# CS-184: Computer Graphics

Lecture #8: Shading

Prof. James O'Brien
University of California, Berkeley

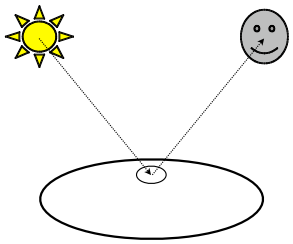
V2005F-08-2.0

# Today

- Local Illumination & Shading
  - The BRDF
  - Simple diffuse and specular approximations
  - Shading interpolation: flat, Gouraud, Phong
  - Some miscellaneous tricks
- Normal Vectors

# Local Shading

- Local: consider in isolation
  - 1 light
  - 1 surface
  - The viewer



- Recall: lighting is linear
  - Almost always...

Counter example: photochromatic materials



# Local Shading

- Examples of non-local phenomena
  - Shadows
  - Reflections
  - Refraction
  - Indirect lighting

# The BRDF

- The **B**i-directional **R**eflectance **D**istribution **F**unction
- Given
  - Surface material
  - Incoming light direction
  - Direction of viewer
  - Orientation of surface

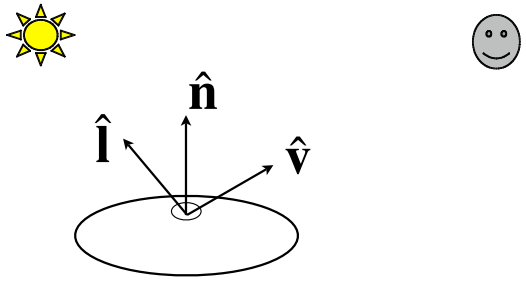
$$\rho = \rho(\theta_V, \theta_L)$$
$$= \rho(\mathbf{v}, \mathbf{l}, \mathbf{n})$$

- Return:
  - fraction of light that reaches the viewer
- We'll worry about physical units later...



# The BRDF

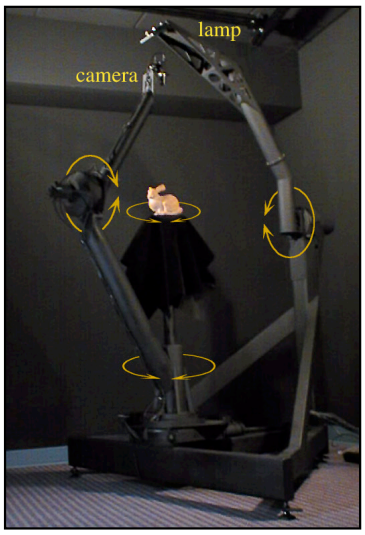
$\rho(\mathbf{v}, \mathbf{l}, \mathbf{n})$

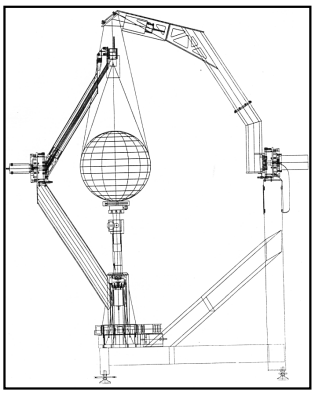



- Spatial variation capture by "the material"
- Frequency dependent
  - Typically use separate RGB functions
  - Does not work perfectly
  - Better: $\rho = \rho(\theta_V, \theta_L, \lambda_{in}, \lambda_{out})$

# Obtaining BRDFs

- Measure from real materials



Images from Marc Levoy



# Obtaining BRDFs

- Measure from real materials
- Computer simulation
  - Simple model + complex geometry
- Derive model by analysis
- Make something up

# Beyond BRDFs

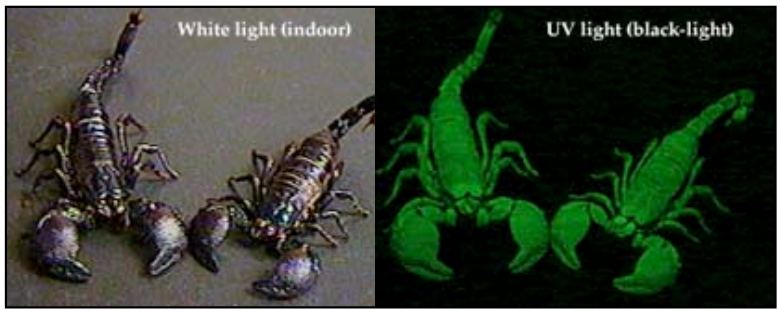
- The BRDF model does not capture everything
  - *e.g.* Subsurface scattering (BSSRDF)

