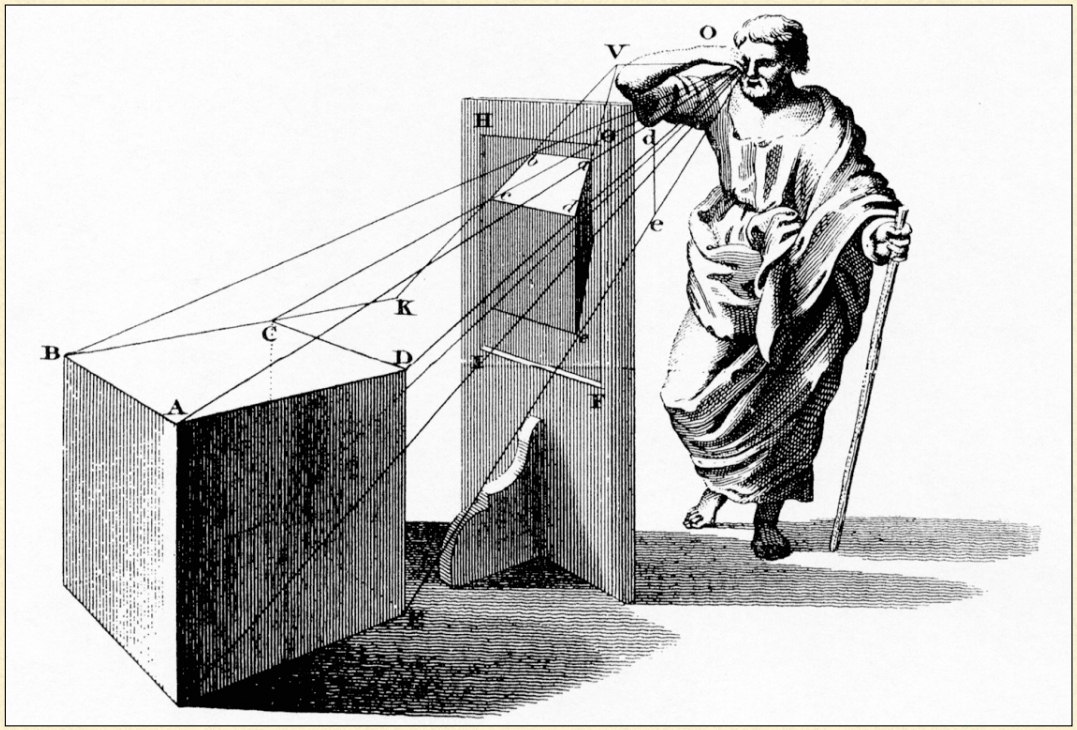
Graphics Overview

CS 294-137: Theory and Applications of Virtual Reality and Immersive Computing

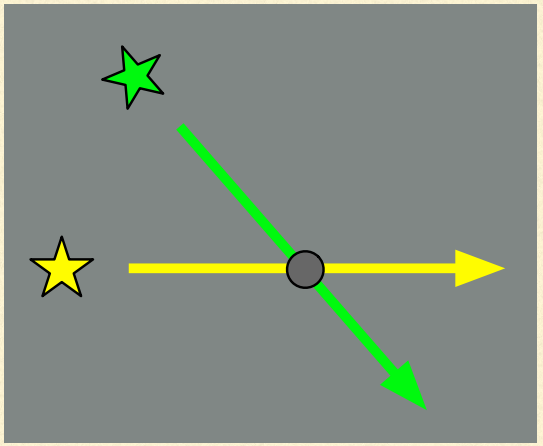
Fooling your Eyes



- Single point linear perspective projection

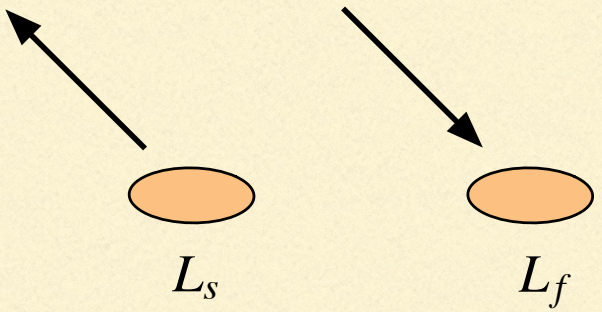
RADIANCE

- Light energy passing though a point in space within a given solid angle
  - Energy per square meter per steradian ($\mathrm{W/m^2/sr}$)
  - S.E.D. per square meter per steradian ($\mathrm{W/m^2/sr/nm}$)
- Constant along straight lines in free space

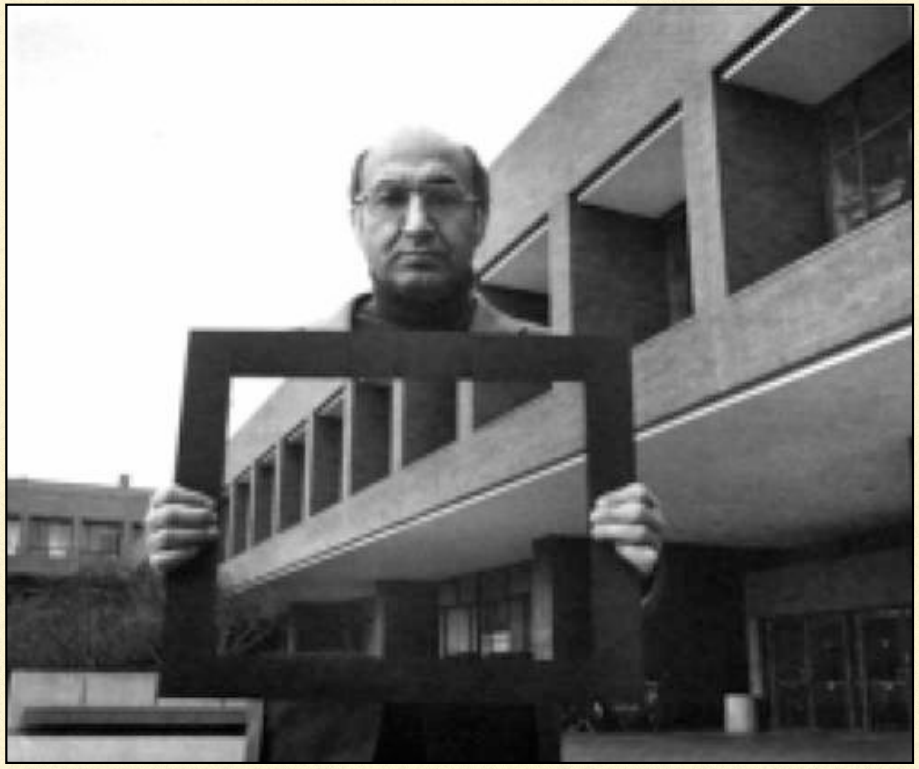


RADIANCE

- Near surfaces, differentiate between
  - Radiance from the surface (surface radiance)
  - Radiance from other things (field radiance)

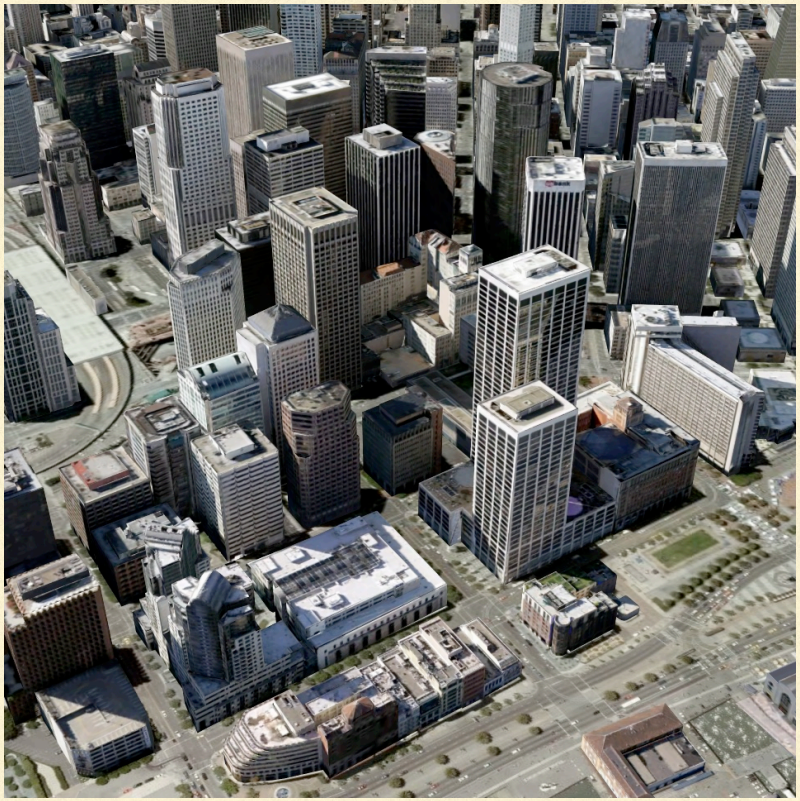


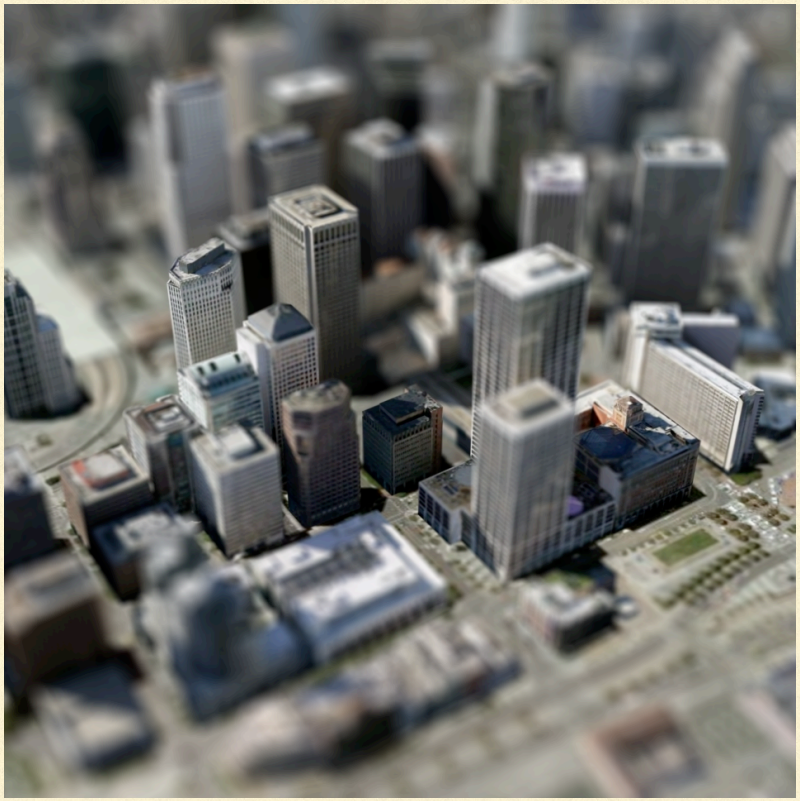
Almost Fooling your Eyes



- Consider:
  - moving viewpoint
  - changing focus

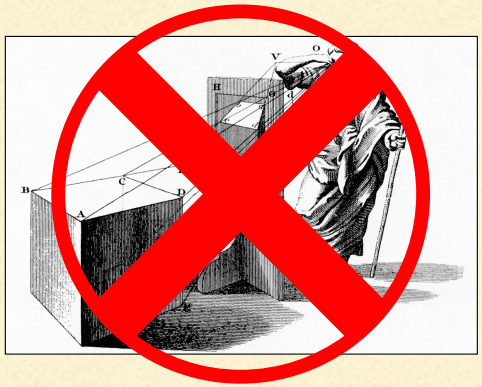
Almost Fooling your Eyes

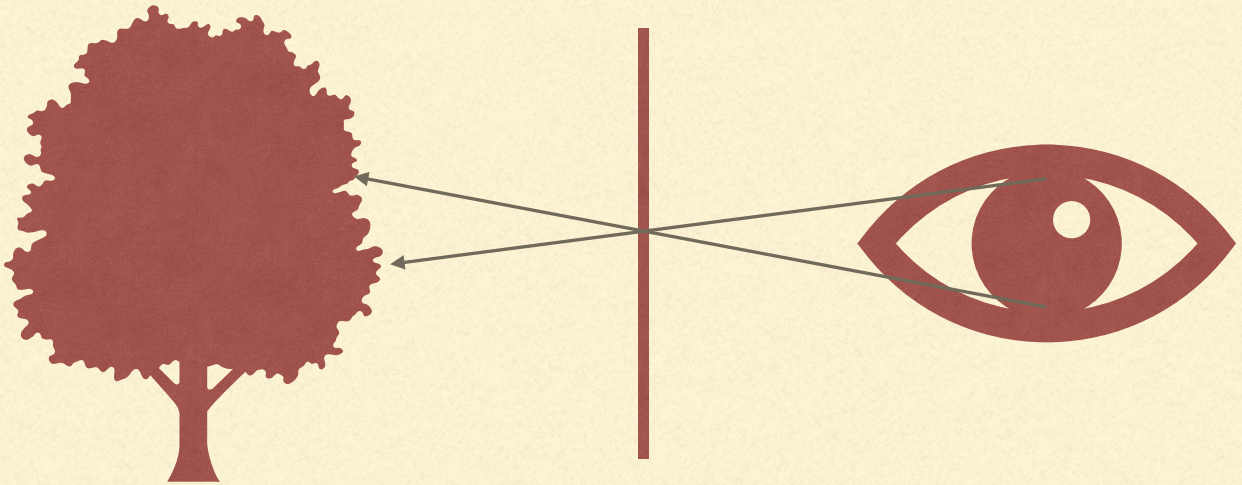




Held, Cooper, O'Brien, Banks, 2010

Almost Fooling your Eyes

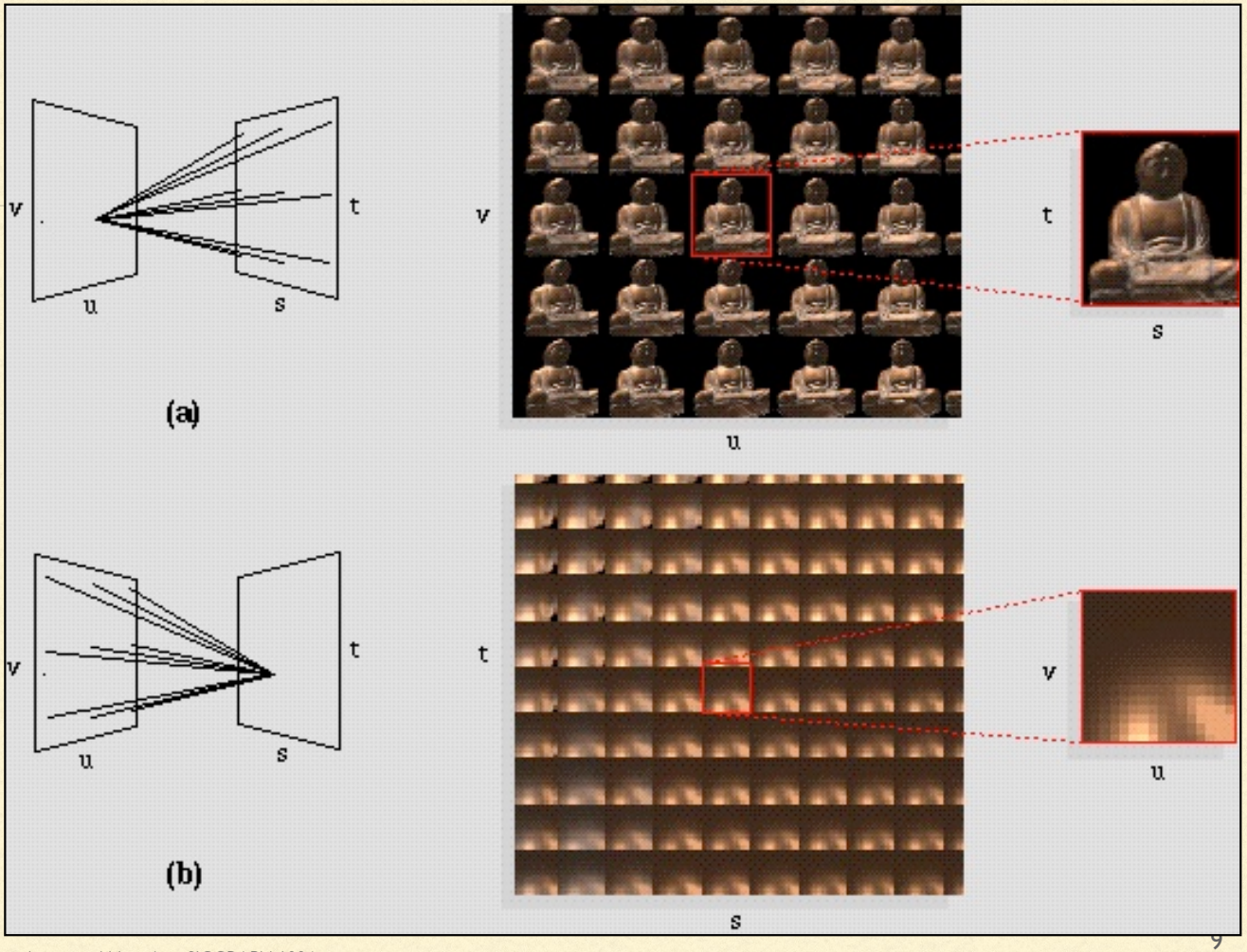




Light Fields

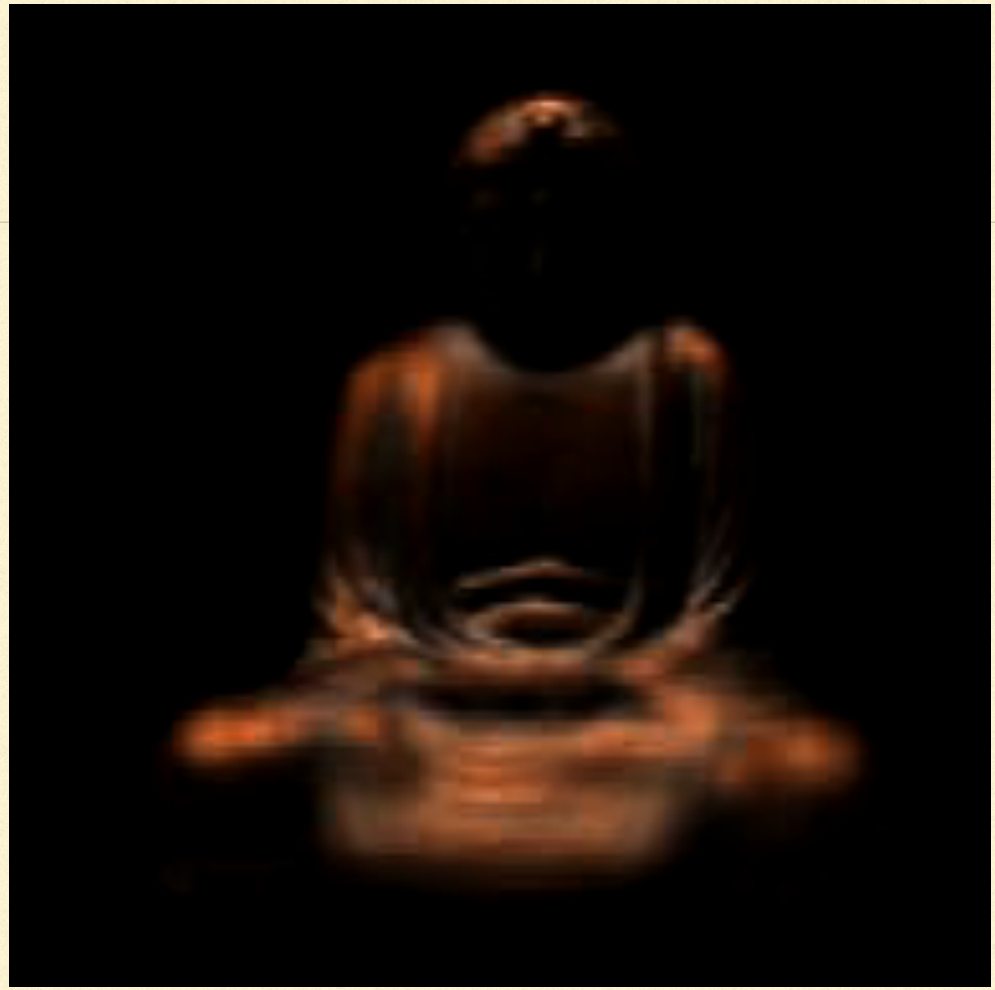
- Radiance at every point in space, direction, and frequency: 6D function
  - Sampled over volume, direction, and wavelength
- Collapse frequency to RGB, and assume free space: 4D function
  - RGB samples over surface and direction

