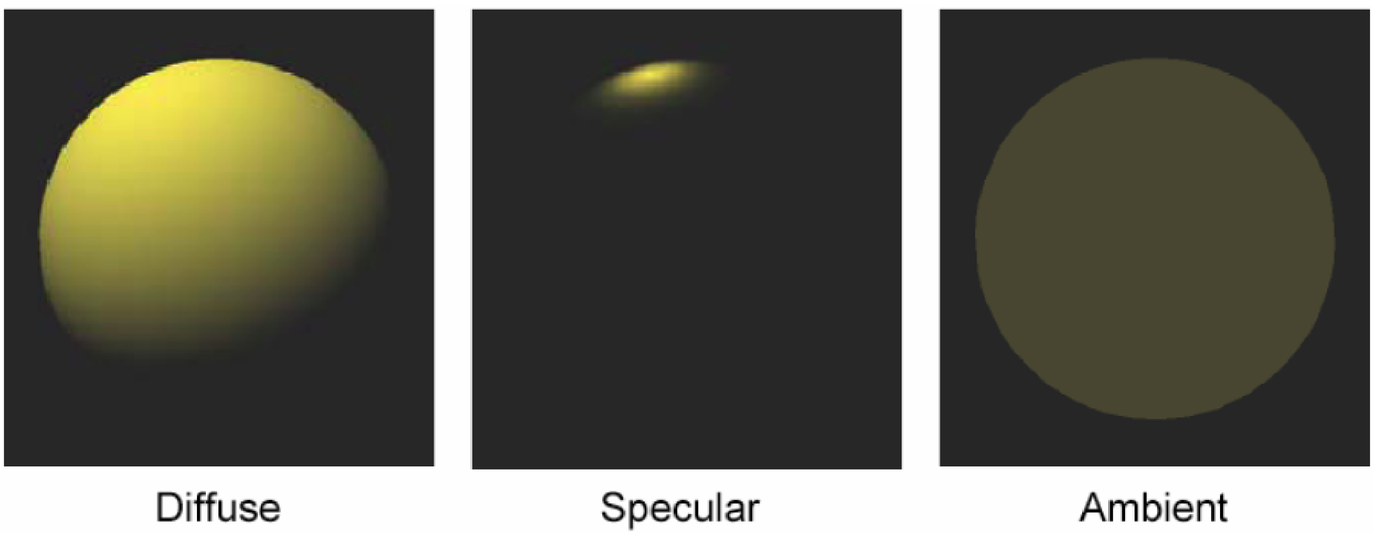
Announcements

- **Assignment 1**
  - Programming (**was due Friday**)
  - Last day to submit for late credit was yesterday
- **Assignment 2**
  - Theory **due next Wednesday**
  - Programming
- **Midterm**
  - **Today 6-7pm in AA204**
  - All you need is a writing utensil and an ID
    (no books/notes/calculators, etc.)
  - You cannot leave the exam room **till 7:15pm**

Last week …

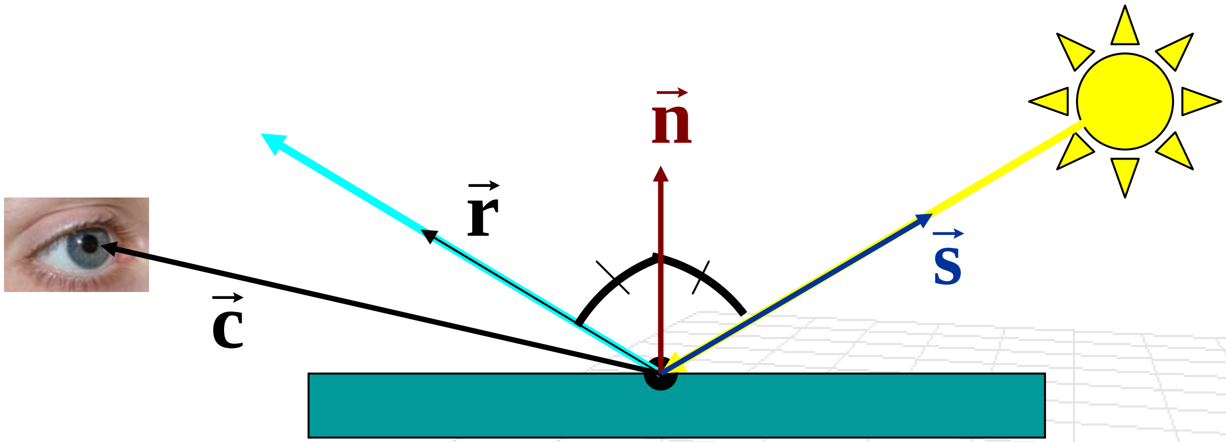
- We started taking about lighting
  - I introduced a **Phong model**