Images from Jensen *et. al, SIGGRAPH 2001*



# Beyond BRDFs

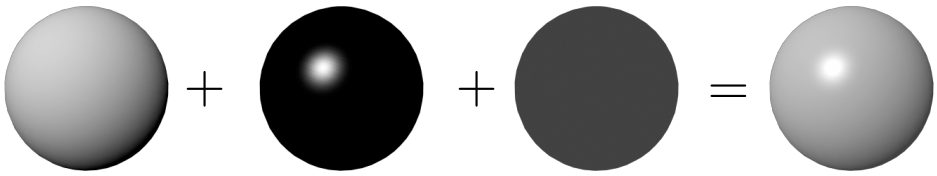
- The BRDF model does not capture everything
  - *e.g.* Inter-frequency interactions




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

# A Simple Model

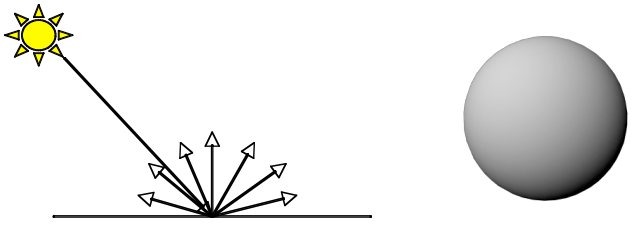
- Approximate BRDF as sum of
  - A diffuse component
  - A specular component
  - A "ambient" term





# 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

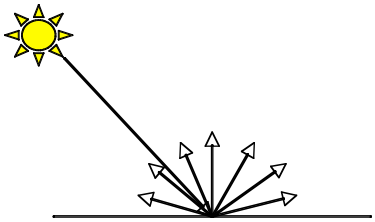


# Diffuse Component

Comment about two-side lighting in text is wrong...

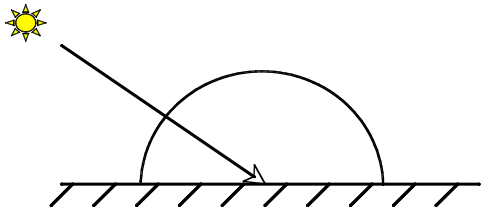
$$k_d I(\hat{\mathbf{l}} \cdot \hat{\mathbf{n}})$$

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

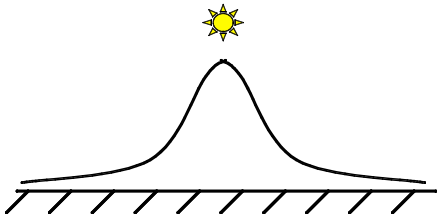


# Diffuse Component

- Plot light leaving in a given direction:

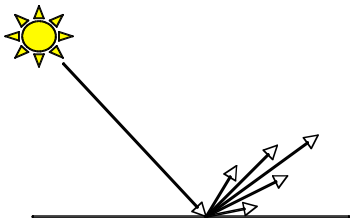


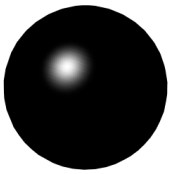
- Plot light leaving from each point on surface



# 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



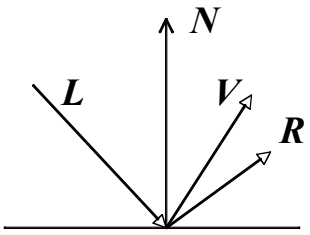


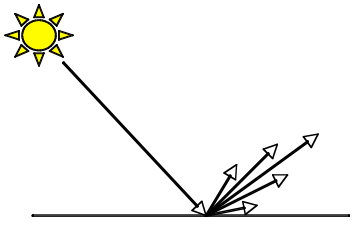


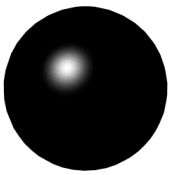
# Specular Component

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

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



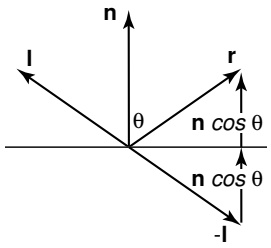





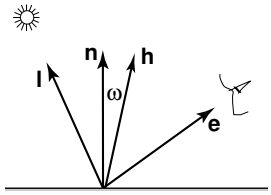
# Specular Component

- Computing the reflected direction

$$\hat{\mathbf{r}} = -\hat{\mathbf{l}} + 2(\hat{\mathbf{l}} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}$$



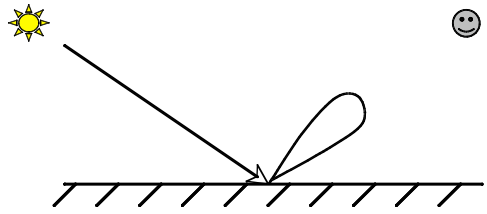

$$\hat{\mathbf{h}} = \frac{\hat{\mathbf{l}} + \hat{\mathbf{v}}}{||\hat{\mathbf{l}} + \hat{\mathbf{v}}||}$$