Light Fields



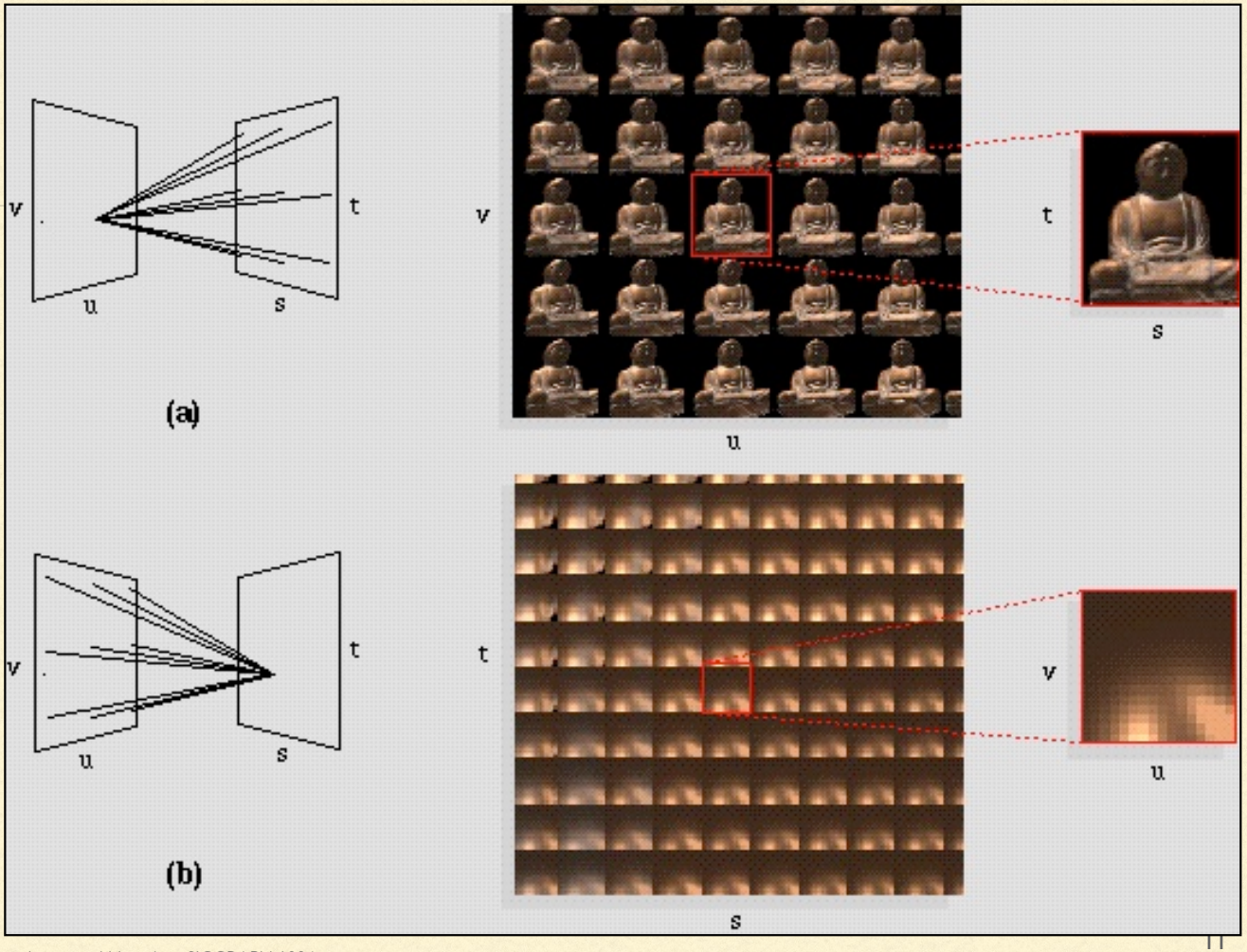
Levoy and Hanrahan, SIGGRAPH 1996

Light Fields



Levoy and Hanrahan, SIGGRAPH 1996

Light Fields



Levoy and Hanrahan, SIGGRAPH 1996

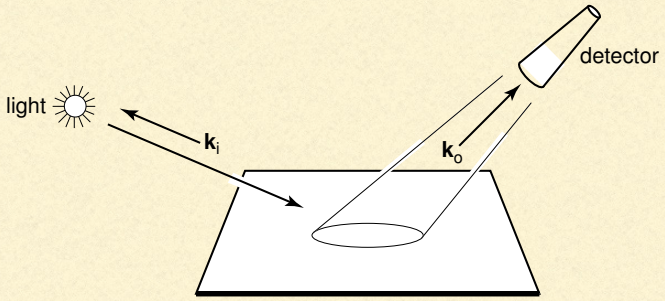
Light Fields

Michelangelo's ***Statue of Night***
From the Digital Michelangelo Project

The BRDF

$$\rho(\mathbf{k}_i, \mathbf{k}_o) = \frac{L_s(\mathbf{k}_o)}{L_f(\mathbf{k}_i)\cos(\theta_i)}$$

- Bidirectional Reflectance Distribution Function
  - How much light from direction k_i goes out in direction k_o
  - Describes the appearance of a particular material
  - Varies spatially (*i.e.* texture)
  - Note: For perfect Lambertian reflector with constant BRDF has $\rho = 1/\pi$



The Rendering Equation

- Total light going out in some direction is given by an integral over all incoming directions:

$$L_s(\mathbf{k}_o) = \int_\Omega \rho(\mathbf{k}_i, \mathbf{k}_o) L_f(\mathbf{k}_i) \cos(\theta_i) \mathrm{d}\sigma_i$$

  - Note, this is recursive (one point's L_f is another's L_s)
  - Consider ray tracing

The Rendering Equation

$$L_s(\mathbf{k}_o) = \int_\Omega \rho(\mathbf{k}_i, \mathbf{k}_o) L_f(\mathbf{k}_i) \cos(\theta_i) \mathrm{d}\sigma_i$$

$$L_f(\mathbf{k}_i) = L_s(-\mathbf{k}_i)$$

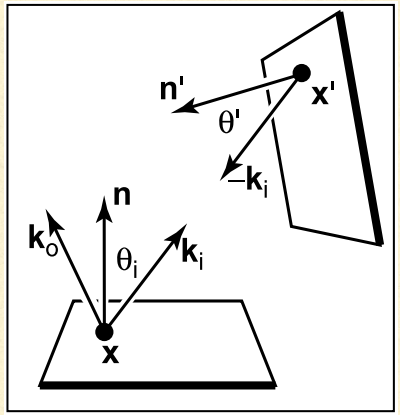
Rewrite explicitly in terms of surface radiances only

$$\Delta\sigma_i = \frac{\Delta A' \cos(\theta')}{||\mathbf{x} - \mathbf{x}'||^2}$$

$$L_s(\mathbf{x}, \mathbf{k}_o) = \int_{x' \text{ visible to } x} \frac{\rho(\mathbf{k}_i, \mathbf{k}_o) L_s(\mathbf{x}', \mathbf{x} - \mathbf{x}') \cos(\theta_i) \cos(\theta')}{||\mathbf{x} - \mathbf{x}'||^2} \mathrm{d}\mathbf{A}'$$

$$L_s(\mathbf{x}, \mathbf{k}_o) = \int_{\text{all } x'} \frac{\rho(\mathbf{k}_i, \mathbf{k}_o) L_s(\mathbf{x}', \mathbf{x} - \mathbf{x}') \delta(\mathbf{x}, \mathbf{x}') \cos(\theta_i) \cos(\theta')}{||\mathbf{x} - \mathbf{x}'||^2} \mathrm{d}\mathbf{A}'$$

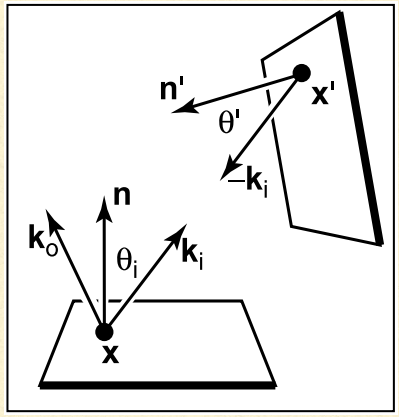
$$\delta(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 \text{ if } \mathbf{x} \text{ and } \mathbf{x}' \text{ are mutually visible} \\ 0 \text{ otherwise} \end{cases}$$



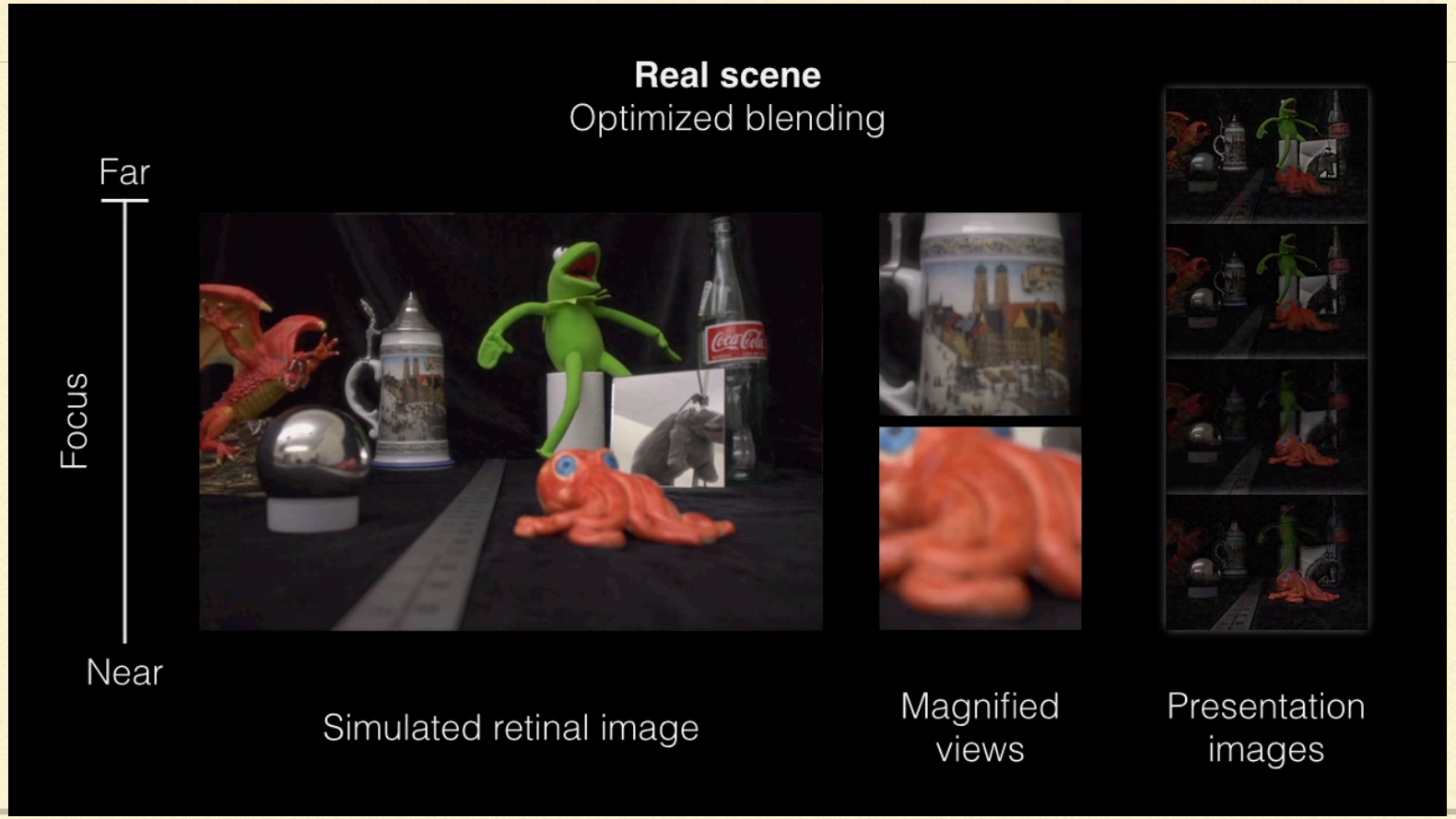
The Rendering Equation

$$L_s(\mathbf{x}, \mathbf{k}_o) = \int_{\text{all } x'} \frac{\rho(\mathbf{k}_i, \mathbf{k}_o) L_s(\mathbf{x}', \mathbf{x} - \mathbf{x}') \delta(\mathbf{x}, \mathbf{x}') \cos(\theta_i) \cos(\theta')}{||\mathbf{x} - \mathbf{x}'||^2} \mathrm{d}\mathbf{A}'$$

- Consider rendering a conventional image
- Consider rendering a light field image
- Consider brute force versus something clever



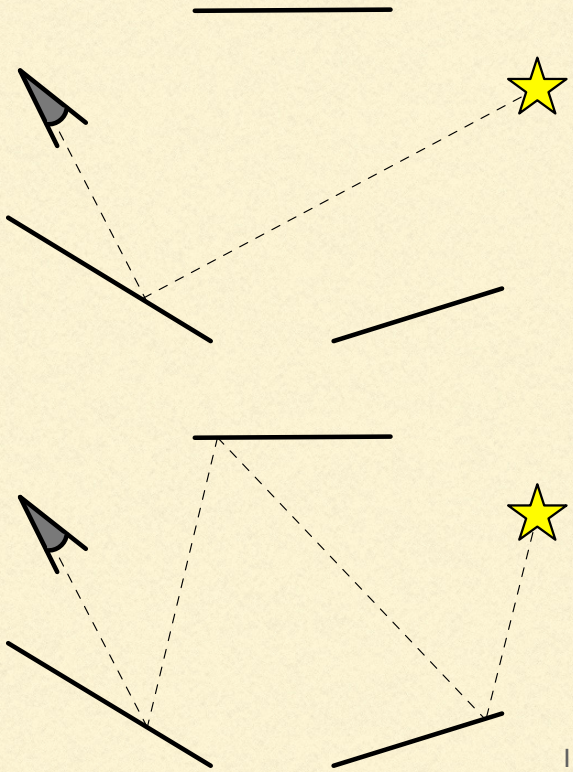
Optimized Display



Narain, Albert, Bulbul, Ward, Banks, O'Brien, 2015

Light Paths

- Many paths from light to eye
- Characterize by the types of bounces
  - Begin at light
  - End at eye
  - "Specular" bounces
  - "Diffuse" bounces

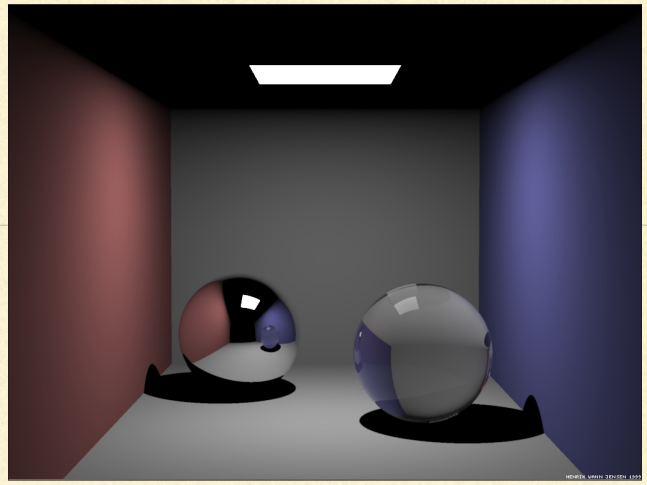


Light Paths

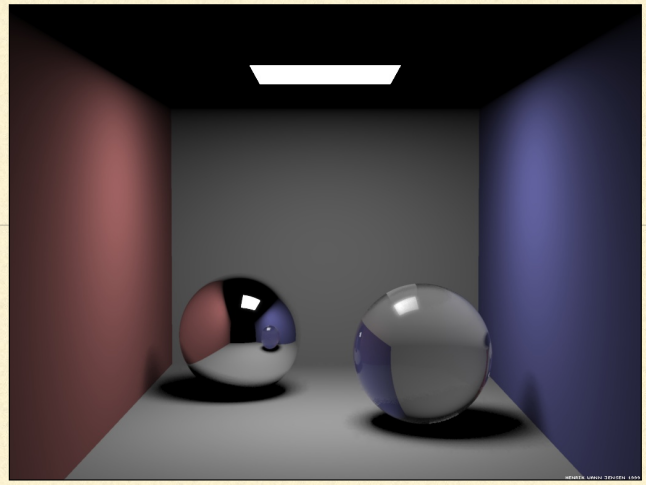
- Describe paths using strings
  - LDE, LDSE, LSE, etc.
- Describe types of paths with regular expressions
  - L{D|S}*E ← Visible paths
  - L{D|S}S*E ← Standard raytracing
  - L{D|S}E ← Local illumination
  - LD*E ← Radiosity method

