Shadows

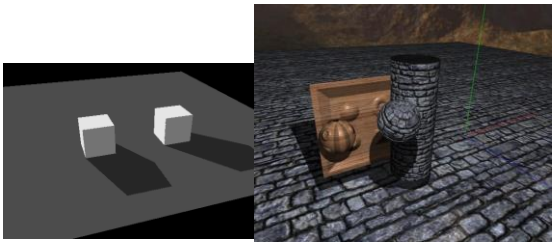
Computer Graphics

Shadows

- Simple geometry tricks
  - (assumptions, such as, light sources are points)
- Shadows don't have to be 100% accurate
- Planar Shadows
- Shadow Mapping

Why do we need shadows?

Why do we need shadows?



Why do we need shadows?

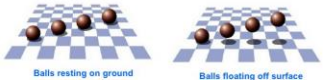
- Shadows:

... make a scene look more three-dimensional

... tell us where the light comes from

... emphasize the spatial relationship of objects among each other

... should really be there (a real-world scene has shadows)

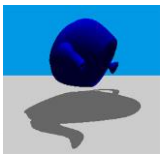


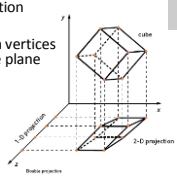
How to do shadows?

- A number of techniques, such as,
  - Planar Shadows
  - Shadow Mapping
  - Shadow Volumes
- Focus on Shadow Mapping
  - two pass algorithm
  - fast on GPUs
  - straightforward to understand and implement

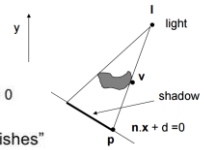
Planar Shadows

- The simplest algorithm – shadowing occurs shadowing occurs when objects cast shadows on planar surfaces (projection shadows)
- Literally projects the mesh vertices onto a 3D plane (using the plane equation)





Planar Shadows



Given

- Ground plane equation, Ax + By + Cz + D = 0
- Light position (x, y, z, w)

Construct projective transform that "squishes" vertices into ground plane based on light position

- Concatenate this with view transform

Rendering 3D object creates shadow-like pile of polygons in ground plane

Planar Shadows

- 4x4 Projection Matrix

$$\begin{bmatrix} P - Lx\,A & -Ly\,A & -Lz\,A & -Lw\,A \\ -Lx\,B & P - Ly\,B & -Lz\,B & -Lw\,B \\ -Lx\,C & -Ly\,C & P - Lz\,C & -Lw\,C \\ -Lx\,D & -Ly\,D & -Lz\,D & P - Lw\,D \end{bmatrix}$$

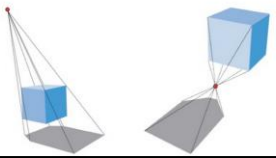
where P = Lx A + Ly B + Lz C + Lw D

Planar Shadows

- Careful about reverse projection

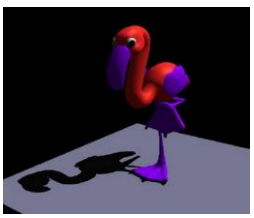
Projection can cast a shadow of an object "behind" the light *w.r.t.* the plane

- Make sure occluders are between light and ground plane



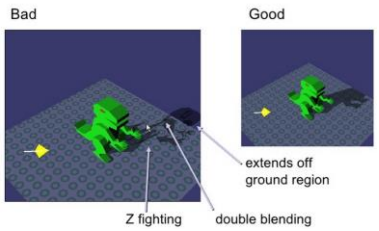
Planar Shadows

- 'Fake' shadows
- Uses a projective geometry trick
- Flattens 3d model onto a plane
- Shadow effect can only be applied to planes
- Supports either positional or directional lights



Planar Shadow Artifacts

- Stacked polygons causes z-fighting artifacts
  - offset polygon to fix
- Infinite plane
  - clip edges



Shadow Maps

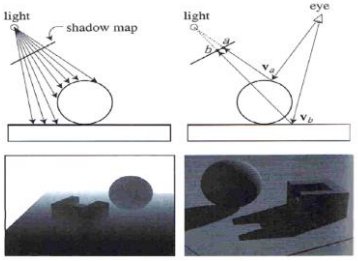
- Leverage GPU hardware
  - Depth buffering + texture mapping
  - Multi-pass algorithm: render depth maps, then project depth maps as textures for the eye view