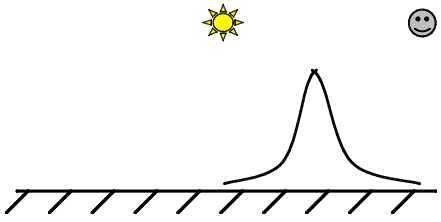



# Specular Component

- Plot light leaving in a given direction:

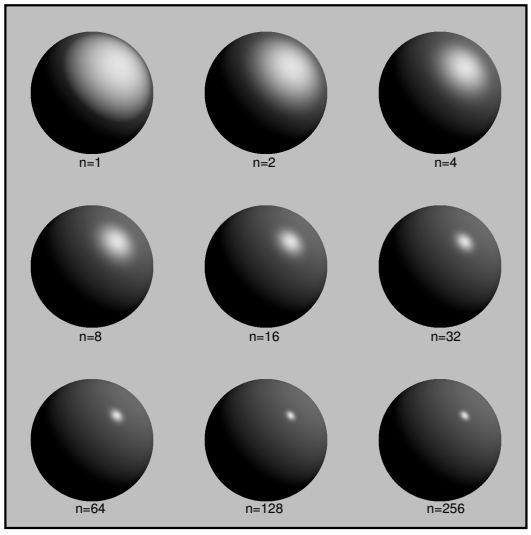


- Plot light leaving from each point on surface



# Specular Component

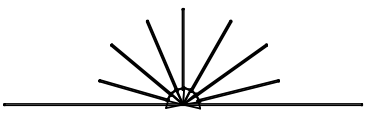
- Specular exponent sometimes called "roughness"





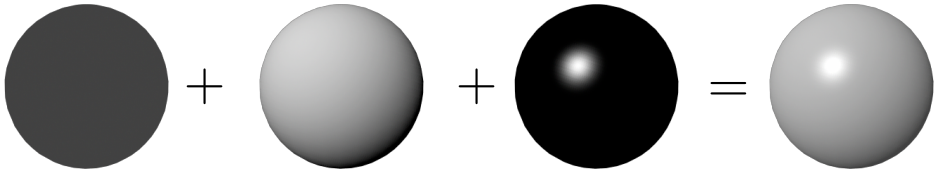

# Ambient Term

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



# Summing the Parts

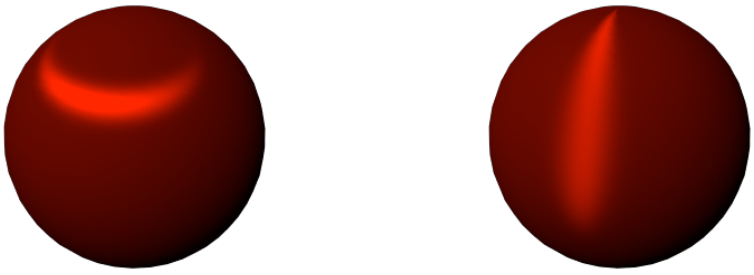
$$R = k_a I + k_d I \max(\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}, 0) + k_s I \max(\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}, 0)^p$$



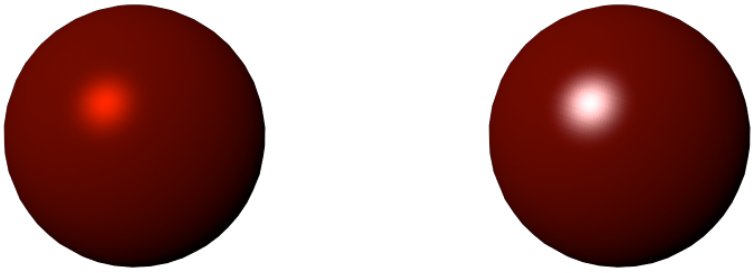
- Recall that the $k_?$ are by wavelength
  - RGB in practice
- Sum over all lights