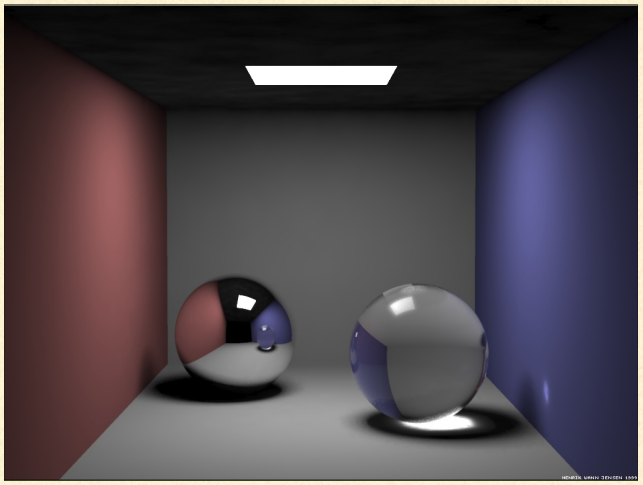
Light Paths

direct and simple specular paths

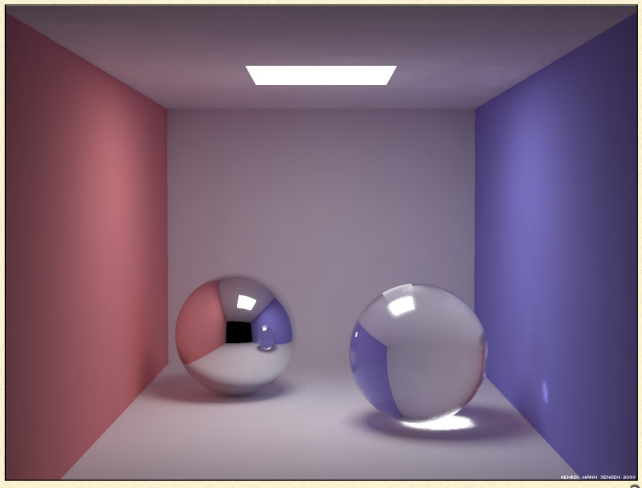


soft shadows





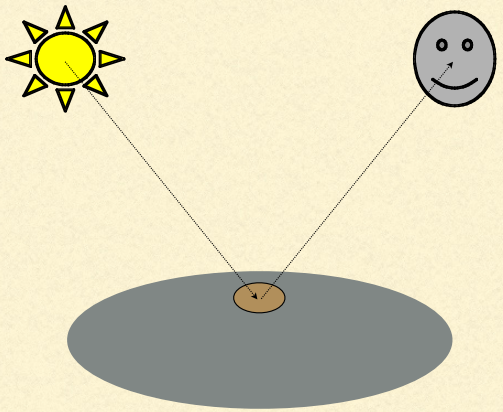
caustics



indirect illumination

Local Lighting

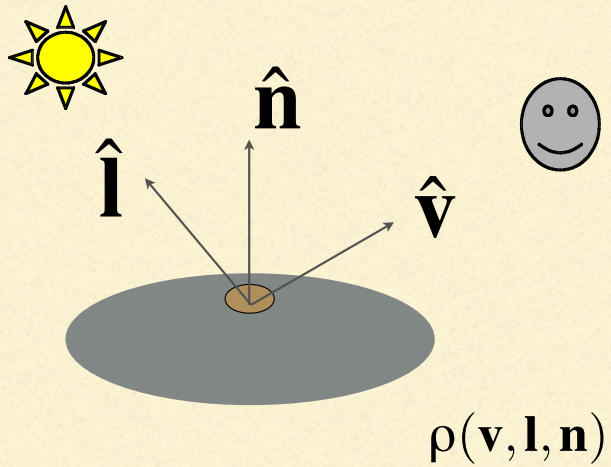
- Local: consider in isolation
  - 1 light
  - 1 surface
  - The viewer
- Recall: lighting is linear
  - Almost always...



Counter example: photochromatic materials

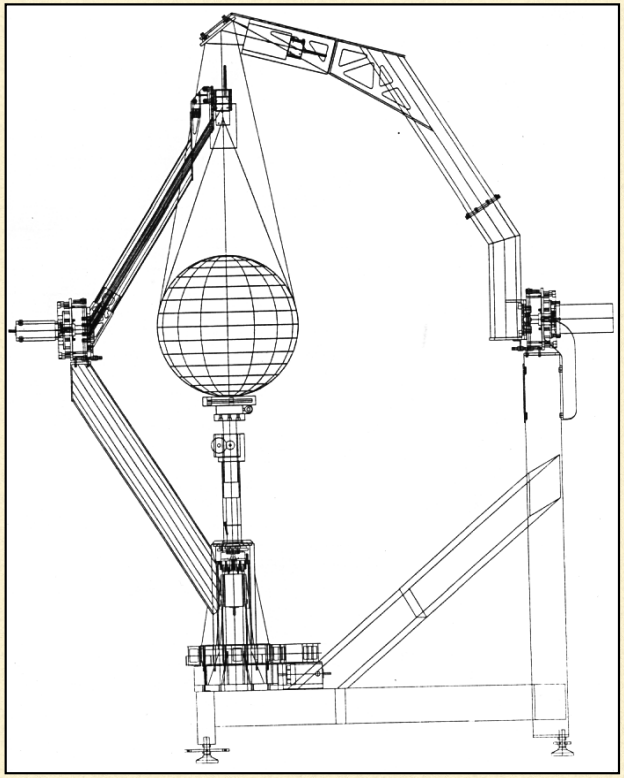
BRDF

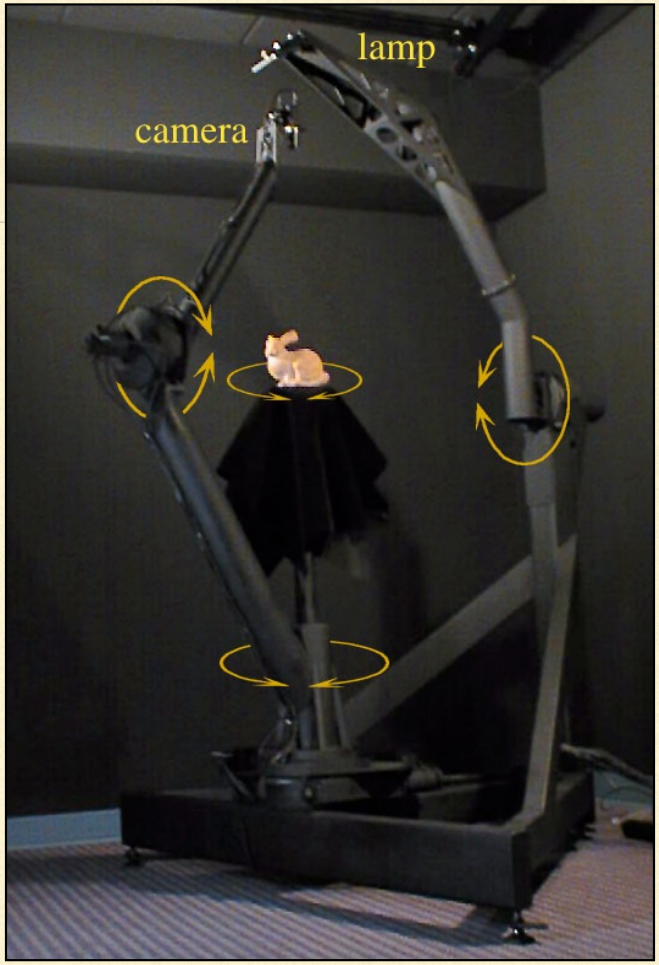
- Spatial variation capture by "the material"
- Frequency dependent
  - Typically use separate RGB functions
  - Does not work perfectly
  - Better: $\rho = \rho(\theta_V, \theta_L, \lambda_{in}, \lambda_{out})$



Obtaining BRDFs

- Measure from real materials





Images from Marc Levoy

The BRDF

Ideal specular

- Perfect mirror reflection

Ideal diffuse

- Equal reflection in all directions

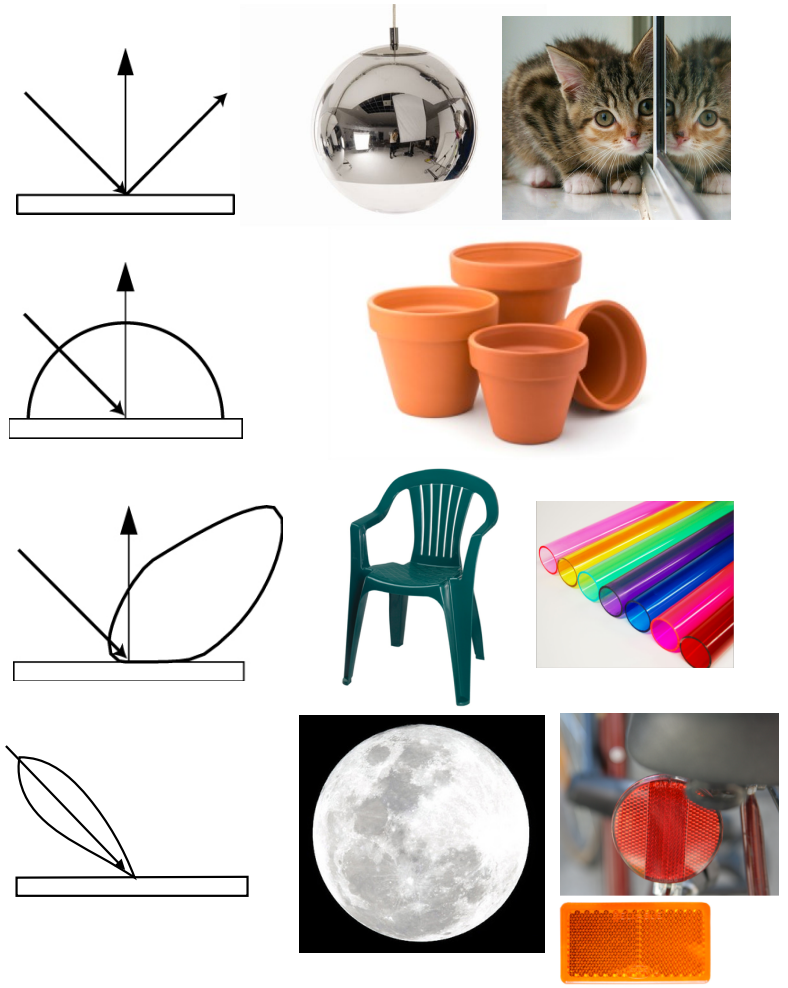
Glossy specular

- Majority of light reflected near mirror direction

Retro-reflective

- Light reflected back towards light source

Diagrams illustrate how light from incoming direction is reflected in various outgoing directions.



from Ren Ng

Beyond BRDFs

- The BRDF model does not capture everything
  - e.g. Subsurface scattering (BSSRDF)

Images from Jensen *et. al, SIGGRAPH 2001*

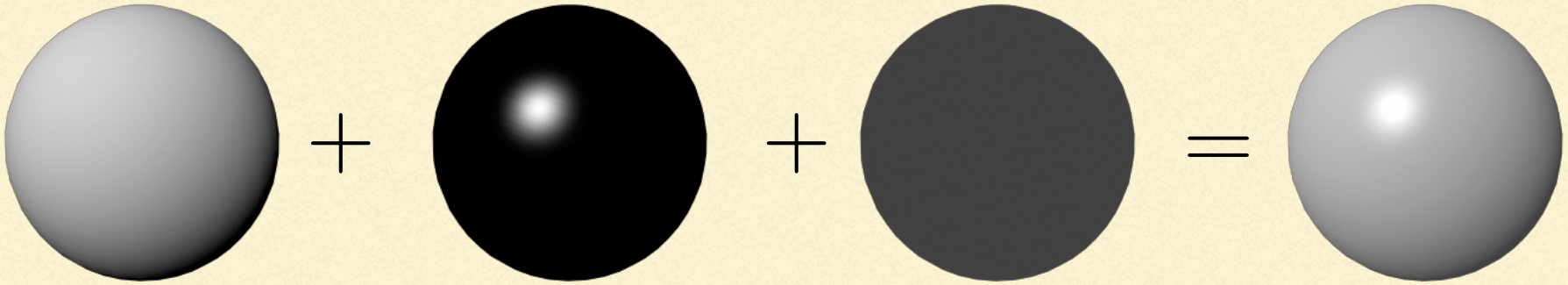
Beyond BRDFs

- The BRDF model does not capture everything
  - *e.g.* Inter-frequency interactions

This version would work: $\rho = \rho(\theta_V, \theta_L, \lambda_{\text{in}}, \lambda_{\text{out}})$

A Simple Model

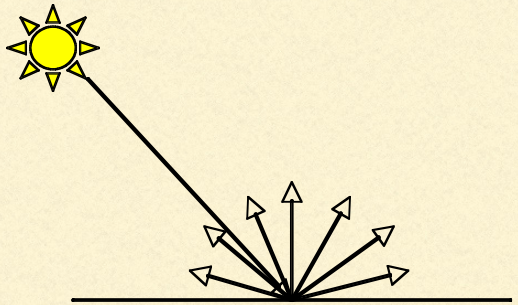
- Approximate BRDF as sum of
  - A diffuse component
  - A specular component
  - A "ambient" term



Diffuse Component

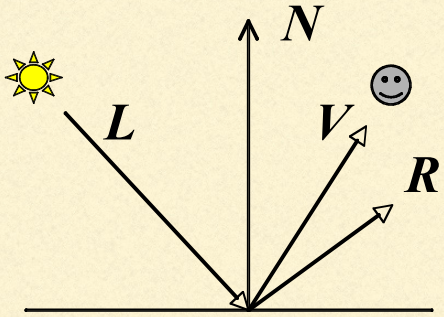
- Lambert's Law
  - Intensity of reflected light proportional to cosine of angle between surface and incoming light direction
  - Applies to "diffuse," "Lambertian," or "matte" surfaces
  - Independent of viewing angle
- Use as a component of non-Lambertian surfaces

$$\max(k_d I(\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}), 0)$$

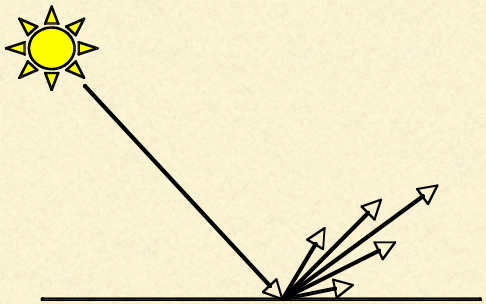


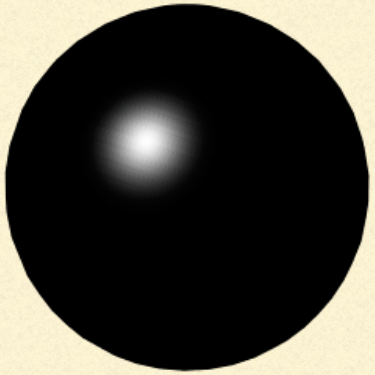
Specular Component

- Specular component is a mirror-like reflection
- Phong Illumination Model
  - A reasonable approximation for some surfaces
  - Fairly cheap to compute
- Depends on view direction



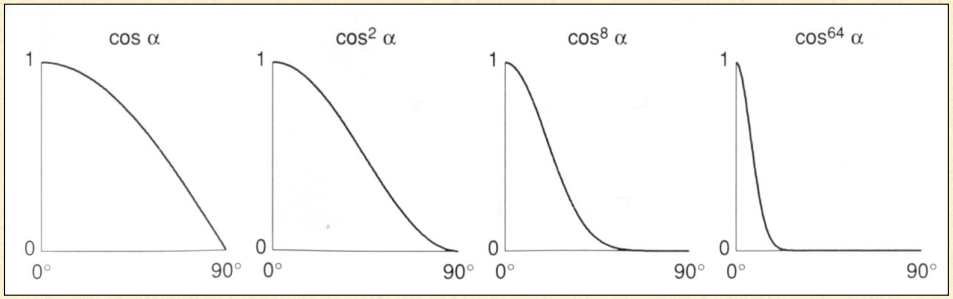
$$k_s I \max(\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}, 0)^p$$

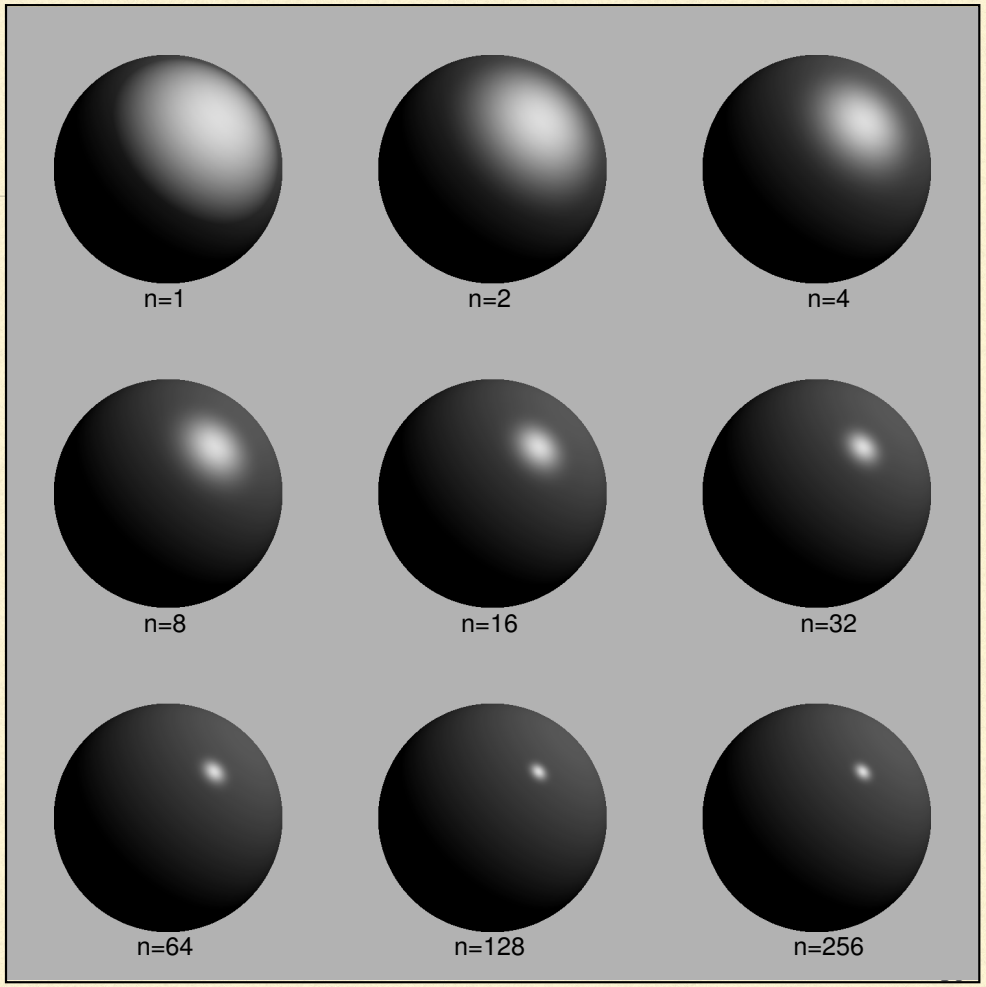




Specular Component

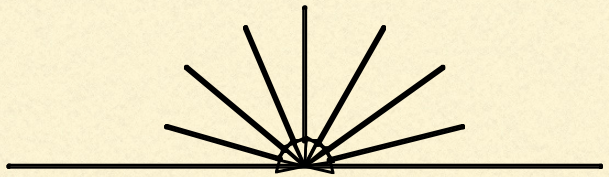
- Specular exponent sometimes called "roughness"

