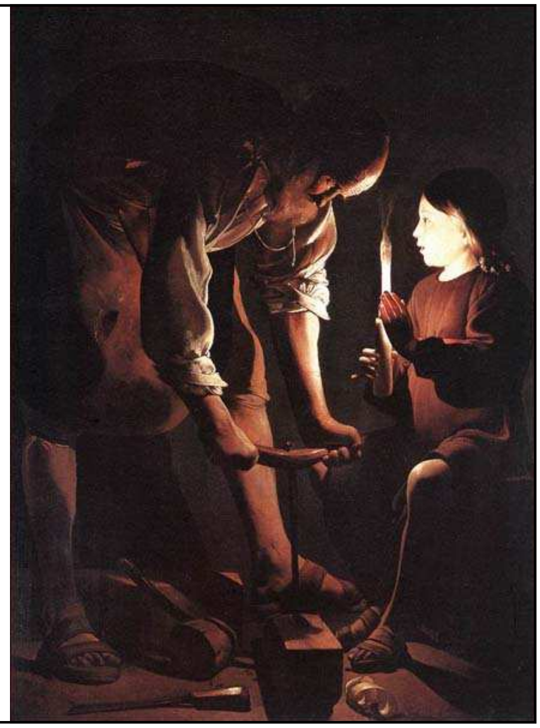
Shadows



(Georges de la Tour)

Slide 1

Why Shadows?

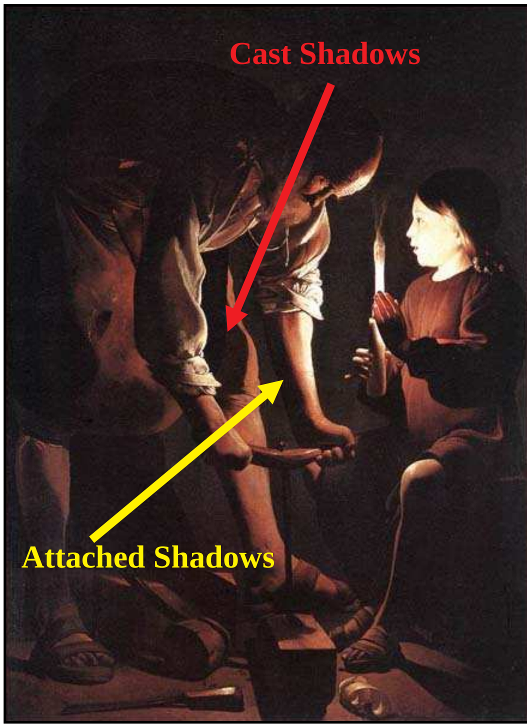
- Shadows give us important visual cues about 3D object placement and motion
- Movies are from:
- http://vision.psych.umn.edu /users/kersten/kersten-lab/demos/shadows.html

Slide 2

Why Shadows ?

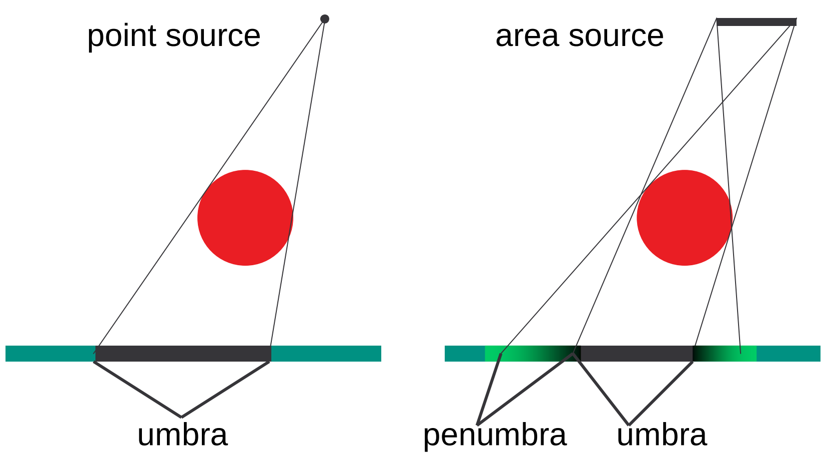
Also, realism …

Image courtesy, Codemasters
Game: Blade of Darkness



- Attached shadows are easy to render
    - Because they are local.
    - We already have discussed this.
- Cast shadows require us to determine whether the surface is visible to the light.