# Anisotropy

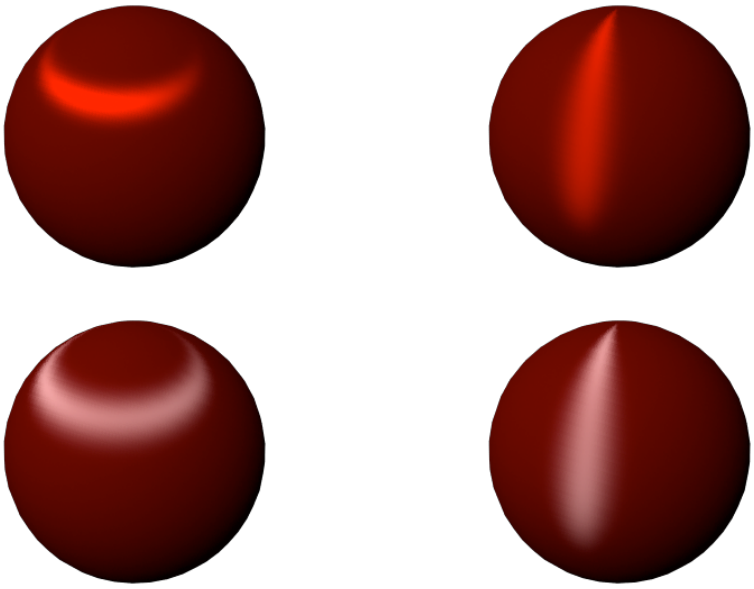


# Metal -vs- Plastic

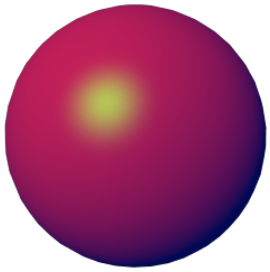




# Metal -vs- Plastic

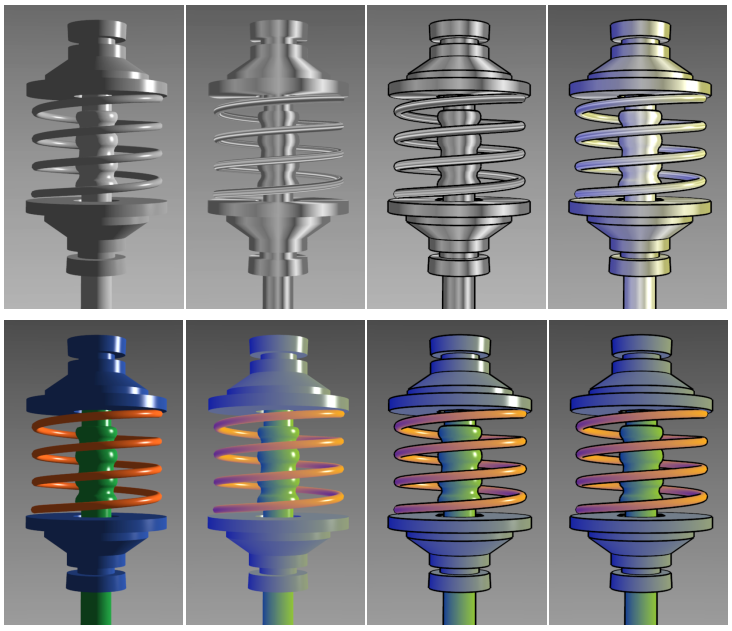


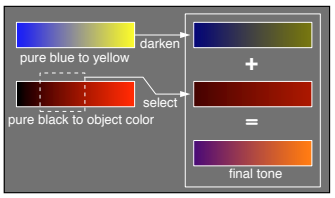
# Other Color Effects





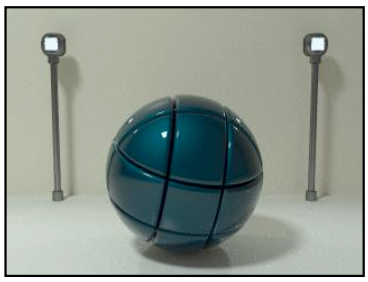
# Other Color Effects






Images from Gooch *et. al, 1998*

# Measured BRDFs



BRDFs for automotive paint

Images from Cornell University Program of Computer Graphics



# Measured BRDFs

BRDFs for aerosol spray paint

Images from Cornell University Program of Computer Graphics

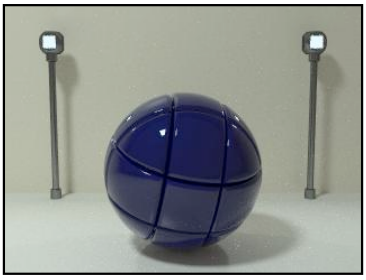
# Measured BRDFs

BRDFs for house paint

Images from Cornell University Program of Computer Graphics



# Measured BRDFs



BRDFs for lucite sheet

Images from Cornell University Program of Computer Graphics

# Details Beget Realism

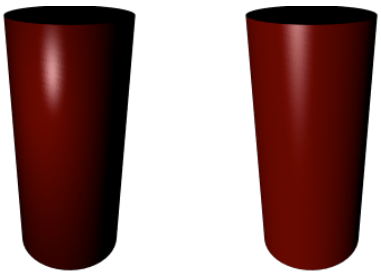
- The "computer generated" look is often due to a lack of fine/subtle details... a lack of richness.