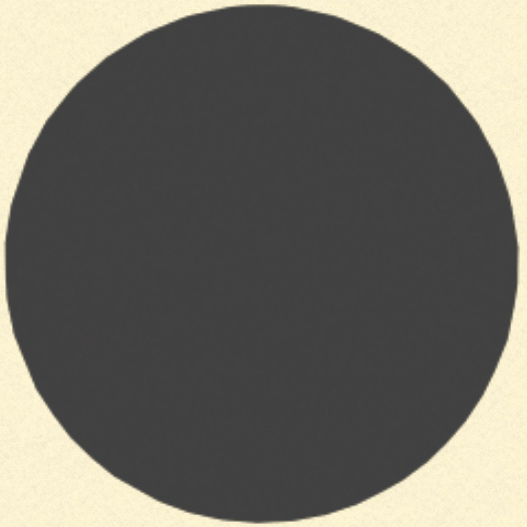




Ambient Component

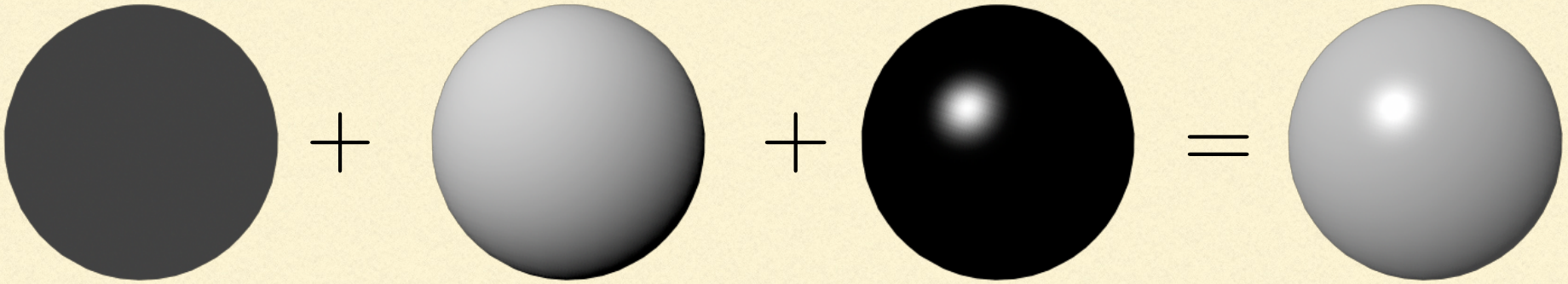
- Really, its a cheap hack
- Accounts for "ambient, omnidirectional light"
- Without it everything looks like it's in space





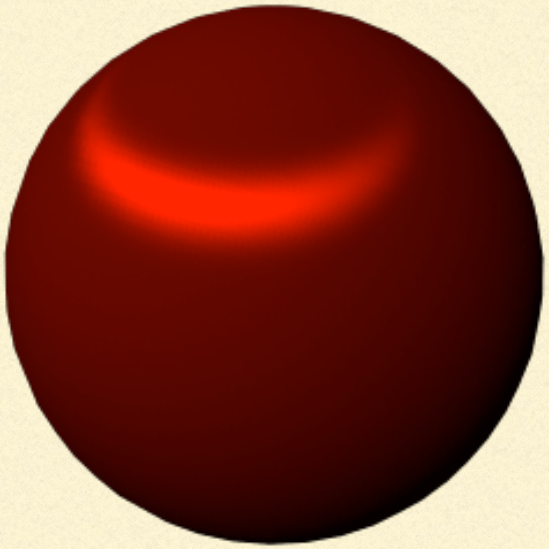
Summing the Parts

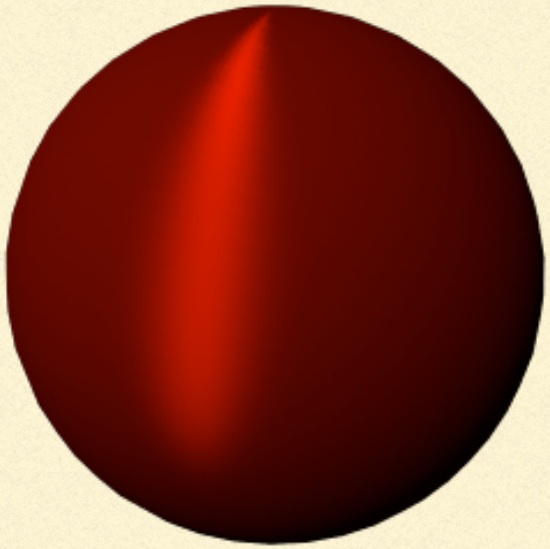
$$R = k_a I + k_d I \max(\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}, 0) + k_s I \max(\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}, 0)^p$$



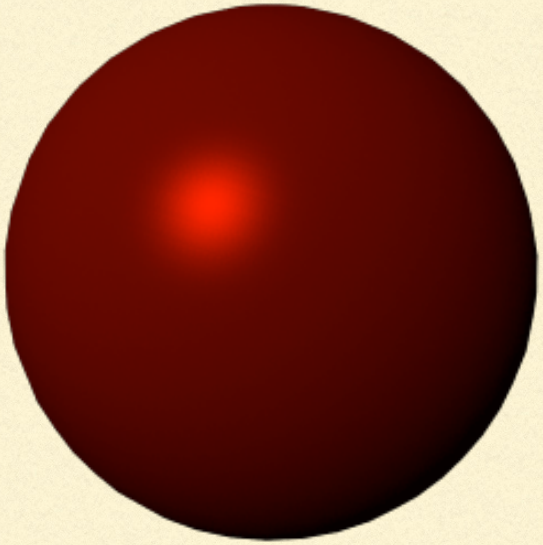
- Recall that the $k_?$ are by wavelength
  - RGB in practice
- Sum over all lights

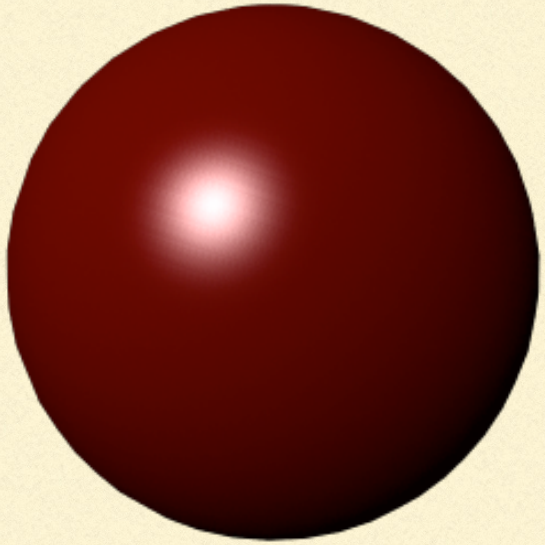
Anisotropy





"METAL" -VS- "PLASTIC"





Materials: Diffuse

Ren Ng

Materials: Plastic

Ren Ng

Materials: Paint

MATERIALS: PAINT

Ren Ng

Materials: Mirror

Ren Ng

Materials: Metallic

ASHIKHMIN-SHIRLEY BRDF

- More realistic specular term (for some materials)
- Anisotropic specularities
- Fresnel behavior (grazing angle highlights)
- Energy preserving diffuse term
- Sum of diffuse and specular terms (as before)

$$\rho(\hat{\mathbf{l}}, \hat{\mathbf{v}}) = \rho_d(\hat{\mathbf{l}}, \hat{\mathbf{v}}) + \rho_s(\hat{\mathbf{l}}, \hat{\mathbf{v}})$$

Michael Ashikhmin and Peter Shirley. 2000. An anisotropic phong BRDF model. J. Graph. Tools 5, 2 (February 2000), 25-32.
https://www.cs.utah.edu/~shirley/papers/jgtbrdf.pdf

Ashikhmin-Shirley BRDF

$$\rho_s(\hat{\mathbf{l}}, \hat{\mathbf{e}}) = \frac{\sqrt{(p_u+1)(p_v+1)}}{8\pi} \frac{(\hat{\mathbf{n}} \cdot \hat{\mathbf{h}})^{p_u \cos^2\phi + p_v \sin^2\phi}}{(\hat{\mathbf{h}} \cdot \hat{\mathbf{e}}) \max\left((\hat{\mathbf{n}} \cdot \hat{\mathbf{e}}), (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})\right)} F(\hat{\mathbf{h}} \cdot \hat{\mathbf{e}})$$

$$F(\hat{\mathbf{h}} \cdot \hat{\mathbf{e}}) = K_s + (1 - K_s)(1 - (\hat{\mathbf{h}} \cdot \hat{\mathbf{e}}))^5$$

Approximate Fresnel function

$\hat{\mathbf{l}}$	Light direction
$\hat{\mathbf{e}}$	Viewer (eye) direction
p_u, p_v	Specular powers
$\hat{\mathbf{n}}$	Normal
$\hat{\mathbf{h}}$	Half angle
K_s	Specular coefficient (color)
$\hat{\mathbf{u}}, \hat{\mathbf{v}}$	Parametric directions

ASHIKHMIN-SHIRLEY BRDF

$$\rho_s(\hat{\mathbf{l}}, \hat{\mathbf{e}}) = \frac{\sqrt{(p_u+1)(p_v+1)}}{8\pi} \frac{(\hat{\mathbf{n}} \cdot \hat{\mathbf{h}})^{\frac{p_u(\hat{\mathbf{h}}\cdot\hat{\mathbf{u}})^2 + p_u(\hat{\mathbf{h}}\cdot\hat{\mathbf{v}})^2}{1-(\hat{\mathbf{h}}\cdot\hat{\mathbf{n}})^2}}}{(\hat{\mathbf{h}} \cdot \hat{\mathbf{e}}) \max\Big((\hat{\mathbf{n}} \cdot \hat{\mathbf{e}}), (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})\Big)} F(\hat{\mathbf{h}} \cdot \hat{\mathbf{e}})$$

$$F(\hat{\mathbf{h}} \cdot \hat{\mathbf{e}}) = K_s + (1 - K_s)(1 - (\hat{\mathbf{h}} \cdot \hat{\mathbf{e}}))^5$$

Approximate Fresnel function

$\hat{\mathbf{l}}$	Light direction
$\hat{\mathbf{e}}$	Viewer (eye) direction
p_u, p_v	Specular powers
$\hat{\mathbf{n}}$	Normal
$\hat{\mathbf{h}}$	Half angle
K_s	Specular coefficient (color)
$\hat{\mathbf{u}}, \hat{\mathbf{v}}$	Parametric directions

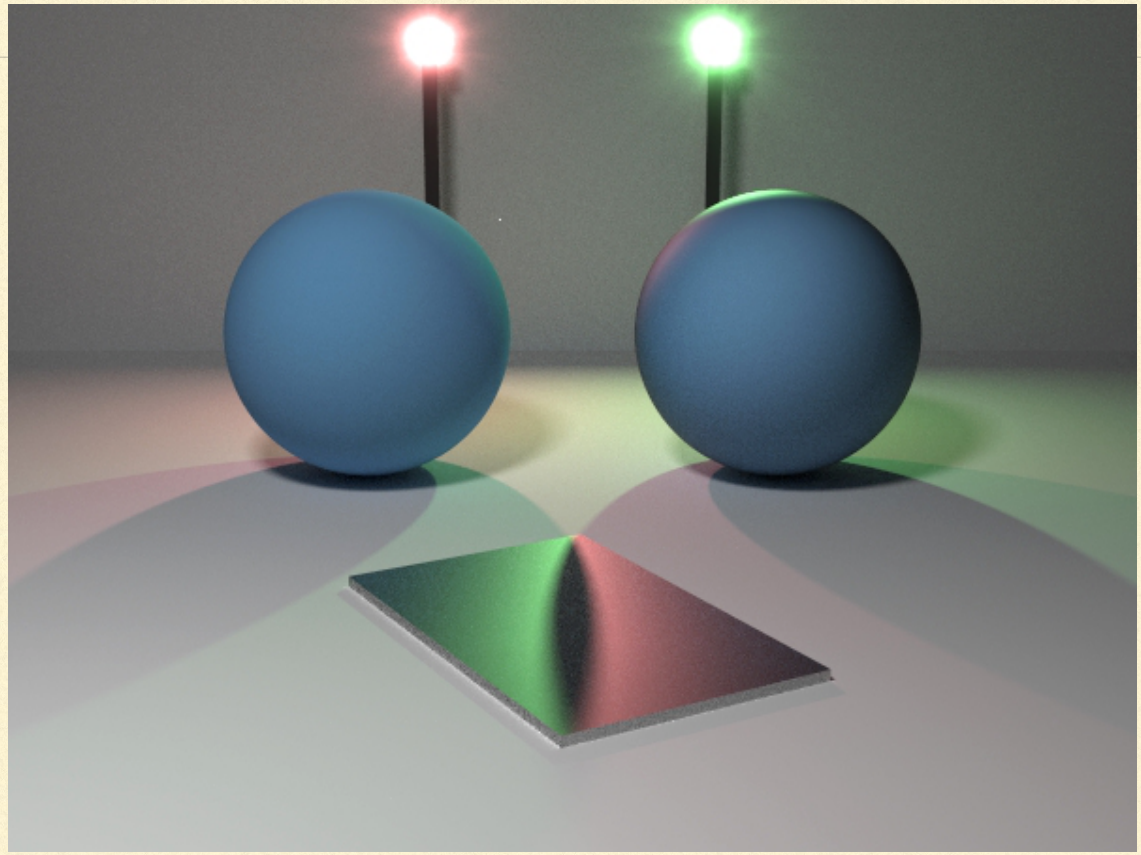
Ashikhmin-Shirley BRDF

$$\rho_d(\hat{\mathbf{l}}, \hat{\mathbf{e}}) = \frac{28K_d}{23\pi}(1 - K_s)\left(1 - \left(1 - \frac{\hat{\mathbf{n}} \cdot \hat{\mathbf{e}}}{2}\right)^5\right)\left(1 - \left(1 - \frac{\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}}{2}\right)^5\right)$$

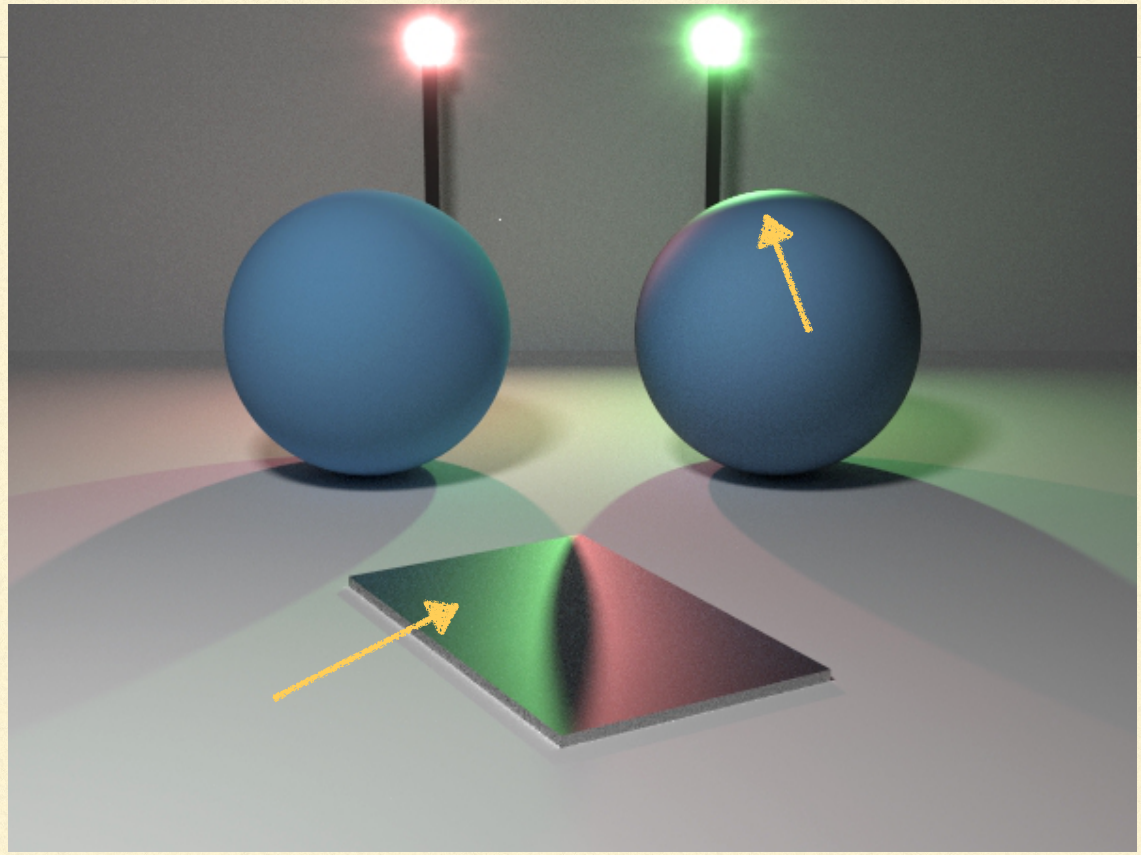
Note: The Phong diffuse term (Lambertian) is independent of view. But this term accounts for unavailable light due to specular/ Fresnel reflection.