$$\mathbf{L}(\overline{\mathbf{p}}, \vec{\mathbf{c}}, \vec{\mathbf{s}}) = \mathbf{r}_d \mathbf{I}_d \max(0, \vec{\mathbf{s}} \cdot \vec{\mathbf{n}}) + \mathbf{r}_s \mathbf{I}_s \max(0, \vec{\mathbf{r}} \cdot \vec{\mathbf{c}})^{\alpha} + \mathbf{r}_a \mathbf{I}_a$$

Last week …

- We started taking about lighting
  - I introduced a **Phong model**



$$L(\overline{\mathbf{p}}, \vec{\mathbf{c}}, \vec{\mathbf{s}}) = \mathbf{r}_d \mathbf{I}_d \max(0, \vec{\mathbf{s}} \cdot \vec{\mathbf{n}}) + \mathbf{r}_s \mathbf{I}_s \max(0, \vec{\mathbf{r}} \cdot \vec{\mathbf{c}})^{\alpha} + \mathbf{r}_a \mathbf{I}_a$$

Shading

Computer Graphics, CSCD18

Fall 2008

Instructor: Leonid Sigal

Shading

- **Goal:** use light/reflectance model we derived last week to shade/color facets of polygonal mesh

- We know how to color a point on objects surface given a point, a normal at that point, a light source, and a camera position (**Phong model**)

- But, **geometry is** not modeled using points (too expensive), it is **modeled using polygonal meshes**.

- This is why we need **shading**

Basic setup

Assume we know

$\bar{\mathbf{e}}^{\mathbf{w}}$ - center of projection (position of eye) in world coordinates

$\bar{\mathbf{l}}^{\mathbf{w}}$ - position of the point light source in world coordinates

$\mathbf{I_a}, \mathbf{I_d}, \mathbf{I_s}$ - intensity of ambient, diffuse, and secular light sources

$\mathbf{r_a}, \mathbf{r_d}, \mathbf{r_s}$ - reflection coefficients of ambient, diffuse, and secular light sources

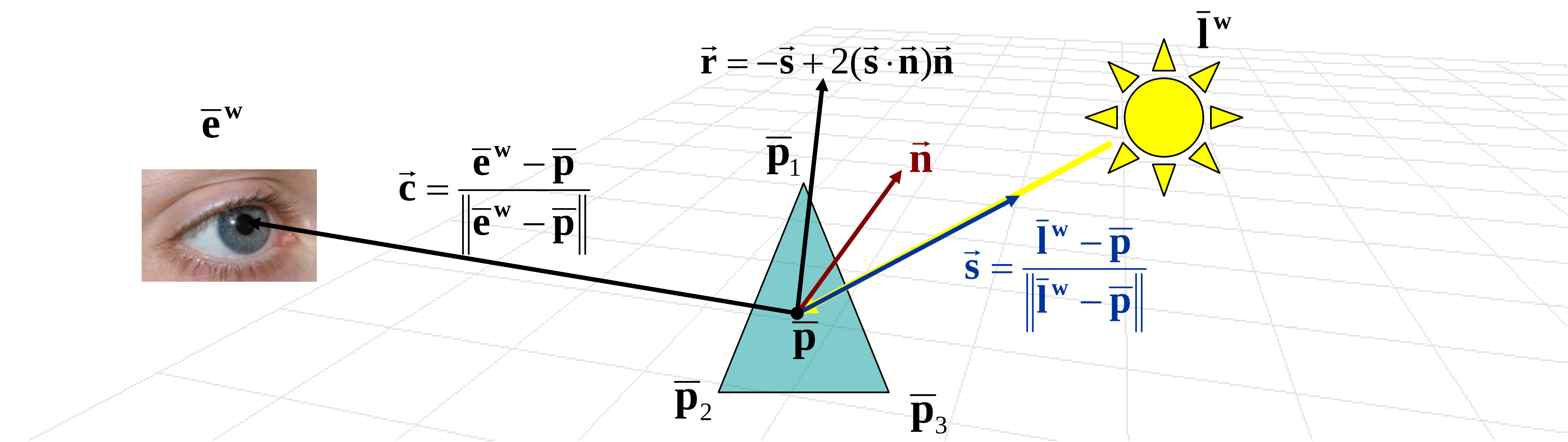
α - exponent controlling width of the specular highlight