Hard and Soft Shadows



Images courtesy, Eric Haines and Tomas Moeller

Real-time Shadows

- Assumptions: hard shadows from point light sources onto planar surfaces
- Let the light source be at infinity in the direction **L**(x_l, y_l, z_l), and we want to compute the shadow S(x_w, y_w, z_w) of the point **P**(x_p, y_p, z_p), on the plane $z = 0$
- It is easy to see that **S** lies on the line defined by **P** and **L** or

  $\mathbf{S} = \mathbf{P} - \alpha\, \mathbf{L}$
- Since the shadow is on the plane $z = 0$,

  $z_w = 0 \Rightarrow \alpha = z_p / z_l$, and

  $x_w = x_p - (z_p / z_l)\, x_l$

  $y_w = y_p - (z_p / z_l)\, y_l$

Real-time Shadows

Restating the equations from previous slide in a matrix form:

$$\begin{bmatrix} x_w \\ y_w \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -x_l/z_l & 0 \\ 0 & 1 & -y_l/z_l & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

Shadows on arbitrary planes can be performed by prefixing the above matrix by a suitable transformation that transforms that plane to $z = 0$

Real-time Shadows

- Remember to set the right shadow color before drawing the shadow
- Z-conflicts can ruin shadows, so remember to slightly offset the shadows to lie above the surface:

```
glEnable(GL_POLYGON_OFFSET_FILL)
glPolygonOffset(GLfloat factor, GLfloat units)
// display the polygon here
glDisable(GL_POLYGON_OFFSET_FILL)
```

- Alternatively if the scene geometry is well understood the following might be possible and simpler:

  Render the plane, turn off the depth test, render the shadows, turn the depth test back on, and render the rest of the scene.

Real-time Shadows

We assumed light source at infinity.

If the light source is local at **L**(x_l, y_l, z_l), we can use the following matrix [Blinn 88]:

$$\begin{bmatrix} x_w \\ y_w \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} z_l & 0 & -x_l & 0 \\ 0 & z_l & -y_l & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & z_l \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

Why is this the right matrix? Let's just work through the x coordinate. To project the point onto the z = 0 plane as a shadow, we need to move it in the direction from the light to the point, and to move it far enough so that it's on the z = 0 plane. That means we need to add a scaled version of (xp-xl,yp-yl,zp-zl) to the point (xp,yp,zp). Our projected point will be (xp,yp,zp) + a(xp-xl,yp-yl,zp-zl) where we choose a so that the resulting z coordinate will be 0. That is, zp + a(zp-zl) = 0, or a = zp(zl-zp).

The x coordinate that we wind up with, then will be xp + zp(xp-xl)/(zl-zp). We can rewrite this as:

(xp(zl-zp) + zp(xp-xl))/(zl-zp) = (xp*zl – zp*xl)/(zl-zp).

Notice that if we apply our matrix to (xp,yp,zp,1), we get (xp*zl – zp*xl, ?, 0, zl-zp), where we haven't calculated the y coordinate yet. This is in homogenous coordinates, but if we divide by the fourth coordinate, we get the correct x coordinate that we have just calculated. We can check the y coordinate in the same way.