$\hat{\mathbf{l}}$	Light direction
$\hat{\mathbf{e}}$	Viewer (eye) direction
p_u, p_v	Specular powers
$\hat{\mathbf{n}}$	Normal
$\hat{\mathbf{h}}$	Half angle
K_s	Specular coefficient (color)
$\hat{\mathbf{u}}, \hat{\mathbf{v}}$	Parametric directions

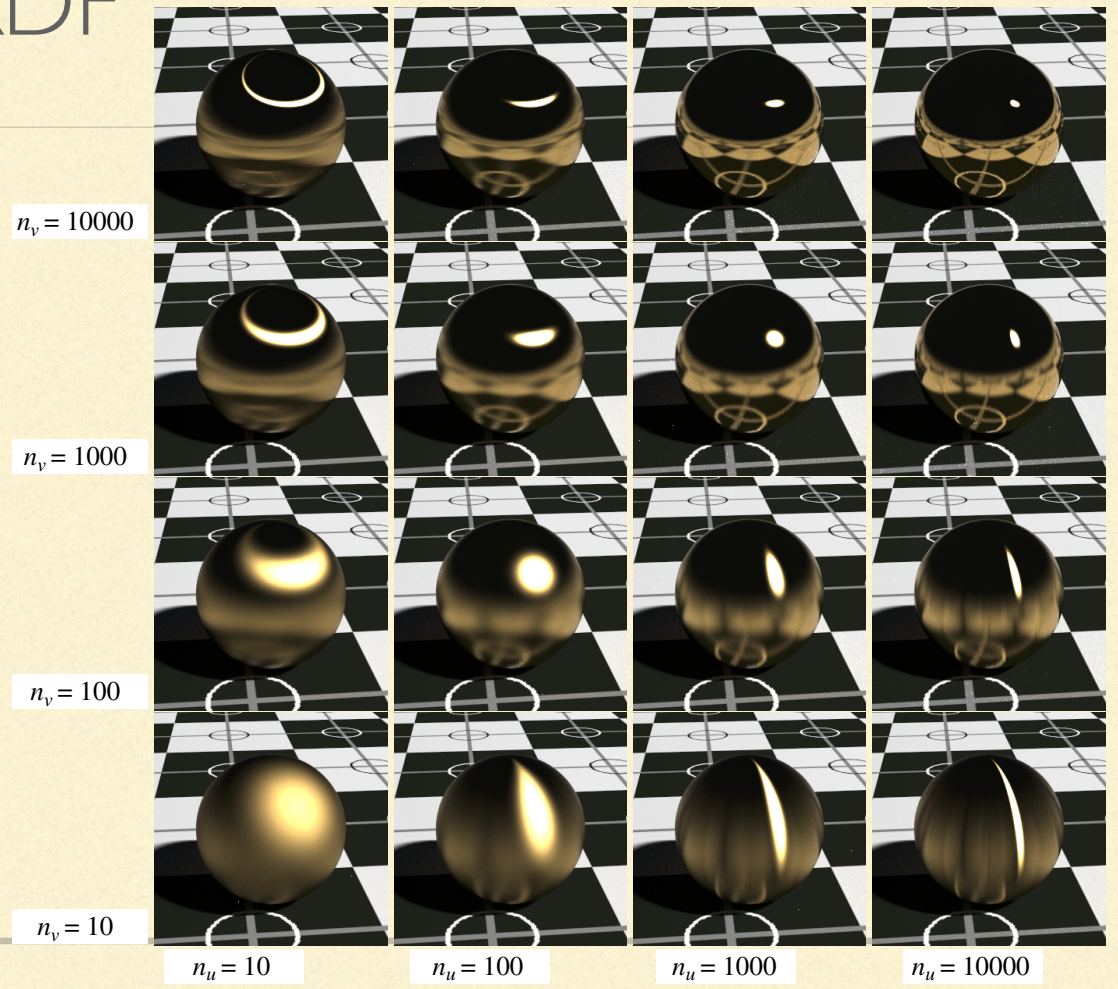
ASHIKHMIN-SHIRLEY BRDF



Ashikhmin-Shirley BRDF



Ashikhmin-Shirley BRDF



Details Beget Realism

- The "computer generated" look is often due to a lack of fine/subtle details... a lack of richness.

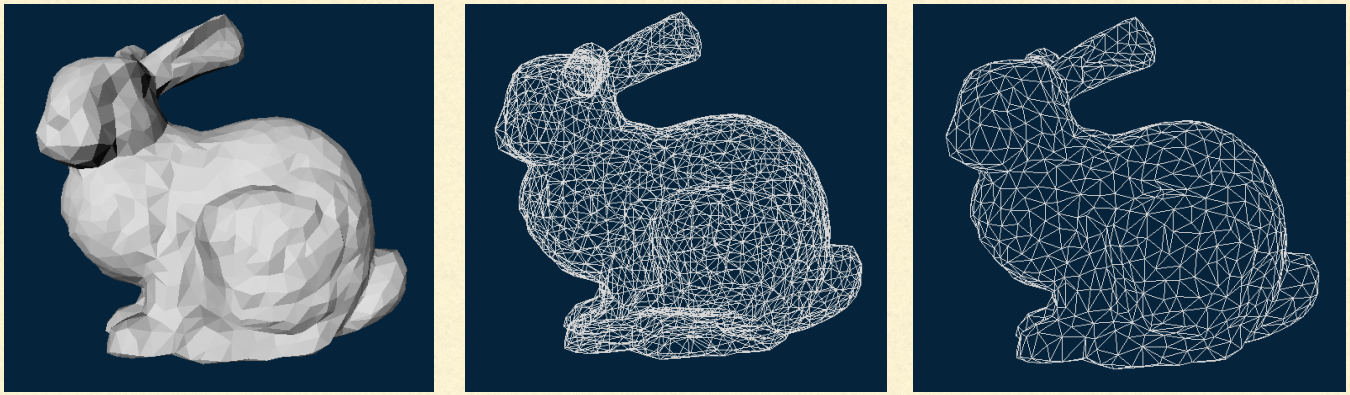
From bustledress.com

Details Beget Realism

- The "computer generated" look is often due to a lack of fine/subtle details... a lack of richness.

Hidden Surface Removal

- True 3D to 2D projection would put every thing overlapping into the view plane.
- We need to determine what's in front and display only that.



Z-Buffers

- Add extra depth channel to image
- Write Z values when writing pixels
- Test Z values before writing

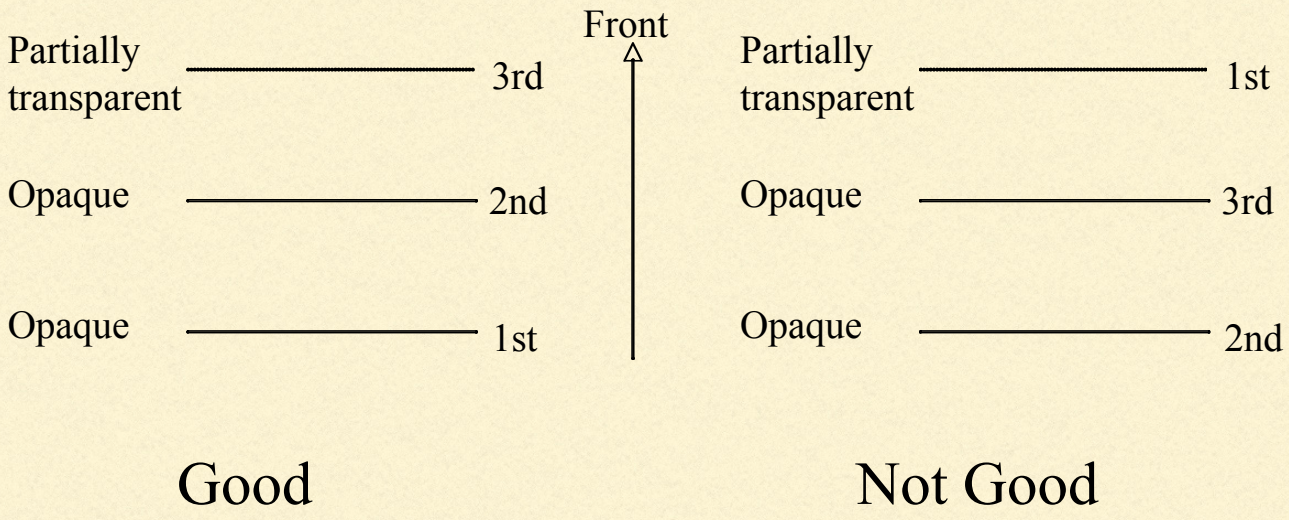
Images from Okan Arikan

Z-Buffers

- Benefits
  - Easy to implement
  - Works for most any geometric primitive
  - Parallel operation in hardware
- Limitations
  - Quantization and aliasing artifacts
  - Overfill
  - Transparency does not work well

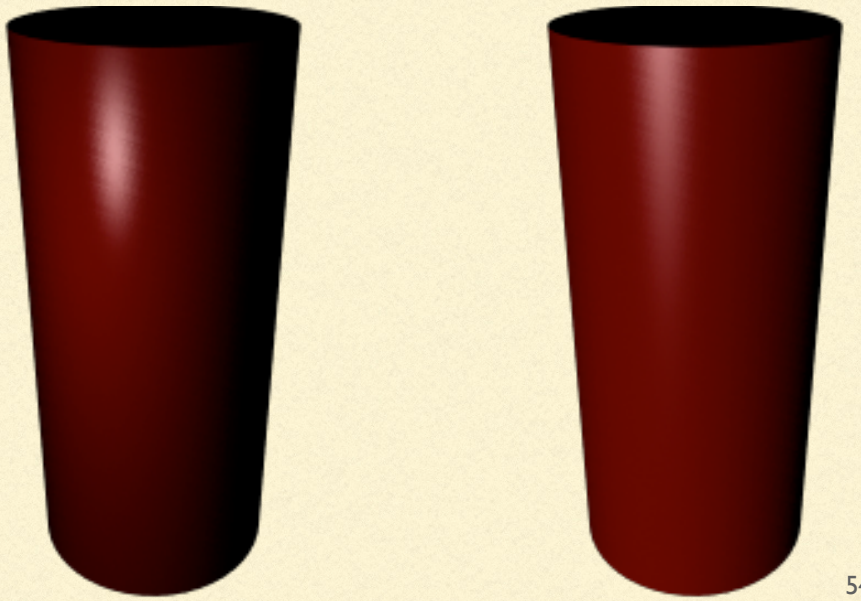
Z-Buffers

- Transparency requires partial sorting:



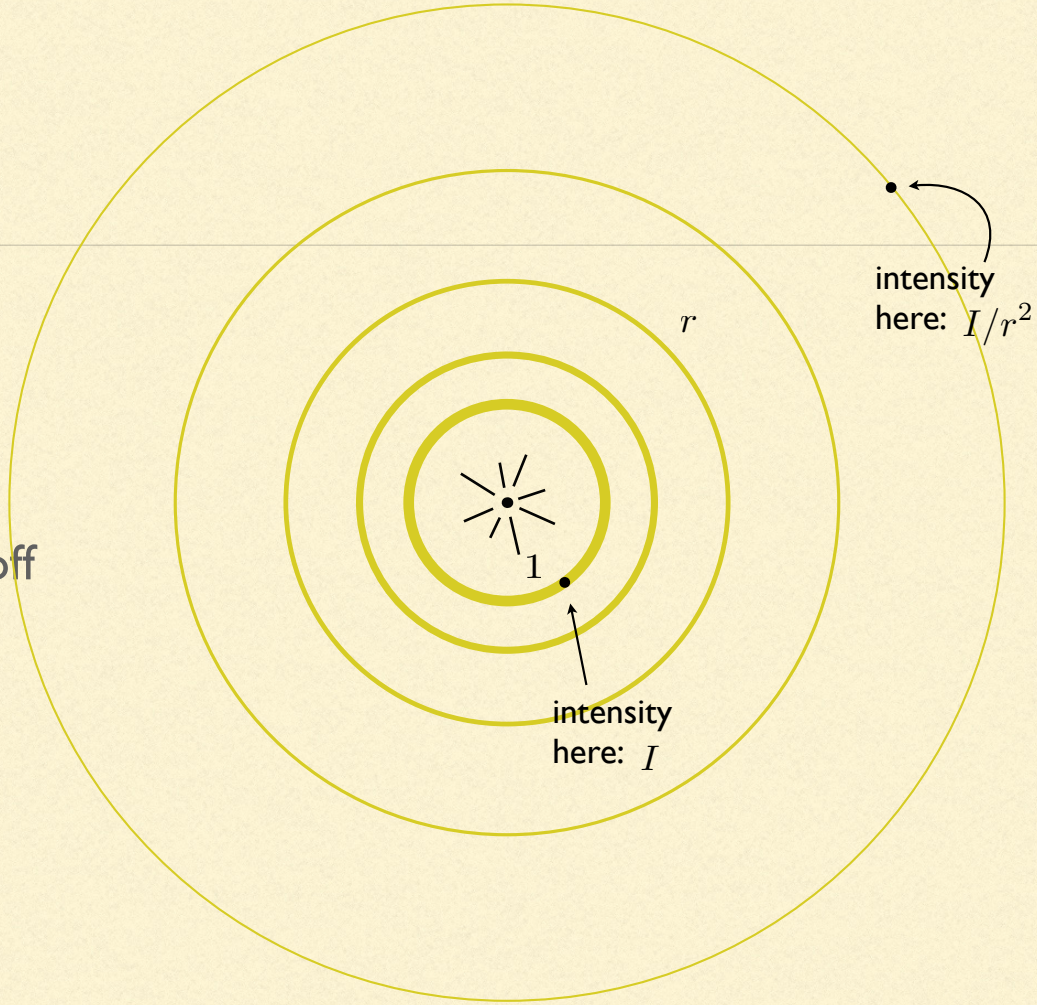
Direction -vs- Point Lights

- For a point light, the light direction changes over the surface
- For "distant" light, the direction is constant
- Similar for orthographic/perspective viewer



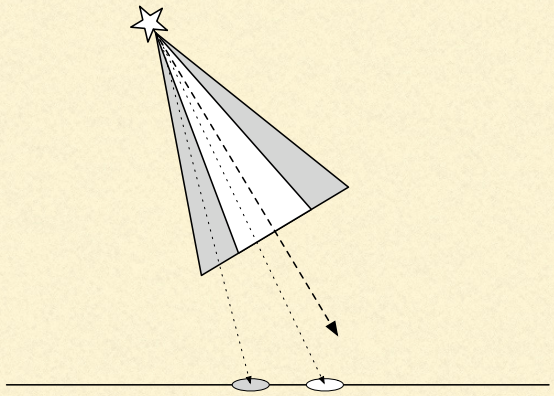
Falloff

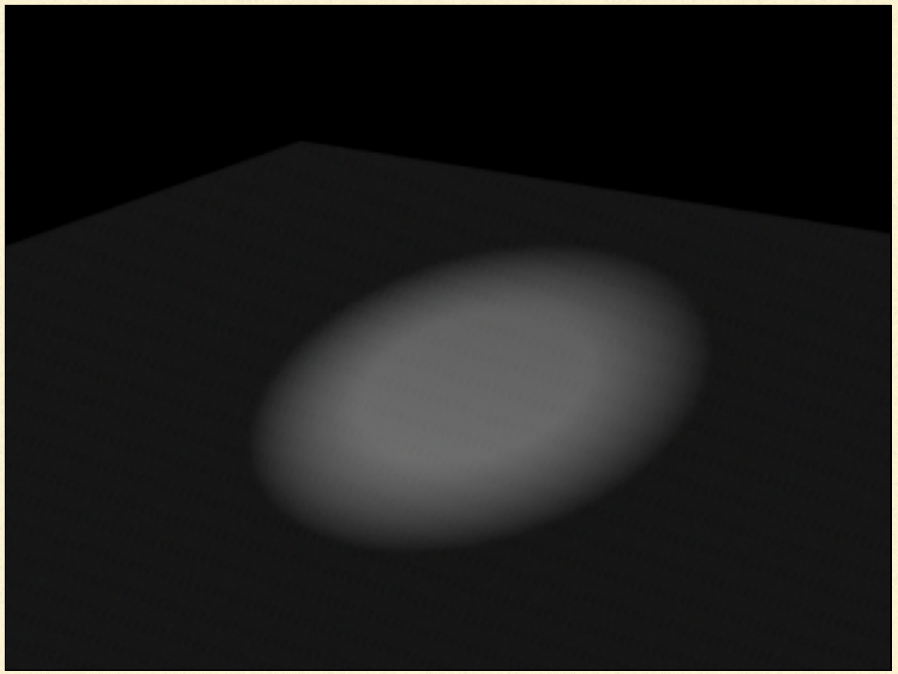
- Physically correct: $1/r^2$ light intensify falloff
  - Tends to look bad (why?)
  - Not used in practice
- Sometimes compromise of $1/r$ used



Spot and Other Lights

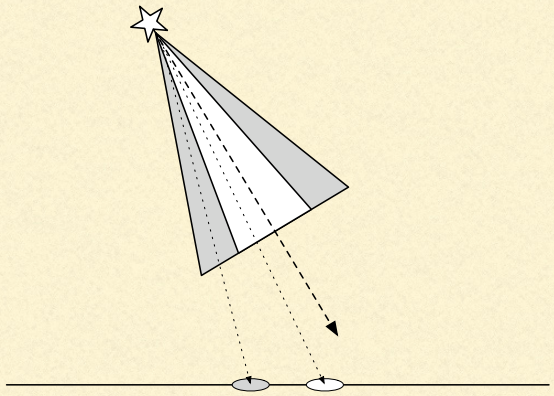
- Other calculations for useful effects
  - Spot light
  - Only light certain objects
  - Negative lights
  - *etc.*

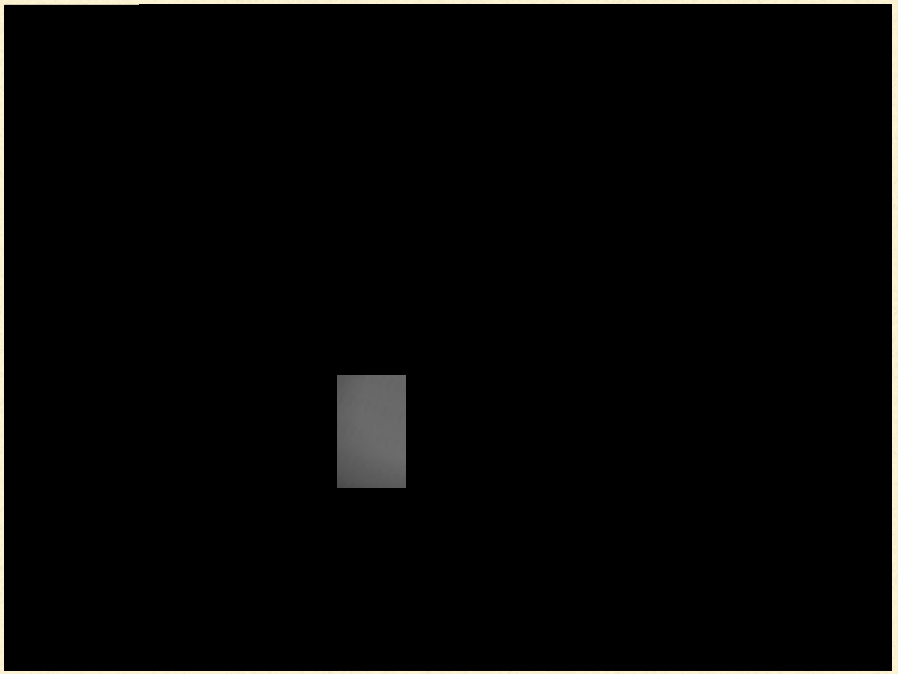




Spot and Other Lights

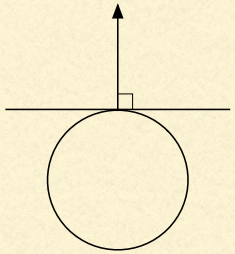
- Other calculations for useful effects
  - Spot light
  - Only light certain objects
  - Negative lights
  - *etc.*