How do we shade a triangle?

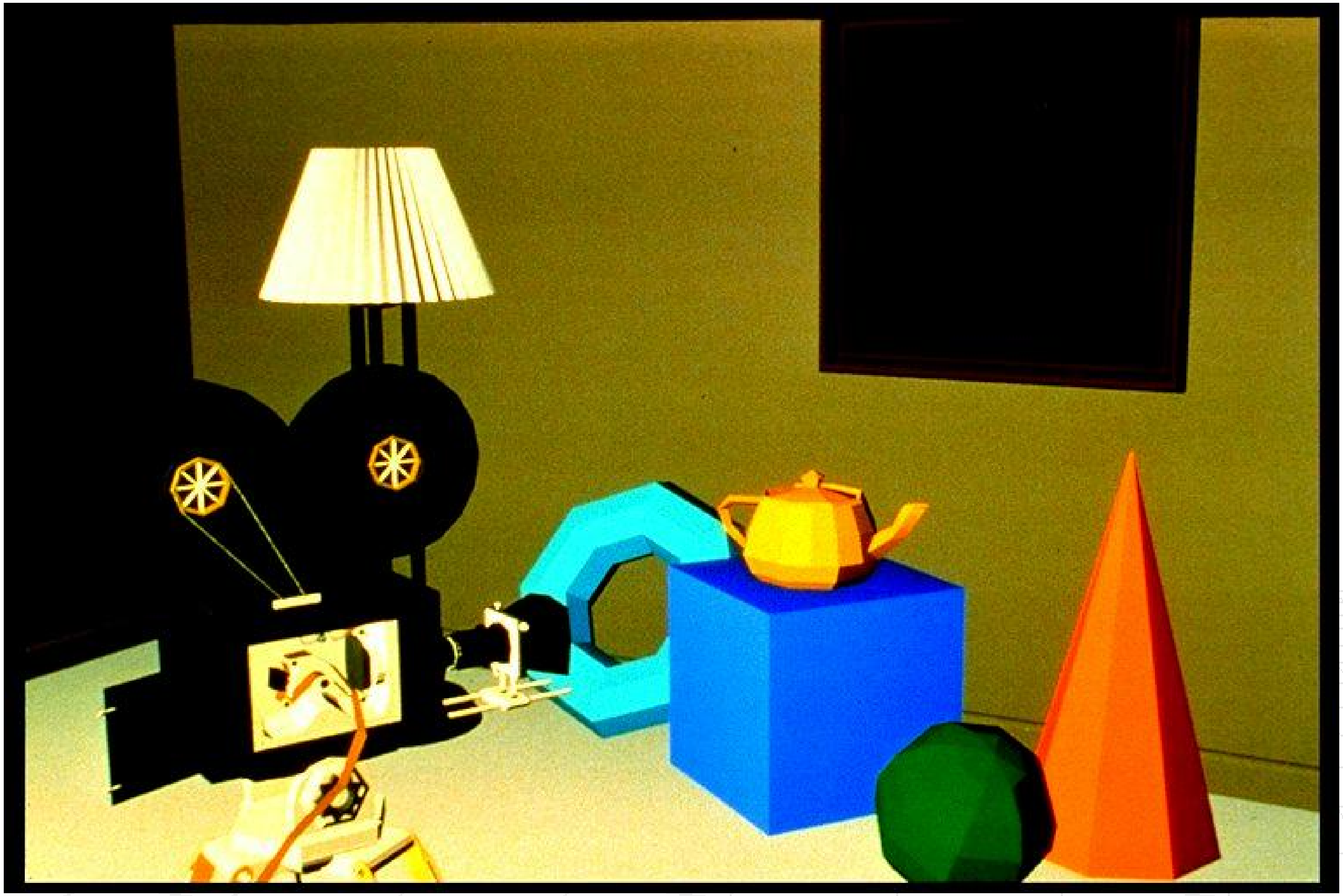
Flat Shading

- **Idea:** Fill each triangle with single color
- Let us assume we have a triangle with vertices $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \bar{\mathbf{p}}_3$ in CCW order
- We can then compute the normal, $\vec{\mathbf{n}} = \dfrac{(\bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_1) \times (\bar{\mathbf{p}}_3 - \bar{\mathbf{p}}_1)}{\left\| (\bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_1) \times (\bar{\mathbf{p}}_3 - \bar{\mathbf{p}}_1) \right\|}$
- And shade entire triangle using Phong model

$$\mathbf{L}(\bar{\mathbf{p}}, \bar{\mathbf{e}}^{\mathrm{w}}, \bar{\mathbf{l}}^{\mathrm{w}}) = \mathbf{r_d I_d} \max(0, \vec{\mathbf{s}} \cdot \vec{\mathbf{n}}) + \mathbf{r_s I_s} \max(0, \vec{\mathbf{r}} \cdot \vec{\mathbf{c}})^{\alpha} + \mathbf{r_a I_a}$$



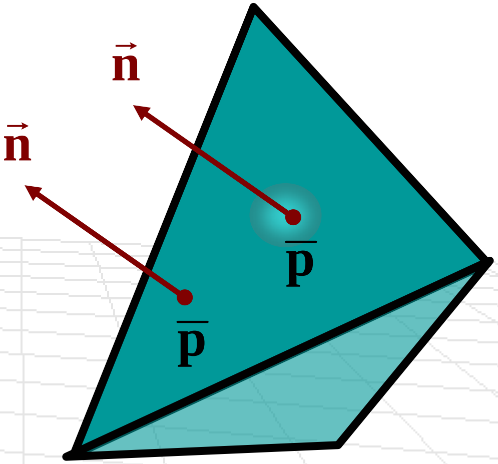
Flat Shading



Foley, van Dam, Feiner, Hughes, Plate II.29

Issues with Flat Shading