Light Maps

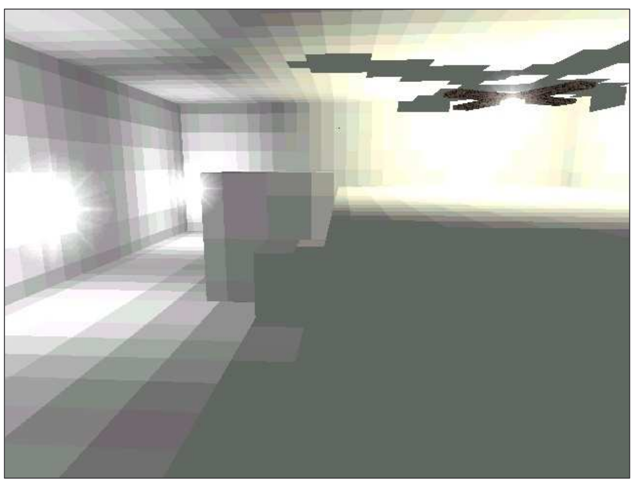
- Idea is to store the view-independent lighting of a scene as a 2D texture map
- Light maps are reasonably effective even when used at low resolutions (since they usually don't have high frequency detail)
- Efficiency involves clustering similarly lighted polygonal patches (Zhukov *et al.* 1998)

Texture-Mapped Scene

Images courtesy, 3D Games by Watt and Policarpo

Light Mapped Scene

No filtering of the light map



Images courtesy, 3D Games by Watt and Policarpo

Light-Mapped Scene

Light map with linear filtering

Images courtesy, 3D Games by Watt and Policarpo

Texture & Light Mapped Scene

Texture mapped * Filtered Light Mapped

Images courtesy, 3D Games by Watt and Policarpo

Light Maps


Texture mapped


Texture + Filtered Light Mapped

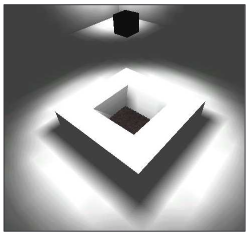
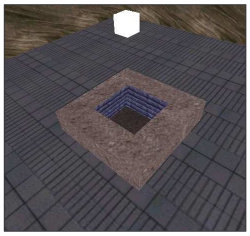
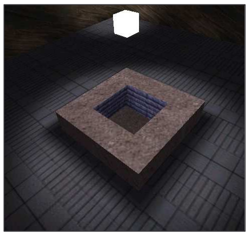
Images courtesy, 3D Games by Watt and Policarpo

Shadow Augmented Light Maps

- If light sources and scene objects are static then the shadows will be static.
- Precompute the shadows as a part of the light map and apply as a texture

"(The world) saw shadows black until Monet discovered they were coloured,…"

Maugham, Of Human Bondage

Images from 3D Games by Watt and Policarpo

Shadow Z-Buffer

- Proposed by Williams 1978
- Render the scene from the light's point of view and store the result in a shadow z-buffer
- Then render the scene from the user's view point and for each pixel that overwrites a previously written pixel:
  - Transform the pixel's screen space coordinates into the light source's coordinate frame
  - Index into the shadow z-buffer to see whether the rendered point's depth is greater than the depth for the corresponding pixel in the shadow z-buffer
  - If the depth is greater ⇒ point is in shadow and use the shadow color, otherwise render normally

Shadow Z-buffer

Low resolution shadow map

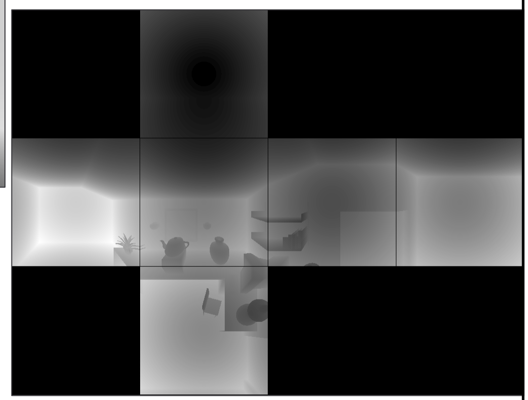
Higher resolution shadow map

Images from 3D Computer Graphics by Watt

Shadow Z-buffer

Depth map from User's View Point

Images from 3D Computer Graphics by Watt



Shadow Environment Z-buffer from Light's View Point