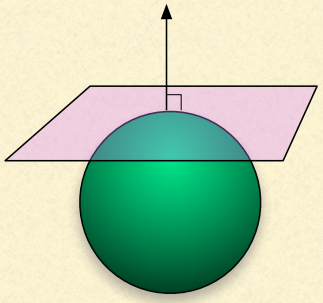




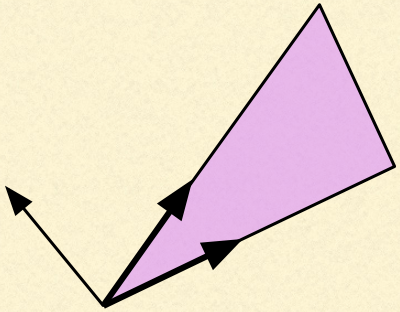
Surface Normals

- The normal vector at a point on a surface is perpendicular to all surface tangent vectors



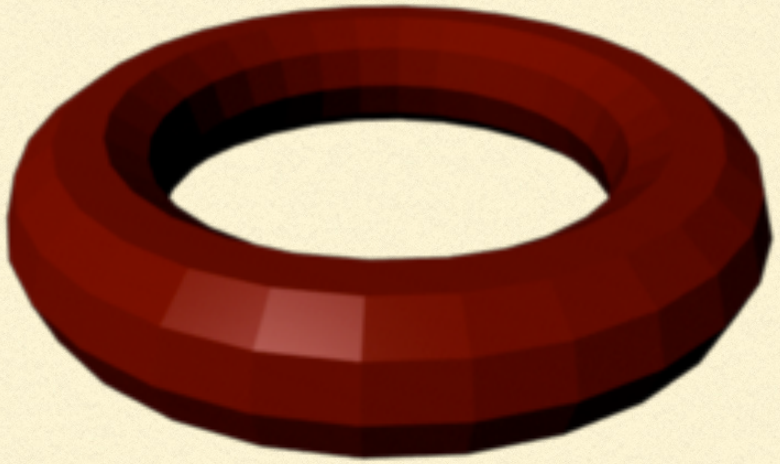


- For triangles normal given by right-handed cross product



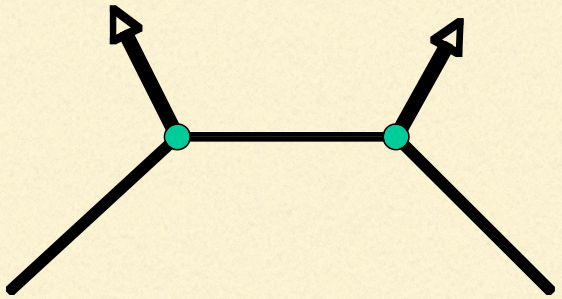
Flat Shading

- Use constant normal for each triangle (polygon)
  - Polygon objects don't look smooth
  - Faceted appearance very noticeable, especially at specular highlights
  - Recall mach bands...

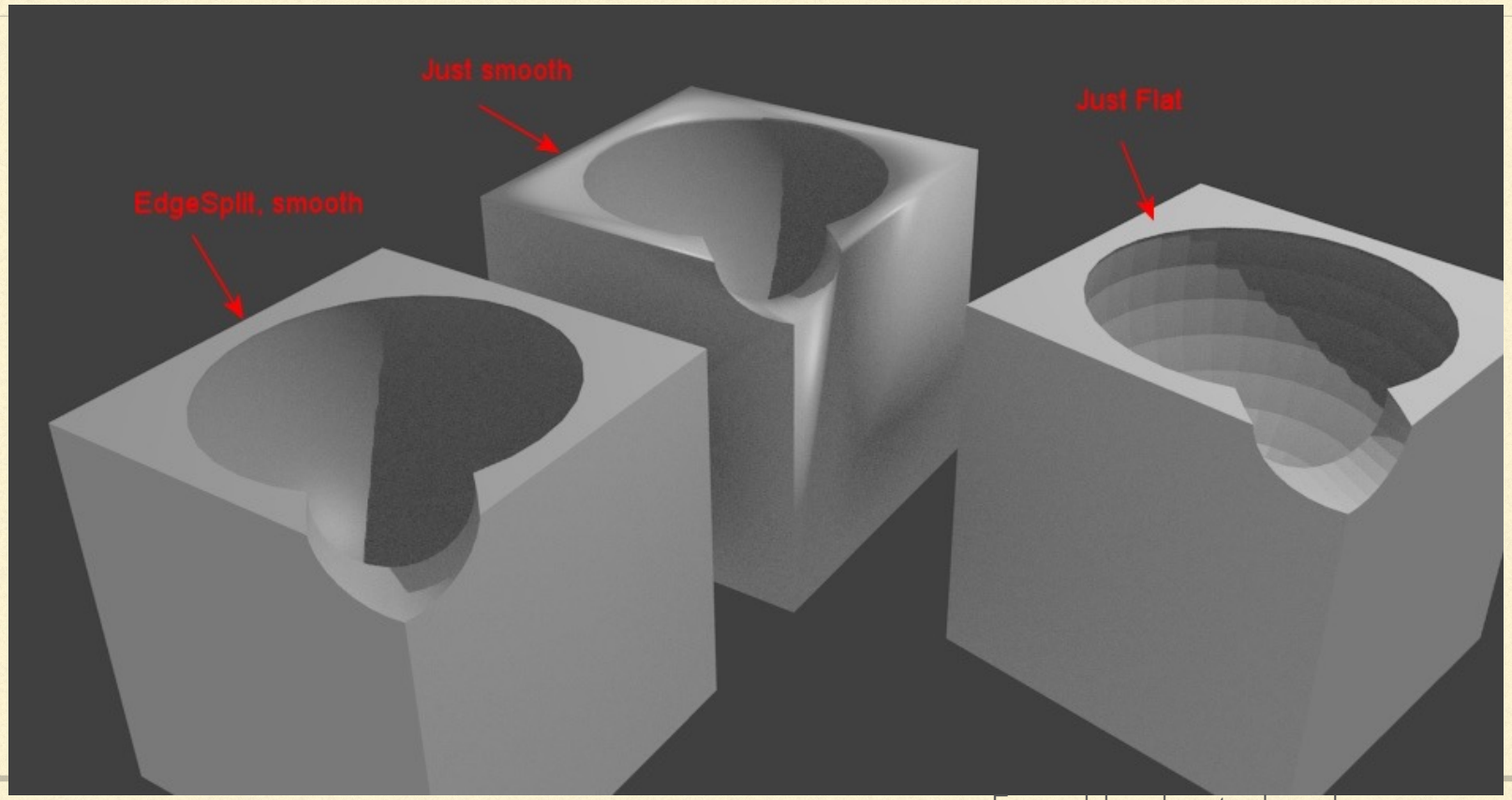


Smooth Shading

- Compute "average" normal at vertices
- Interpolate across polygons
- Use threshold for "sharp" edges
  - Vertex may have different normals for each face



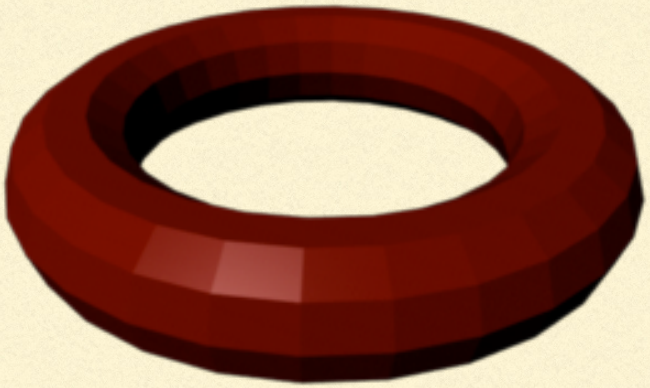
Smooth Shading



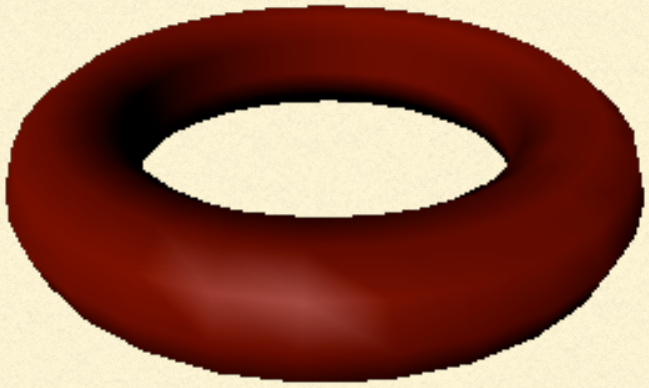
From blender.stackexchange.com

Gouraud Shading

- Compute shading at each vertex
  - Interpolate colors from vertices
  - Pros: fast and easy, looks smooth
  - Cons: terrible for specular reflections



Flat

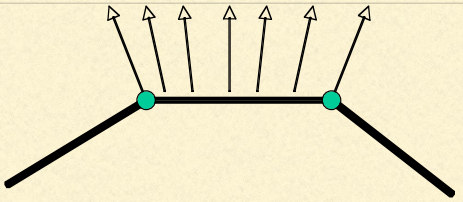


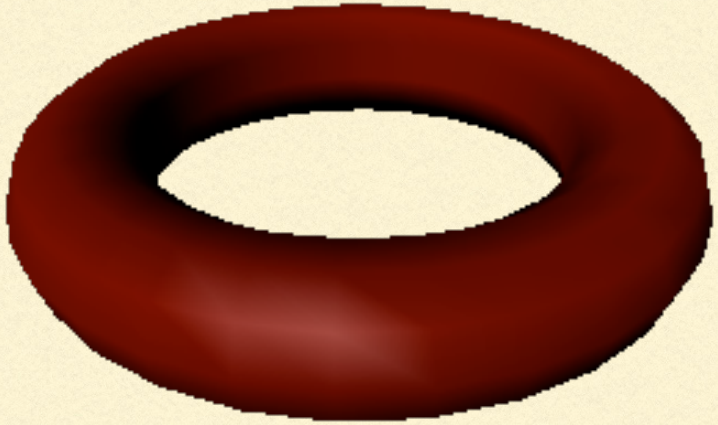
Gouraud

Note: Gouraud was hardware rendered...

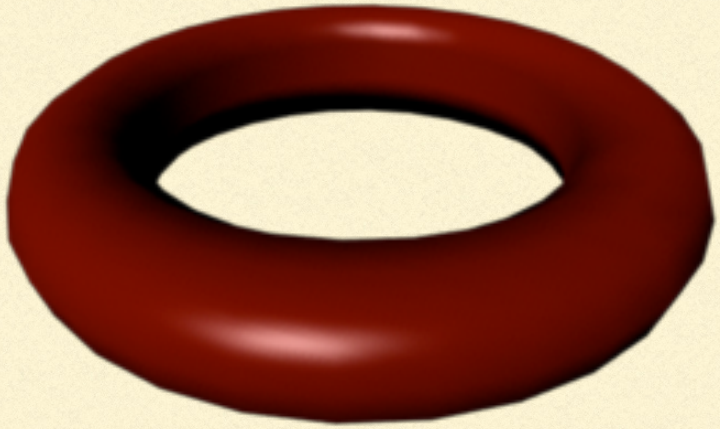
Phong Shading

- Compute shading at each pixel
  - Interpolate *normals* from vertices
  - Pros: looks smooth, better speculars
  - Cons: expensive





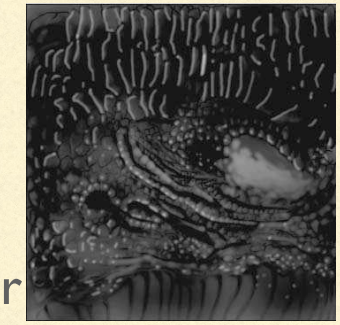
Gouraud



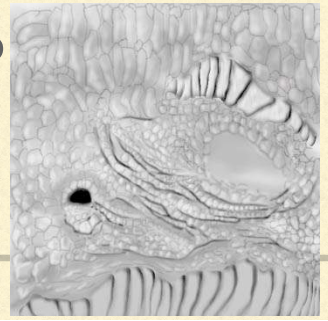
Phong

2D Texture Mapping of Images

- Use a 2D image and map it to the surface of an object

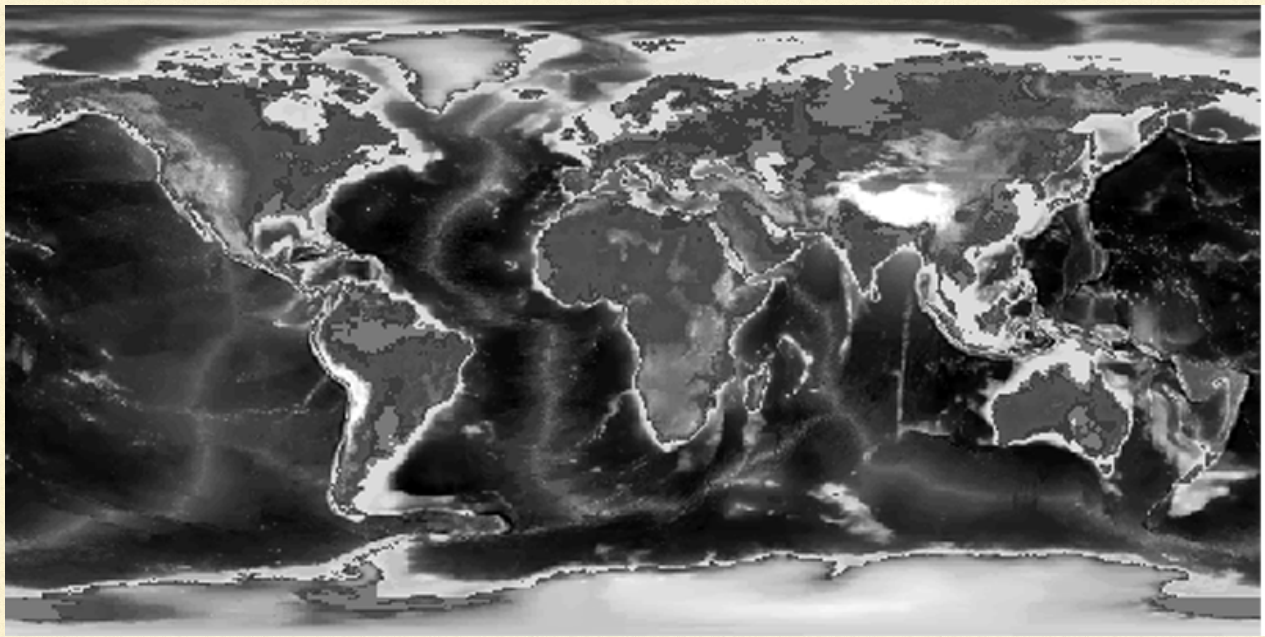
Specular 

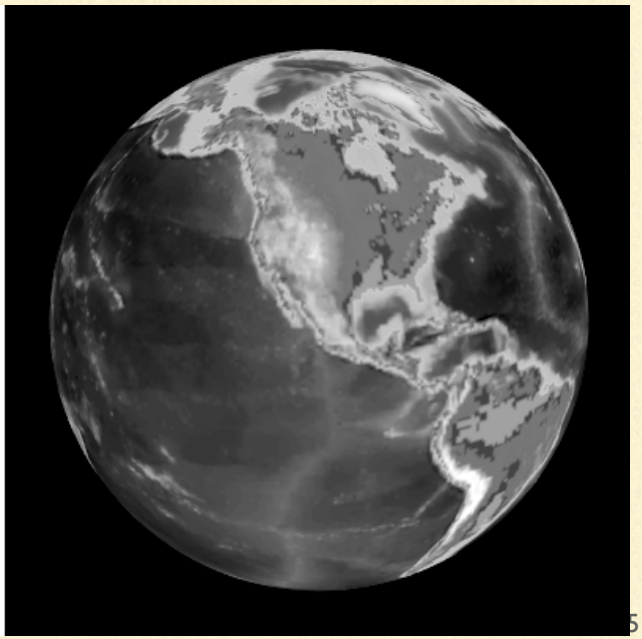
Color

Bump 

2D Texture Mapping of Images

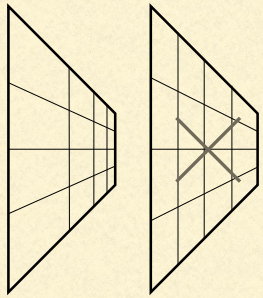
- Example of texture distortion

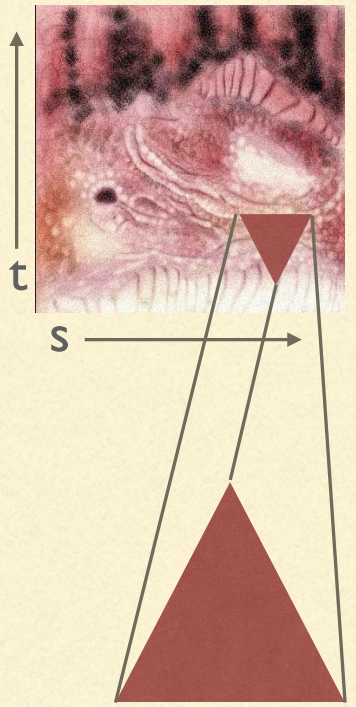




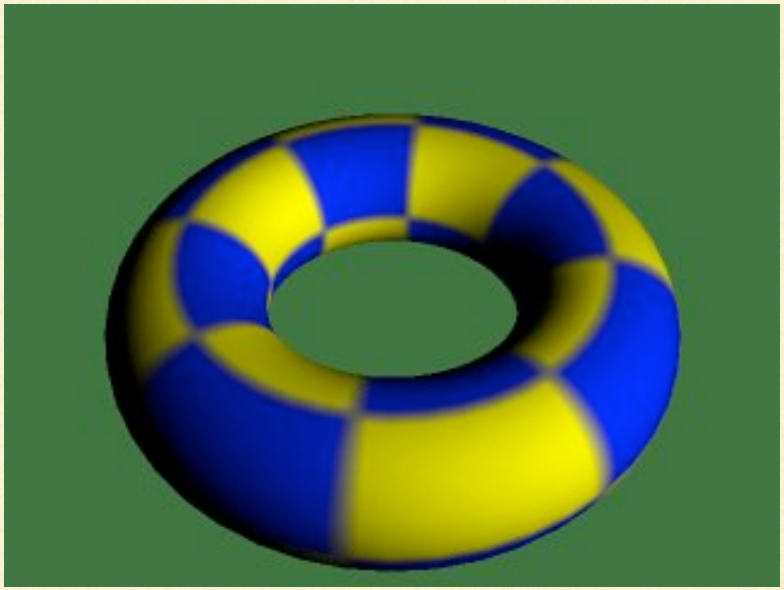
Texture Coordinates

- Assign coordinates to each vertex
- Within each triangle use linear interpolation
- Correct for distortion!