- For large faces secularities are impractical, since highlight is often sharp
  - Because of this, typically the secular term is dropped
- Mesh binderies are visible
  - People are very sensitive to this
- **Solutions**
  - Use small patches (but this is inefficient)
  - Use interpolative shading



Interpolative Shading

- **Idea:** Average intensities at vertices of the triangle to smoothly interpolate over pixels within a face

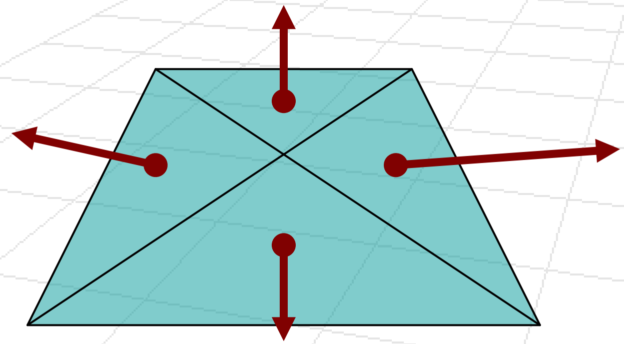
- **Algorithm**, for a triangular face with vertices $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \bar{\mathbf{p}}_3$
  - Compute normals at each vertex
  - Compute radiance $\mathbf{E}_j = \mathbf{L}(\bar{\mathbf{p}}_j, \bar{\mathbf{e}}^w, \bar{\mathbf{l}}^w)$ for each vertex point $\bar{\mathbf{p}}_j$

$$\mathbf{L}(\bar{\mathbf{p}}_j, \bar{\mathbf{e}}^w, \bar{\mathbf{l}}^w) = \mathbf{r}_d \mathbf{I}_d \max(0, \vec{\mathbf{s}}_j \cdot \vec{\mathbf{n}}_j) + \mathbf{r}_s \mathbf{I}_s \max(0, \vec{\mathbf{r}}_j \cdot \vec{\mathbf{c}}_j)^\alpha + \mathbf{r}_a \mathbf{I}_a$$