From bustledress.com



# 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



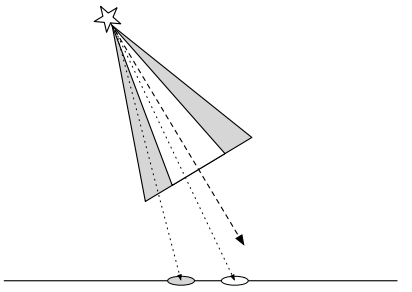
# Falloff

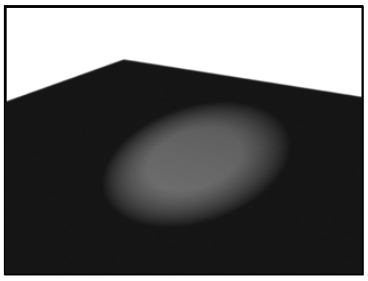
- Physically correct: $1/r^2$ light intensify falloff
  - Tends to look bad (why?)
  - Not used in practice
- Sometimes compromise of $1/r$ used



# Spot and Other Lights

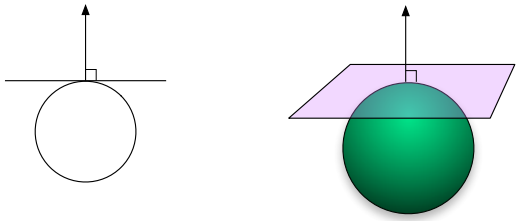
- Other calculations for useful effects
  - Spot light
  - Only light certain objects
  - Negative lights
  - *etc.*



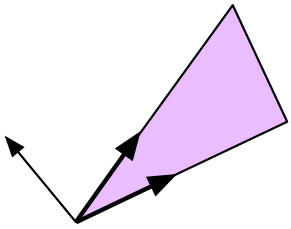


# Surface Normals

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



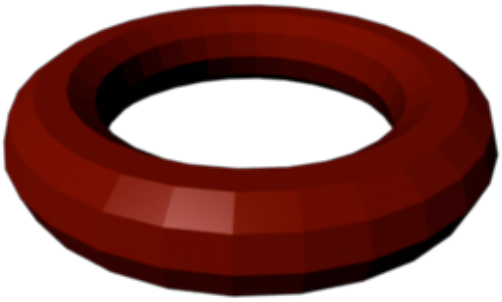
- For triangles normal given by right-handed cross product





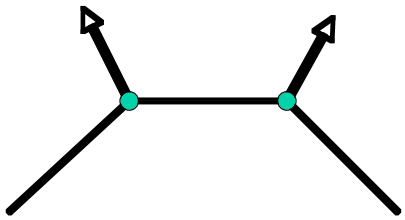
# Flat Shading

- 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

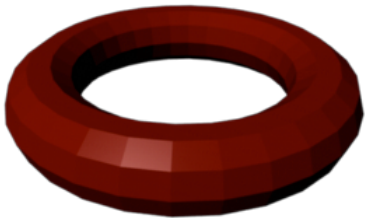
- Compute "average" normal at vertices
- Interpolate across polygons
- Use threshold for "sharp" edges
  - Vertex may have different normals for each face



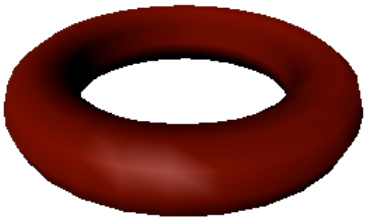


# Gouraud Shading

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



Flat



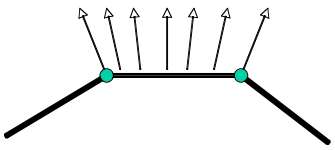
Gouraud

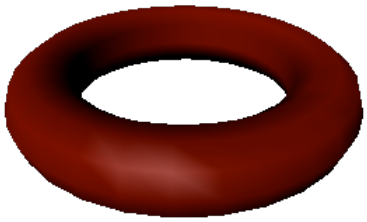


Note: Gouraud was hardware rendered...

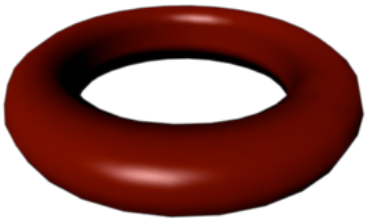
# Phong Shading

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





Gouraud



Phong



Note: Gouraud was hardware rendered...