Shadow Mapping

- Basic idea: objects that are not visible to the light are in shadow
- How to determine whether an object are visible to the eye?
  - Use z-buffer algorithm, but now the "eye" is the light, i.e., the scene is rendered from light's point of view
  - This particular z-buffer for the eye is called the shadow map

Shadow Mapping

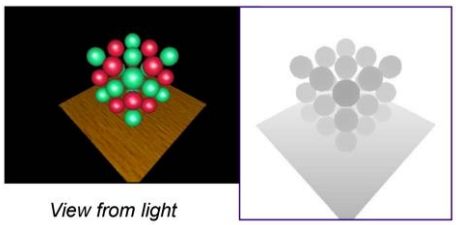
- Illustration



Shadow Mapping (Algorithm)

- 1. Render the scene using the light as the camera and perform z-buffering
- 2. Generate a light z-buffer (called a shadow map)
- 3. Render the scene using the regular camera, perform z-buffering, and run the following steps:
- [] For each 'fragment' in local space (x,y,z), perform a transformation to the light's clip space (i.e., as seen by the eye) to (x1,y1,z1)
- [] Compare the z1 with the shadow-map z[x1,y1] If z1<=z (closer to the light) - then this pixel is not in shadow; otherwise, the pixel is shadowed

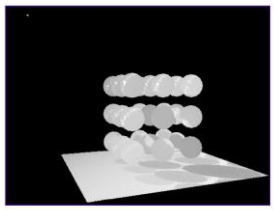
Shadow Mapping (1st Pass)



View from light

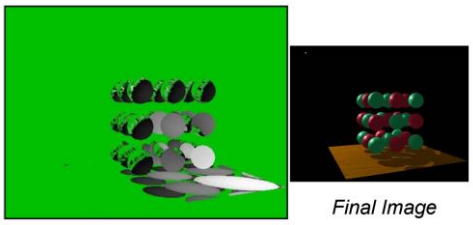
Depth Buffer (shadow map)

Shadow Mapping (2nd Pass)



Visible surface depth

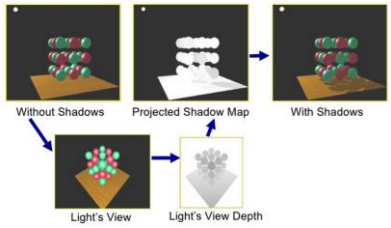
Shadow Mapping (Final)



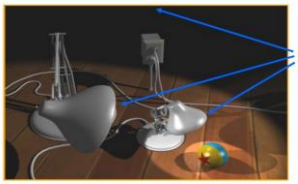
Non-green in shadow

Final Image

Shadow Mapping

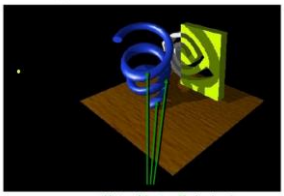


Shadow Map Example



3 light sources =
3 depth map
renders per frame

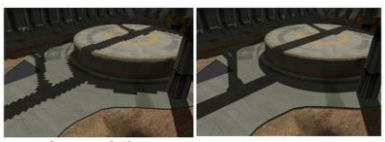
Shadow Map Example



Note object self-shadowing

Shadow Map Artifacts

- Resolution of the rendered depth map determines shadow mapping precision



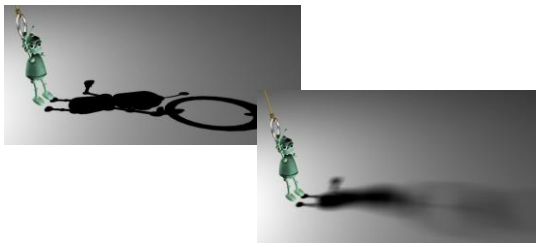
low-resolution
shadow map blockiness

sufficient
shadow map resolution

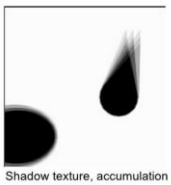
Shadow Mapping (Quality)

- Shadow quality depends on
  - Shadow map resolution – aliasing problem
  - Z resolution – the shadow map is often stored in one channel of texture, used to be 8 bits but now most of hardware supports 24 bits
  - Self-shadow aliasing – caused by different sample positions in the shadow map and the screen

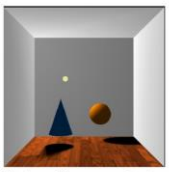
Hard vs Soft



Soft Shadows



Shadow texture, accumulation of nine jittered occluder projections



Final resulting scene, shadow texture modulated with wooden floor

Summary

- Real-Time Rendering, CRC Press, 2008
- 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