  - Project vertices onto image plane
  - Fill polygon by interpolating radiance along the triangle (scan conversion)

Interpolative Shading in Detail

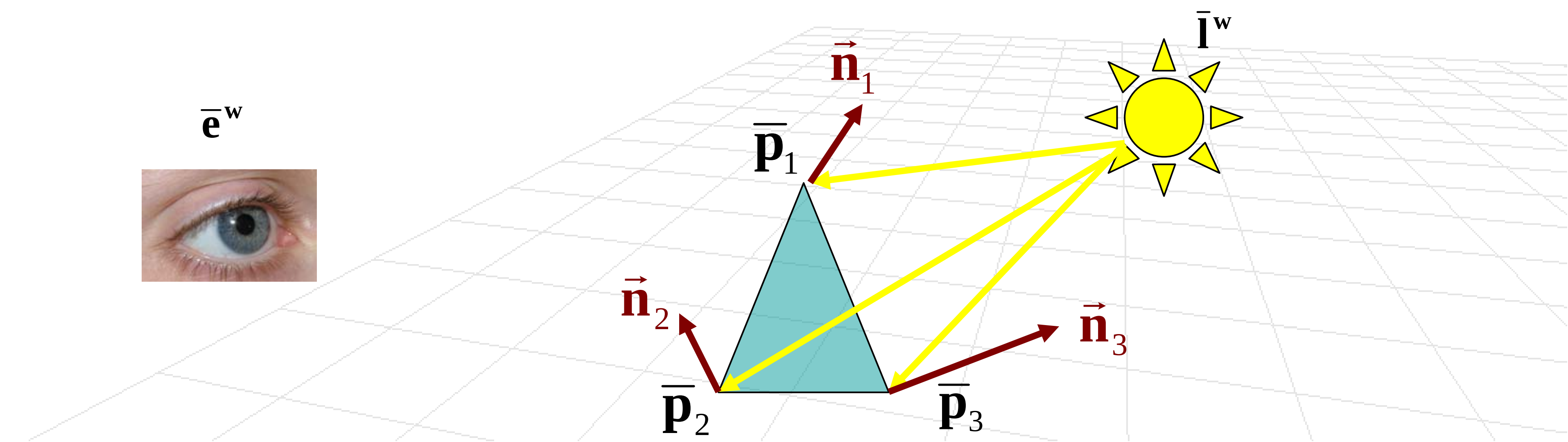
- Compute normals at each vertex
  - Many approaches are possible
  - Given parametric shape, compute normal $\vec{\mathbf{n}}_j$ when sampling vertices of the mesh $\overline{\mathbf{p}}_j$
    - Implicit form $\vec{\mathbf{n}}_j(\overline{\mathbf{p}}_j) = \nabla \mathbf{f}(\overline{\mathbf{p}}_j)$
    - Explicit form $\vec{\mathbf{n}}_j(\overline{\mathbf{p}}_j) = \left.\dfrac{\partial \mathbf{s}(\alpha,\beta)}{\partial \alpha}\right|_{\alpha_0,\beta_0} \times \left.\dfrac{\partial \mathbf{s}(\alpha,\beta)}{\partial \beta}\right|_{\alpha_0,\beta_0}$
  - Let $\vec{\mathbf{n}}_j$ be average of "**face normals**" of all adjacent faces