# Transforming Normals

- Normals are directions
  - But they must maintain underlaying geometric property

$$\hat{\mathbf{n}} \cdot (\mathbf{a} - \mathbf{b}) = \mathbf{0}$$

$$\hat{\mathbf{n}}^{\mathsf{T}} (\mathbf{a} - \mathbf{b}) = \mathbf{0}$$

  - A bit of manipulation

$$\hat{\mathbf{n}}_{\text{new}}^{\mathsf{T}} (\mathbf{T}\mathbf{a} - \mathbf{T}\mathbf{b}) = \mathbf{0}$$

$$\hat{\mathbf{n}}_{\text{new}}^{\mathsf{T}} \mathbf{T}(\mathbf{a} - \mathbf{b}) = \mathbf{0}$$

$$\hat{\mathbf{n}}^{\mathsf{T}} \mathbf{T}^{-1} \mathbf{T}(\mathbf{a} - \mathbf{b}) = \mathbf{0}$$

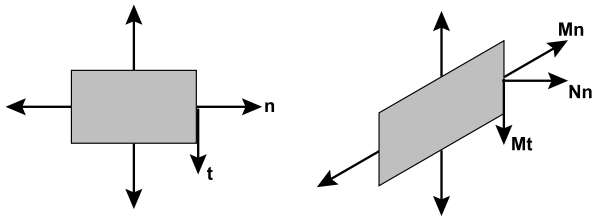
$$\boxed{\hat{\mathbf{n}}_{\text{new}} = \mathbf{T}^{-\mathsf{T}} \hat{\mathbf{n}}}$$

# Transforming Normals

- Special transformation rule for normals

$$\hat{\mathbf{n}}_{new} = \mathbf{T}^{-\mathsf{T}}\hat{\mathbf{n}}$$

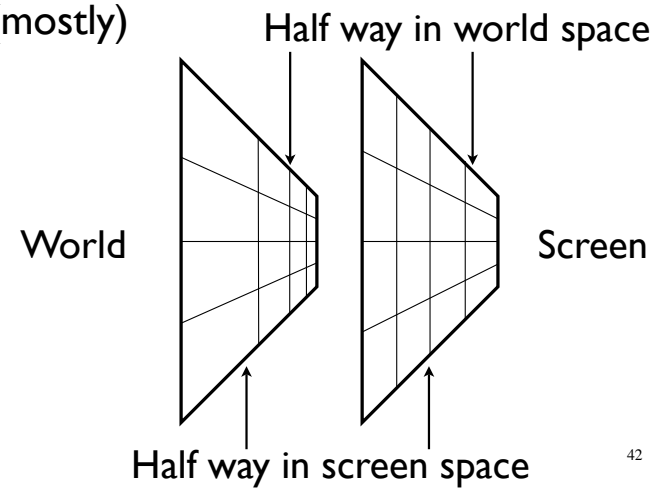
- Note: A rotation is its own inverse transpose



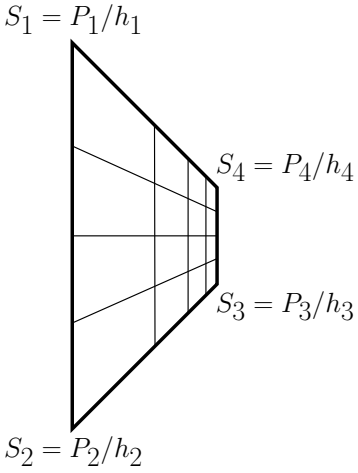


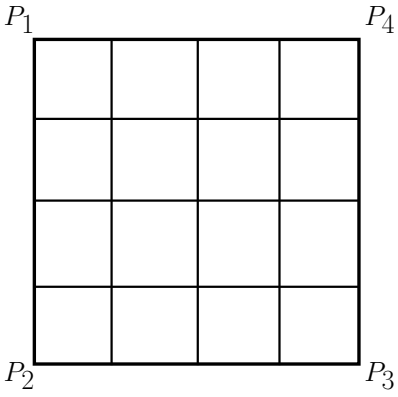

# Depth Distortion

- Recall depth distortion from perspective
  - Interpolating in screen space different than in world
  - Ok, for shading (mostly)
  - Bad for texture