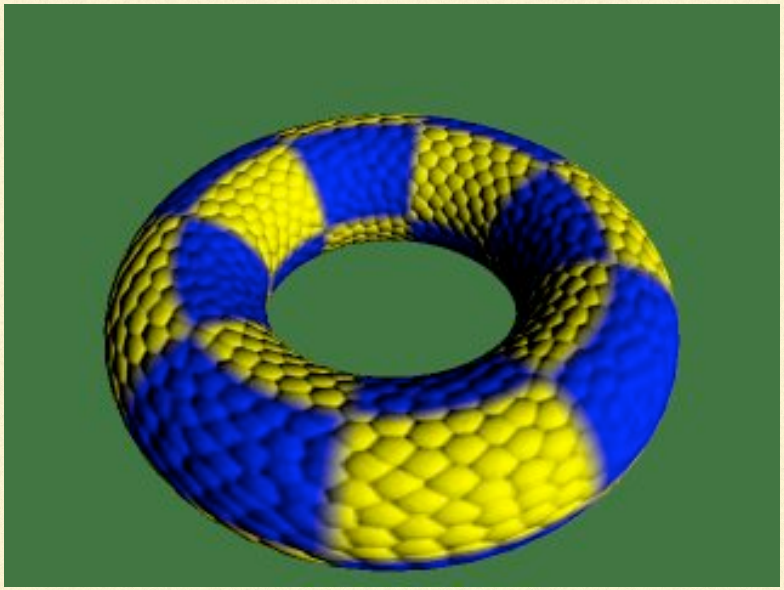




Bump Mapping



No bump mapping



With bump mapping

Images by Paul Baker
www.paulsprojects.net

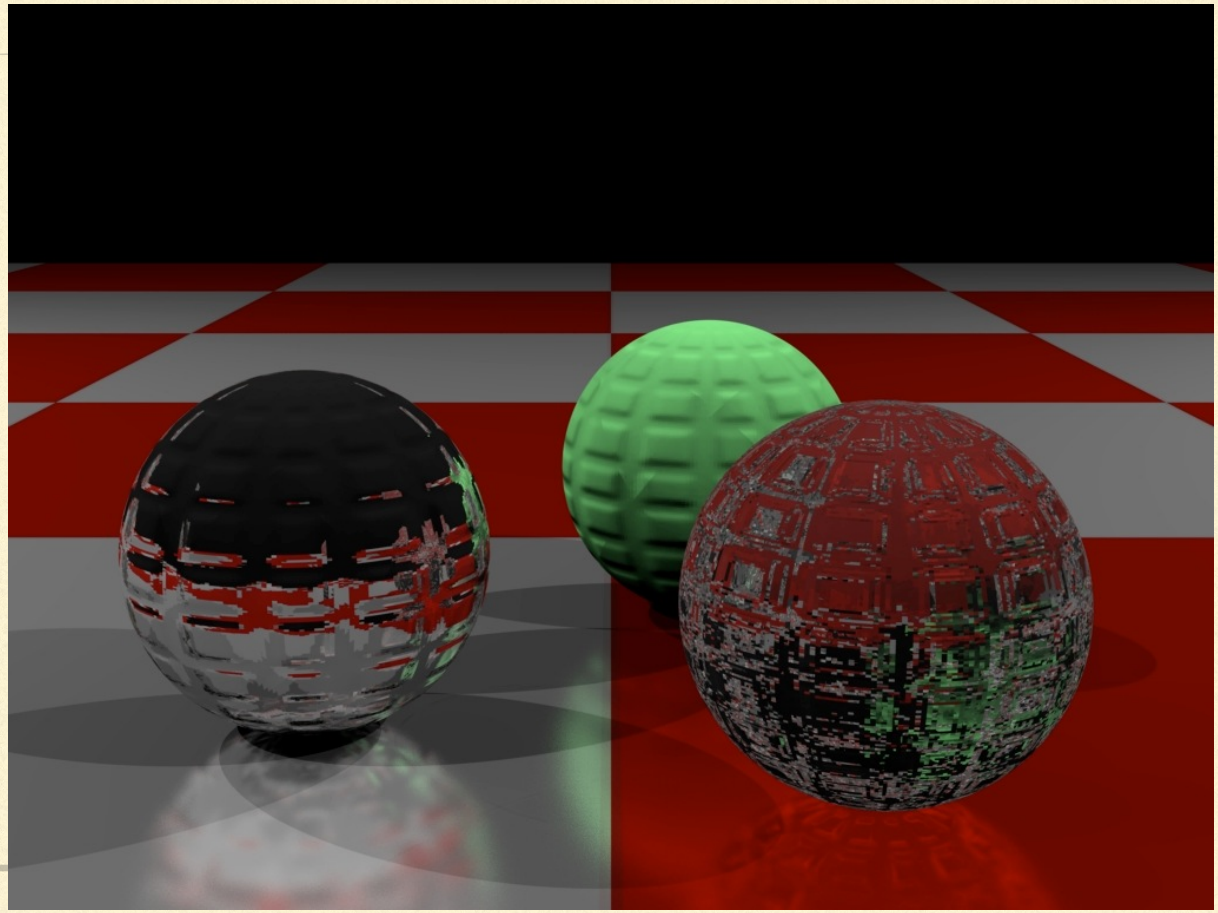
Bump Mapping

- Add offset to normal
  - Offset is in texture coordinates S,T,N
  - Store normal offsets in RGB image components
  - Should use correctly orthonormal coordinate system
- Normal offsets from gradient of a grayscale image

$$\mathbf{b}(u,v) = [s,t,n](u,v) = \nabla i(u,v)$$

$$\nabla = \left[\frac{\partial}{\partial u}, \frac{\partial}{\partial v}\right]^{\mathsf{T}}$$

Bump Map Example



Catherine Bendebury and Jonathan Michaels
CS 184 Spring 2005

2D Texture Mapping of Images

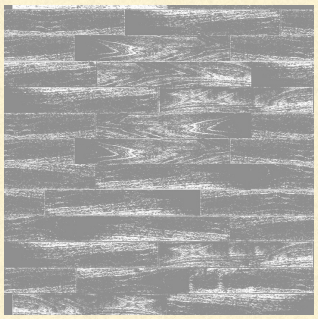
- Use a 2D image and map it to the surface of an object

2D Texture Mapping of Images

- Use a 2D image and map it to the surface of an object

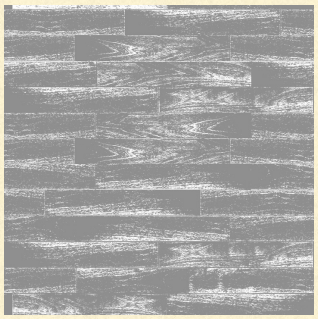
2D Texture Mapping of Images

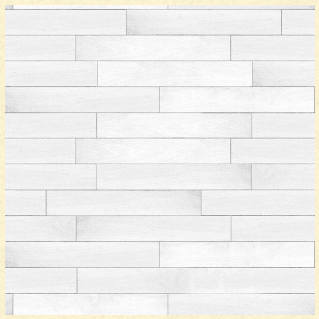
- Use a 2D image and map it to the surface of an object



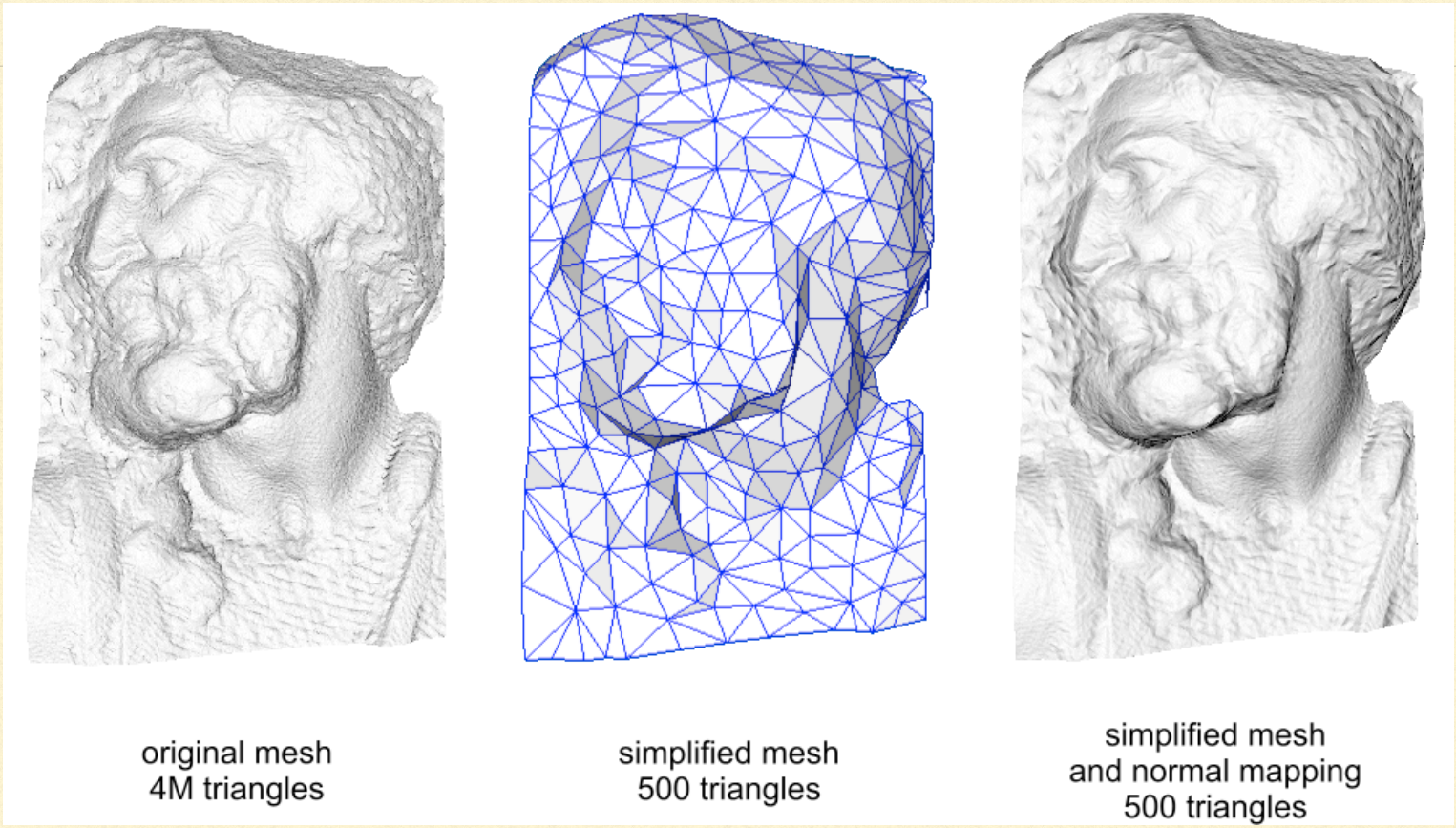
2D Texture Mapping of Images

- Use a 2D image and map it to the surface of an object



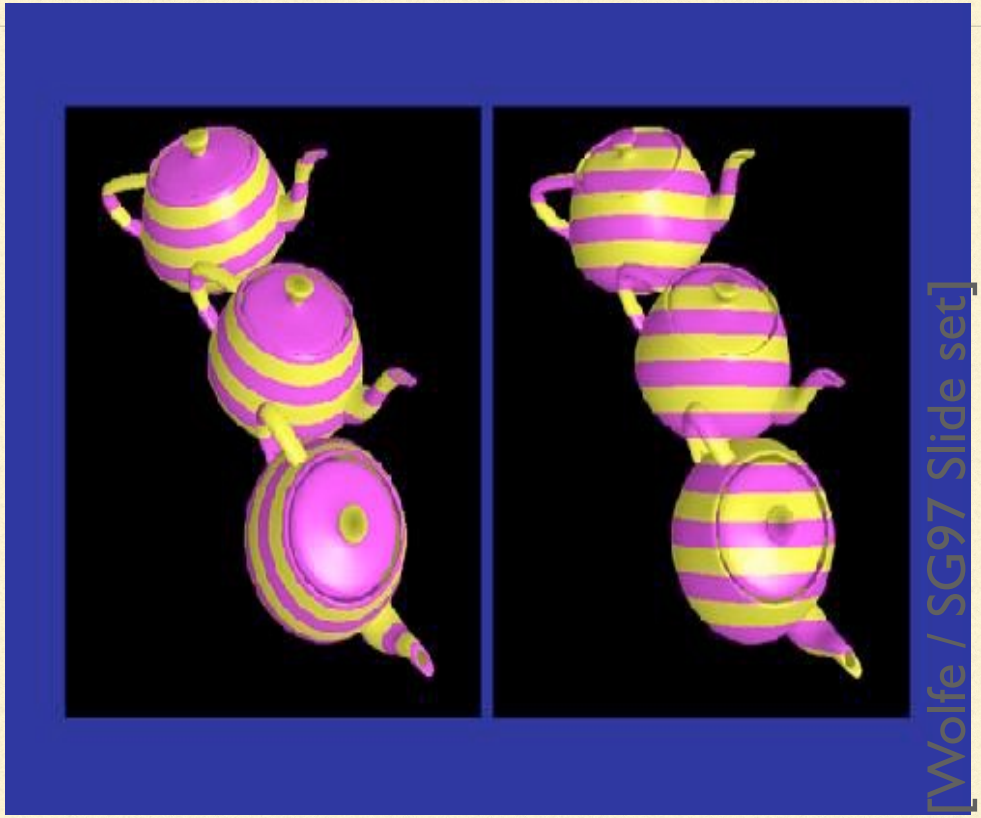


Normals Maps



[Paolo Cignoni]

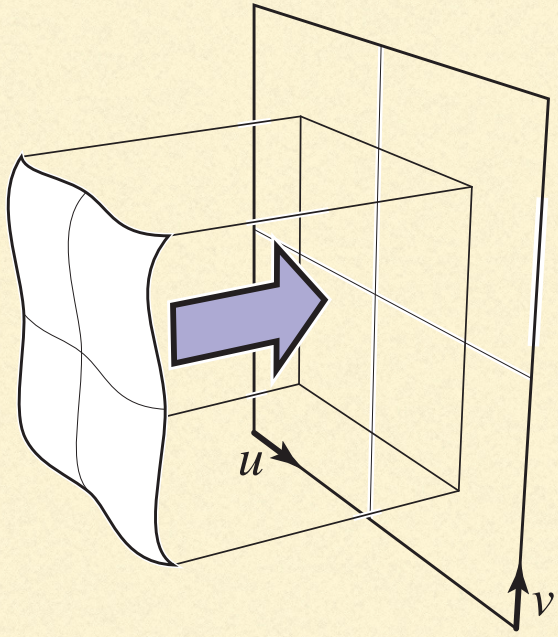
Texture Coordinate Assignment



[Wolfe / SG97 Slide set]

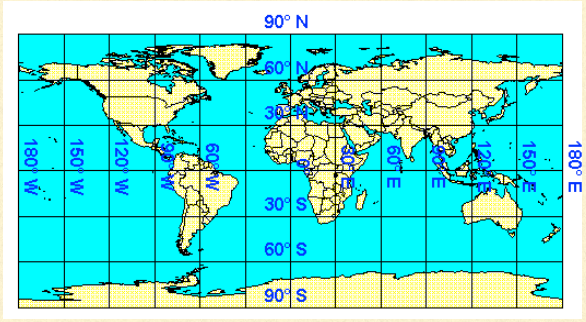
Texture Coordinate Assignment

- Planar projection



Marschner, 2014

Texture Coordinate Assignment

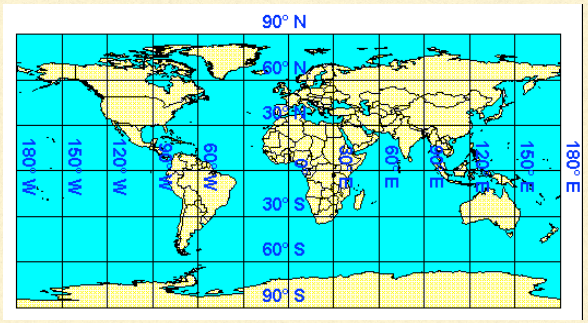


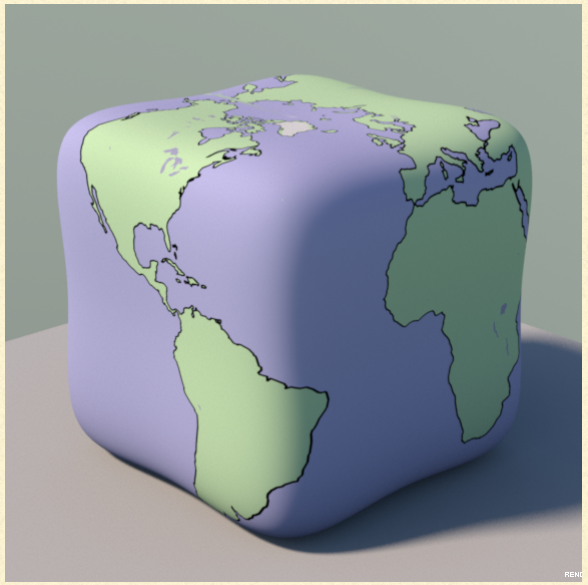
Marschner, 2014

- Spherical Coordinates

Texture Coordinate Assignment

- Spherical Projection



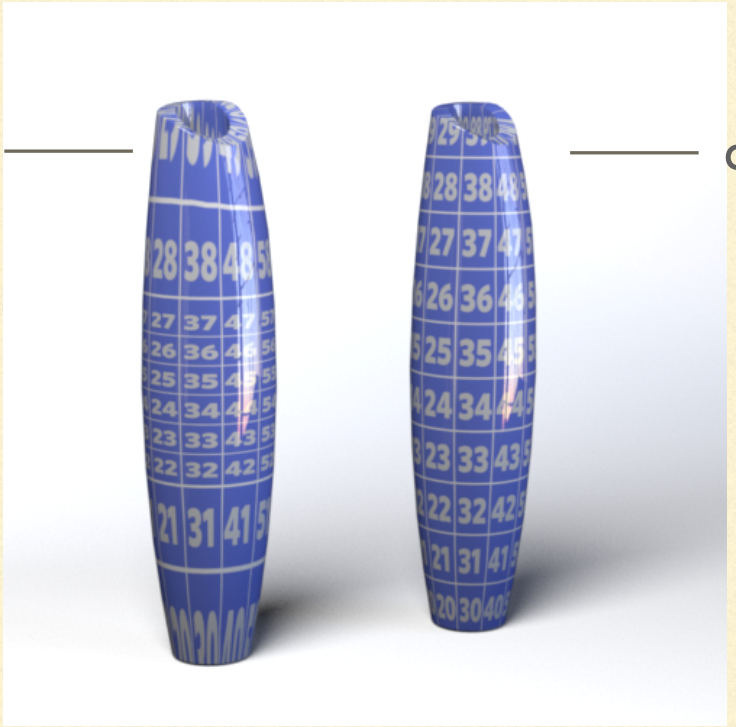


Marschner, 2014

Texture Coordinate Assignment

- Cylindrical Projection

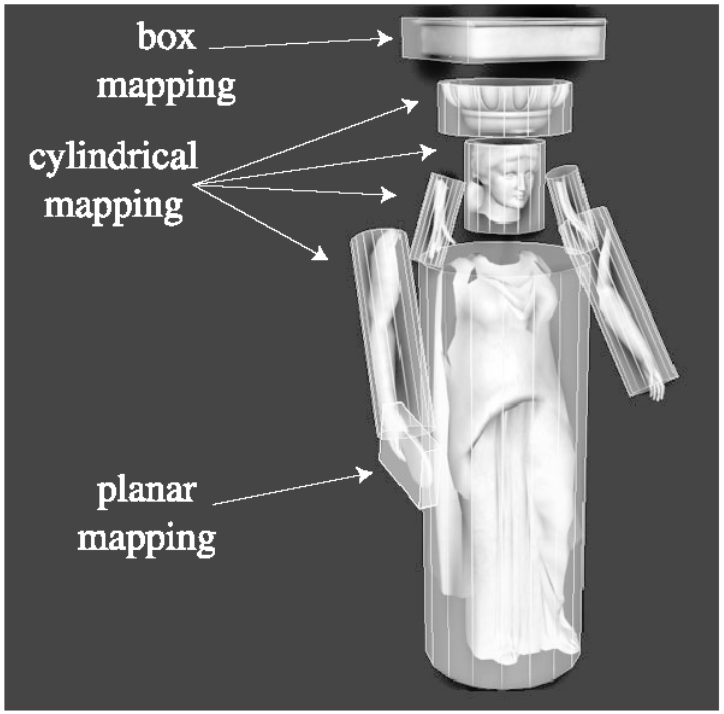
spherical —— —— cylindrical



Marschner, 2014

Texture Coordinate Assignment

- Multiple Projections



[Tito Pagan]