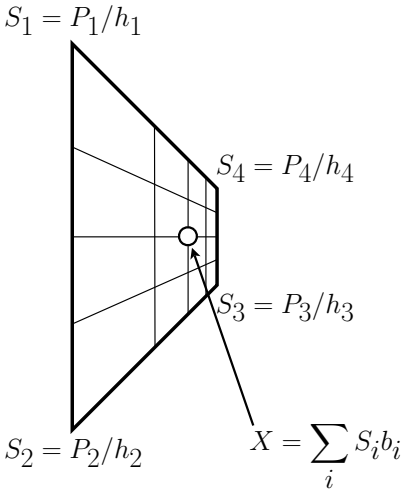
# Depth Distortion

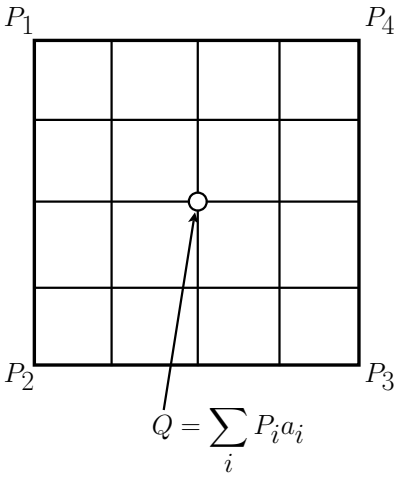




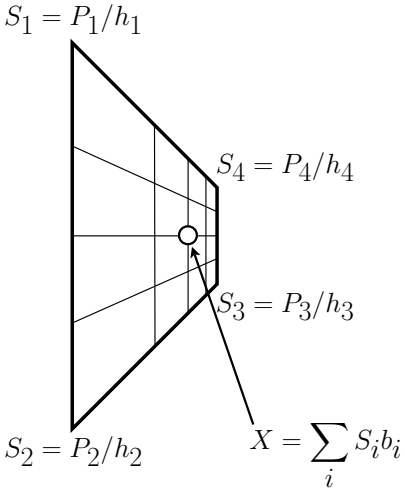

# Depth Distortion

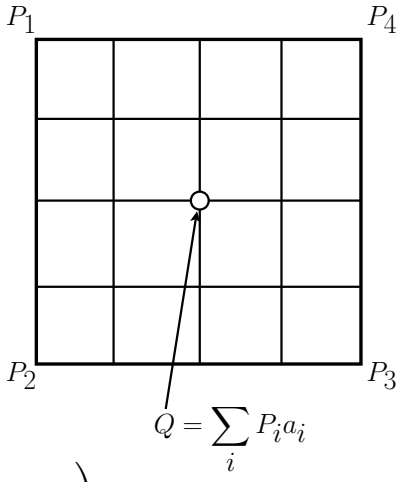



We know the $S_i$, $P_i$, and $b_i$, but not the $a_i$.

# Depth Distortion

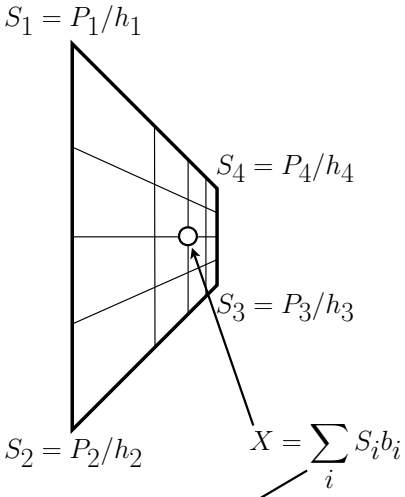


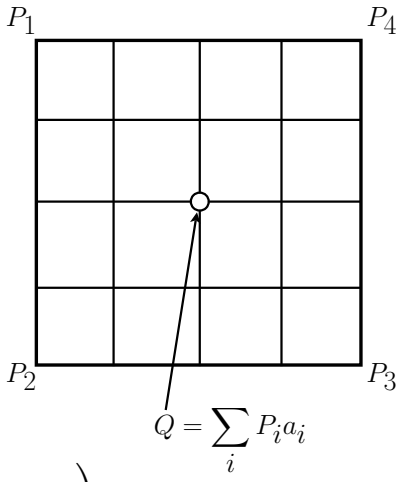




$$X = Q/h = \left(\sum_i P_i a_i\right) / \left(\sum_j h_j a_j\right)$$



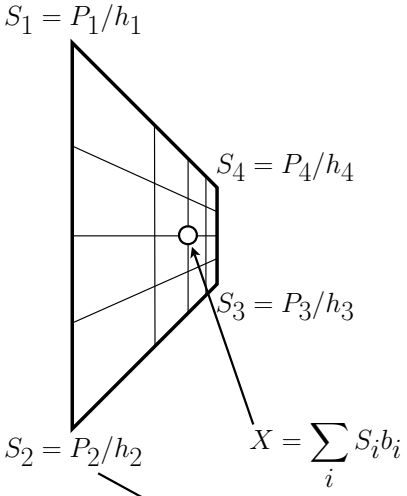
# Depth Distortion

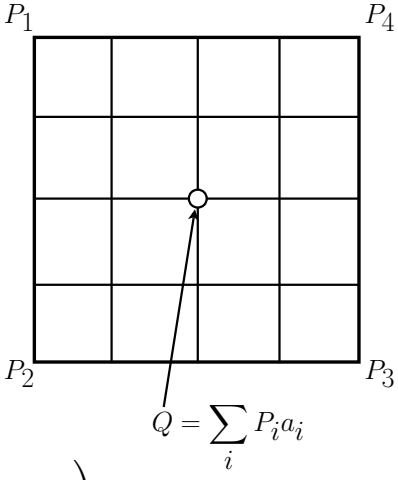



$$\sum_i S_i b_i = \left(\sum_i P_i a_i\right) / \left(\sum_j h_j a_j\right)$$

# Depth Distortion

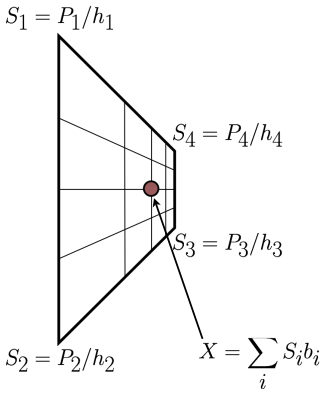


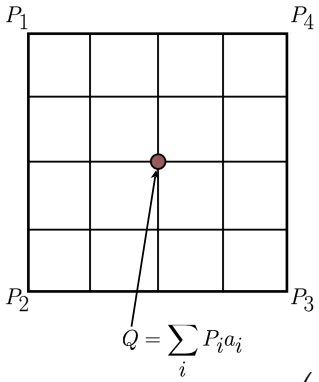




$$\sum_i P_i b_i / h_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right)$$



# Depth Distortion



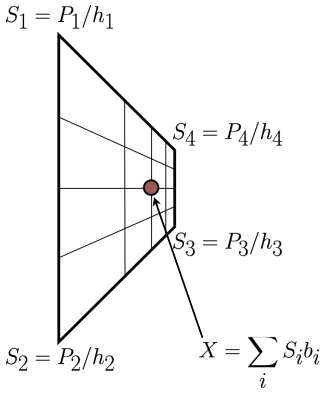



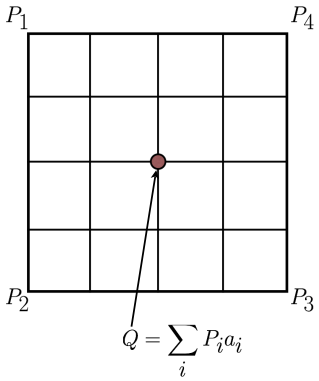

Independent of given vertex locations.

$$\sum_i P_i b_i / h_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right)$$

$$b_i / h_i = a_i / \left( \sum_j h_j a_j \right) \quad \forall i$$

# Depth Distortion



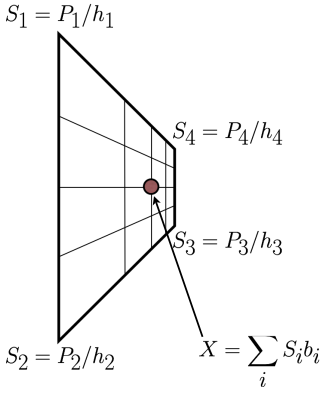

$$b_i/h_i = a_i/\left(\sum_j h_j a_j\right) \quad \forall i$$

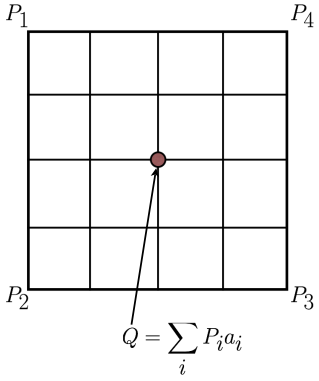
Linear equations in the $a_i$ .

$$\left(\sum_j h_j a_j\right) b_i/h_i - a_i = 0 \quad \forall i$$



# Depth Distortion



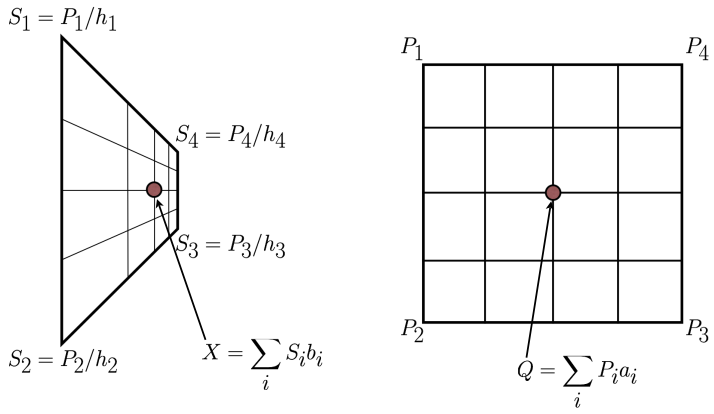

Linear equations in the $a_i$ .

$$\left(\sum_j h_j a_j\right) b_i/h_i - a_i = 0 \quad \forall i$$

Not invertible so add some extra constraints.

$$\sum_i a_i = \sum_i b_i = 1$$

# Depth Distortion




For a line: $a_1 = h_2 b_i / (b_1 h_2 + h_1 b_2)$

For a triangle: $a_1 = h_2 h_3 b_1 / (h_2 h_3 b_1 + h_1 h_3 b_2 + h_1 h_2 b_3)$

Obvious Permutations for other coefficients.