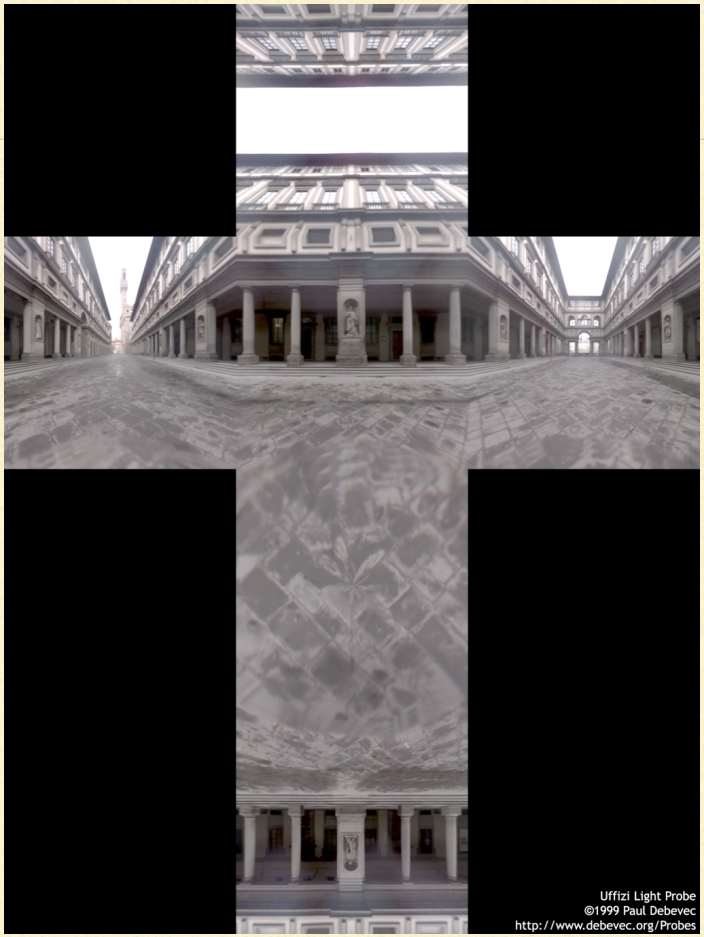
Texture Coordinate Assignment

- 3D Textures

[Wolfe / SG97 Slide set]

Environment Maps

- Environment maps allow crude reflections
- Treat object as infinitesimal
  - Reflection only based on surface normal
- Errors hard to notice for non-flat objects

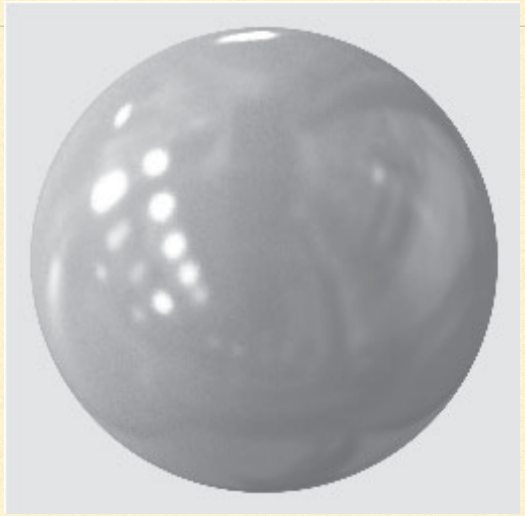


Environment Maps

Images by Paul Haeberli

- Sphere based parameterization
  - Wide angle image or
  - Photo of a silver ball

Environment Maps



(a)

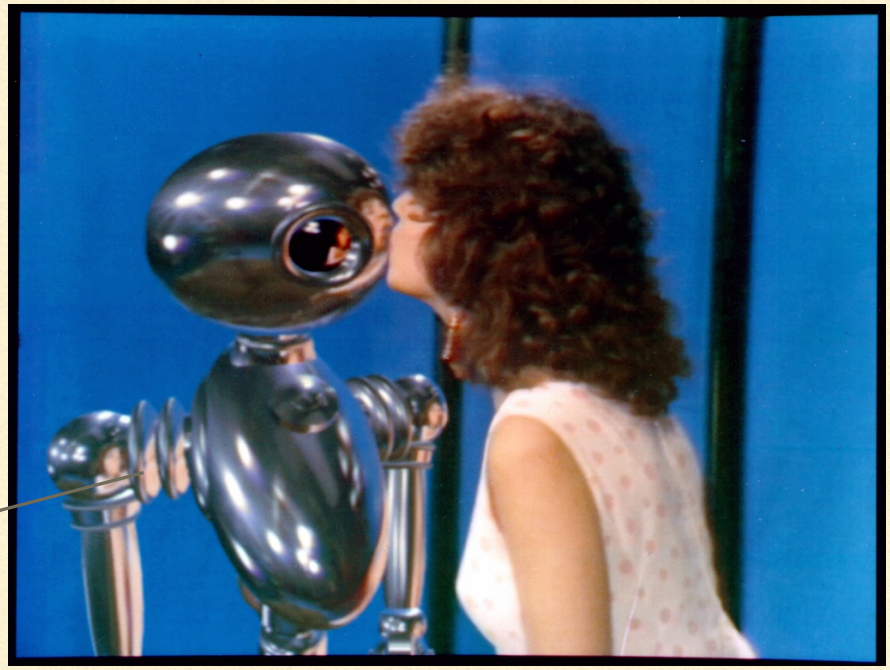
(b)

Dror, Willsky, & Adelson 2004

Figure 2. (a). A shiny sphere rendered under photographically acquired real-world illumination. (b). The same sphere rendered under illumination by a point light source.

Environment Maps

- Used in 1985 in movie Interface
  - Lance Williams from the New York Institute of Technology



Note errors

Environment Maps

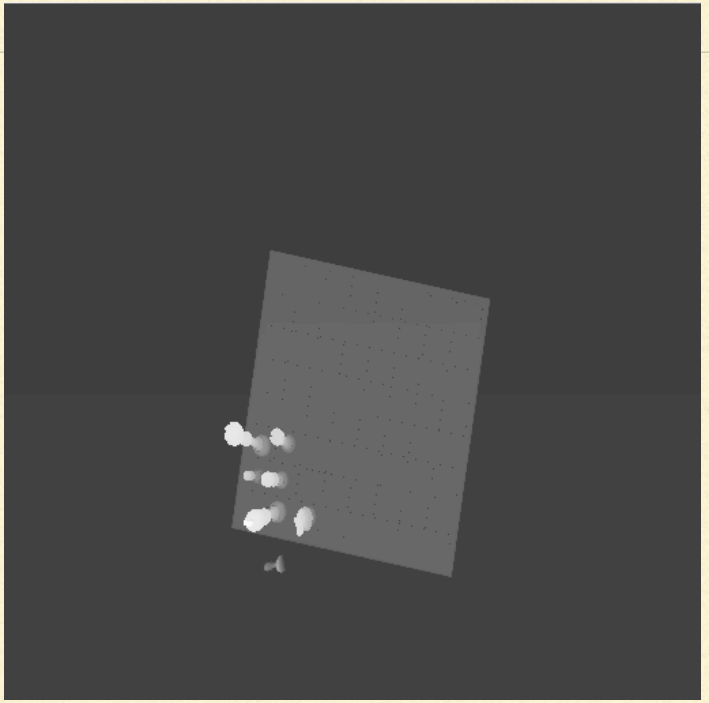
- Used in 1985 in movie Interface
  - Lance Williams from the New York Institute of Technology

Shadow Maps

- Pre-render scene from perspective of light source
  - Only render Z-Buffer (the shadow buffer)
- Render scene from camera perspective
  - Compare with shadow buffer
  - If nearer light, if further shadow

Shadow Maps

From Stamminger and Drettakis
SIGGRAPH 2002



Shadow Buffer

Image w/ Shadows

Note: These images don't really go together, see the paper...

Photon Mapping

A basic ray traced image

Note:
- Dark shadows
- Unlit corners
- Nice reflections

Image by Per Christensen

Photon Mapping

Raw photons

Note:
- Noisy
- Sparse

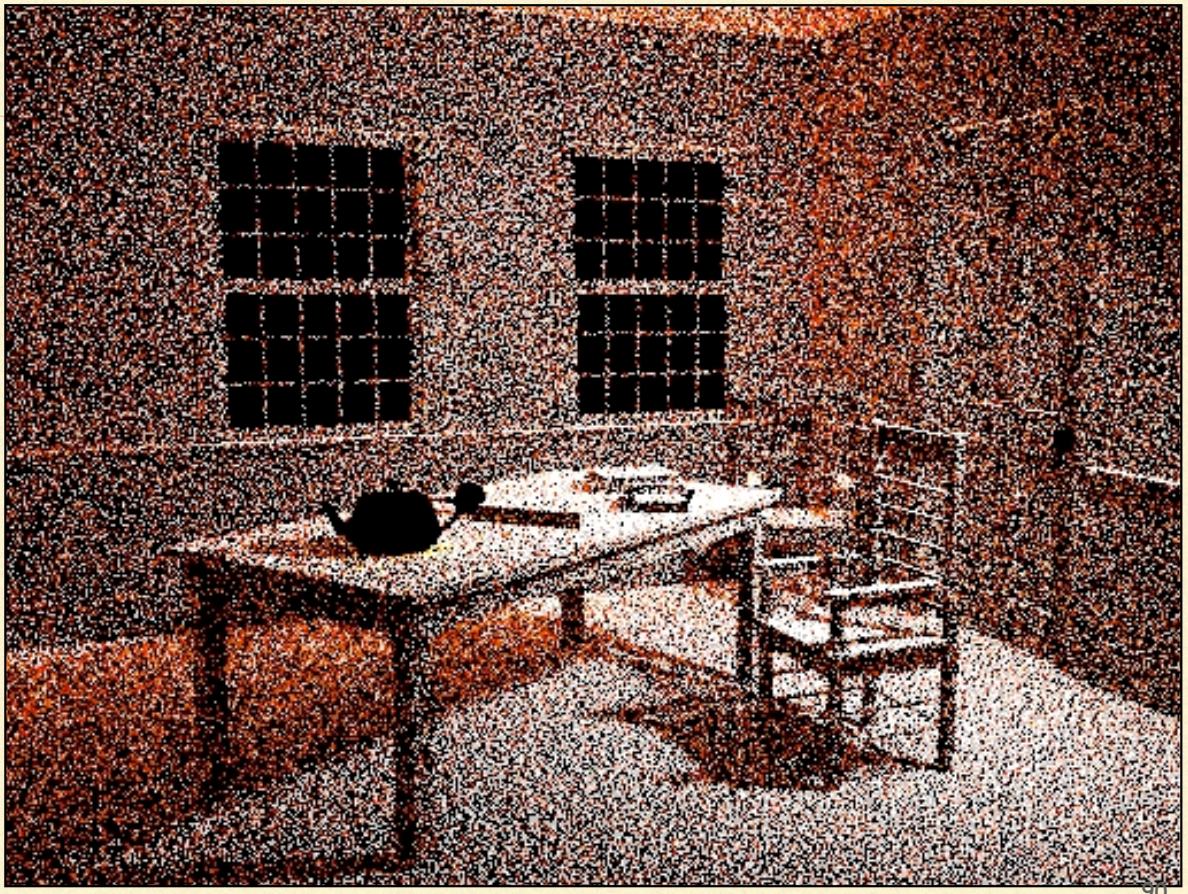


Image by Per Christensen

Photon Mapping

Interpolated Photons
(multiplied by diffuse)

Note:
- Still noisy
- Biased

Image by Per Christensen

Photon Mapping

Final Image

Note:
- Not noisy
- Nice lighting
- Reflections
- May still be biased

- Final gather often bottleneck...

Image by Per Christensen

RADIOSITY

Diffuse transport only
  View-independent solution

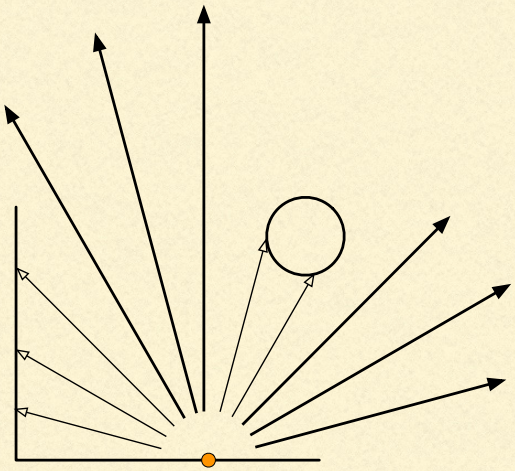
Can be "baked" into models
  Vertex colors or texture

from Hanrahan 2000

Ambient Occlusion

- A "hack" to create more realistic ambient illumination cheaply
- Assume light from everywhere is partially blocked by local objects
  - At a point on the surface cast rays at random
  - Ambient term is proportional to percent of rays that hit nothing
  - Weight average by cosine of angle with normal
  - Optional: Take into account how far before occluded
  - Optional: consider color of occluding object

Ambient Occlusion

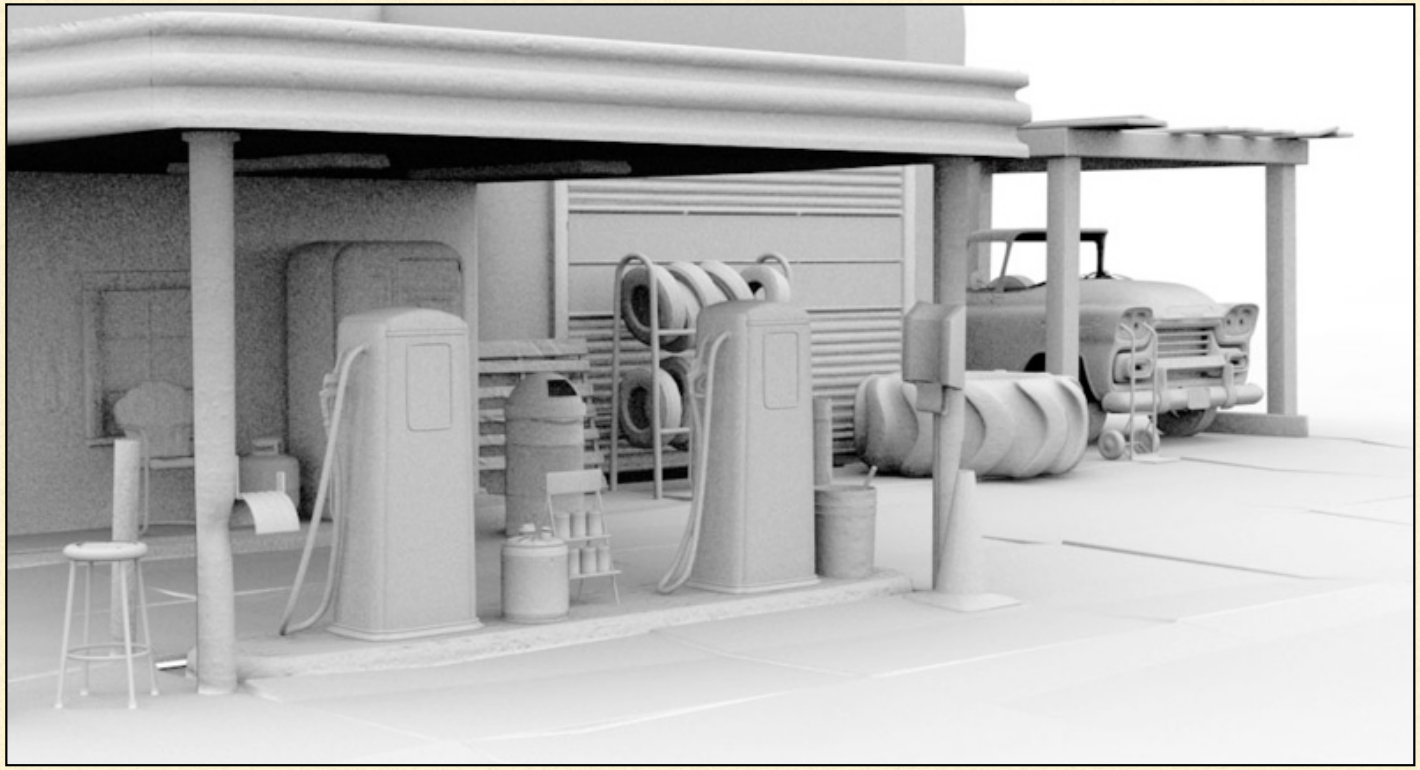


Diffuse Only

Ambient Occlusion

Combined

Ambient Occlusion



nVidia Gelato Demo Image

360 + Stereo ??

Pre-baked Lighting? Radiosity?