Interpolative Shading in Detail

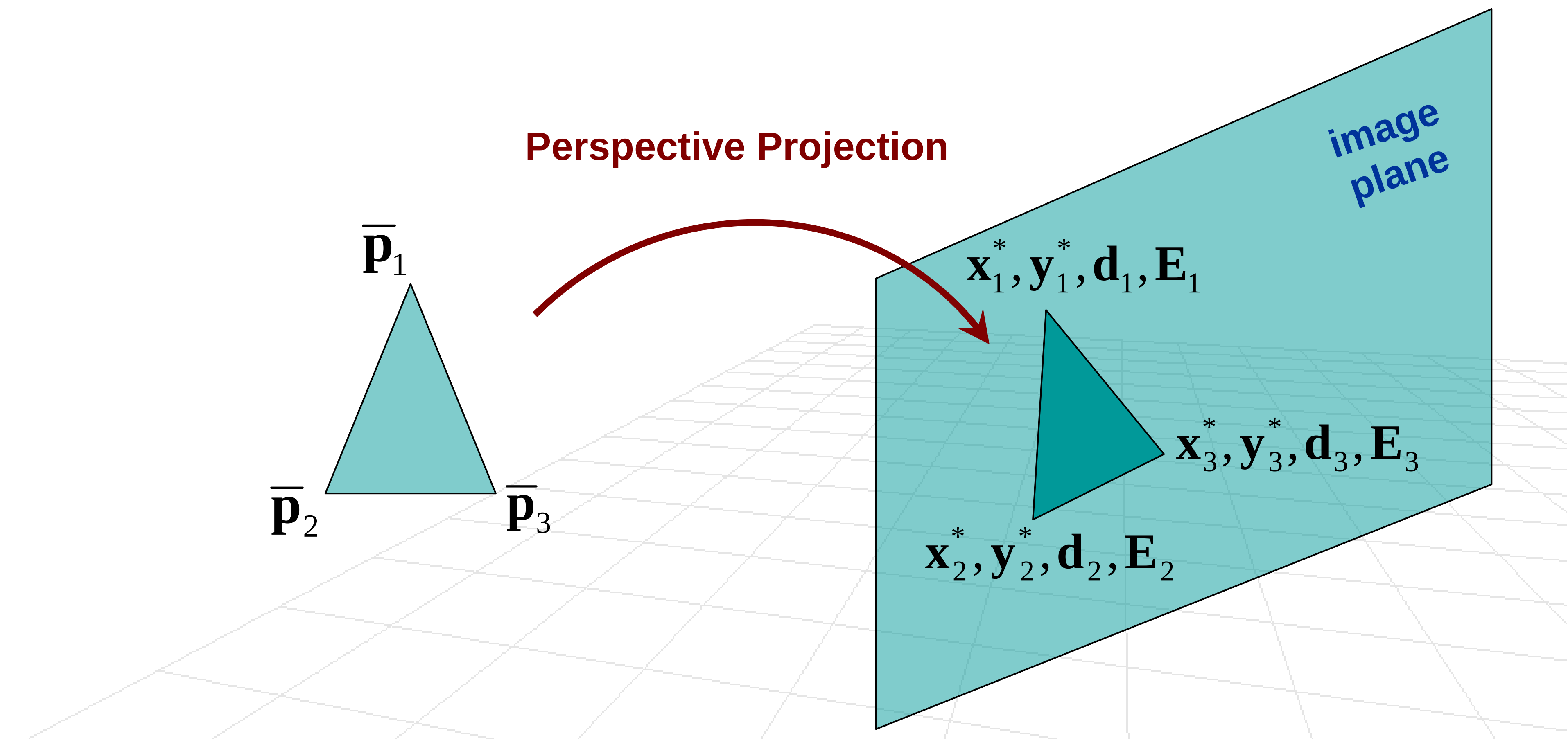
- Compute radiance $\mathbf{E_j}$ for each vertex point $\mathbf{p_j}$
  - Same as in flat shading (using Phong model)

$$\mathbf{E_j} = \mathbf{L}(\overline{\mathbf{p}}_\mathbf{j}, \overline{\mathbf{e}}^\mathrm{w}, \overline{\mathbf{l}}^\mathrm{w}) = \mathbf{r_d I_d} \max(0, \vec{\mathbf{s}}_\mathbf{j} \cdot \vec{\mathbf{n}}_\mathbf{j}) + \mathbf{r_s I_s} \max(0, \vec{\mathbf{r}}_\mathbf{j} \cdot \vec{\mathbf{c}}_\mathbf{j})^\alpha + \mathbf{r_a I_a}$$



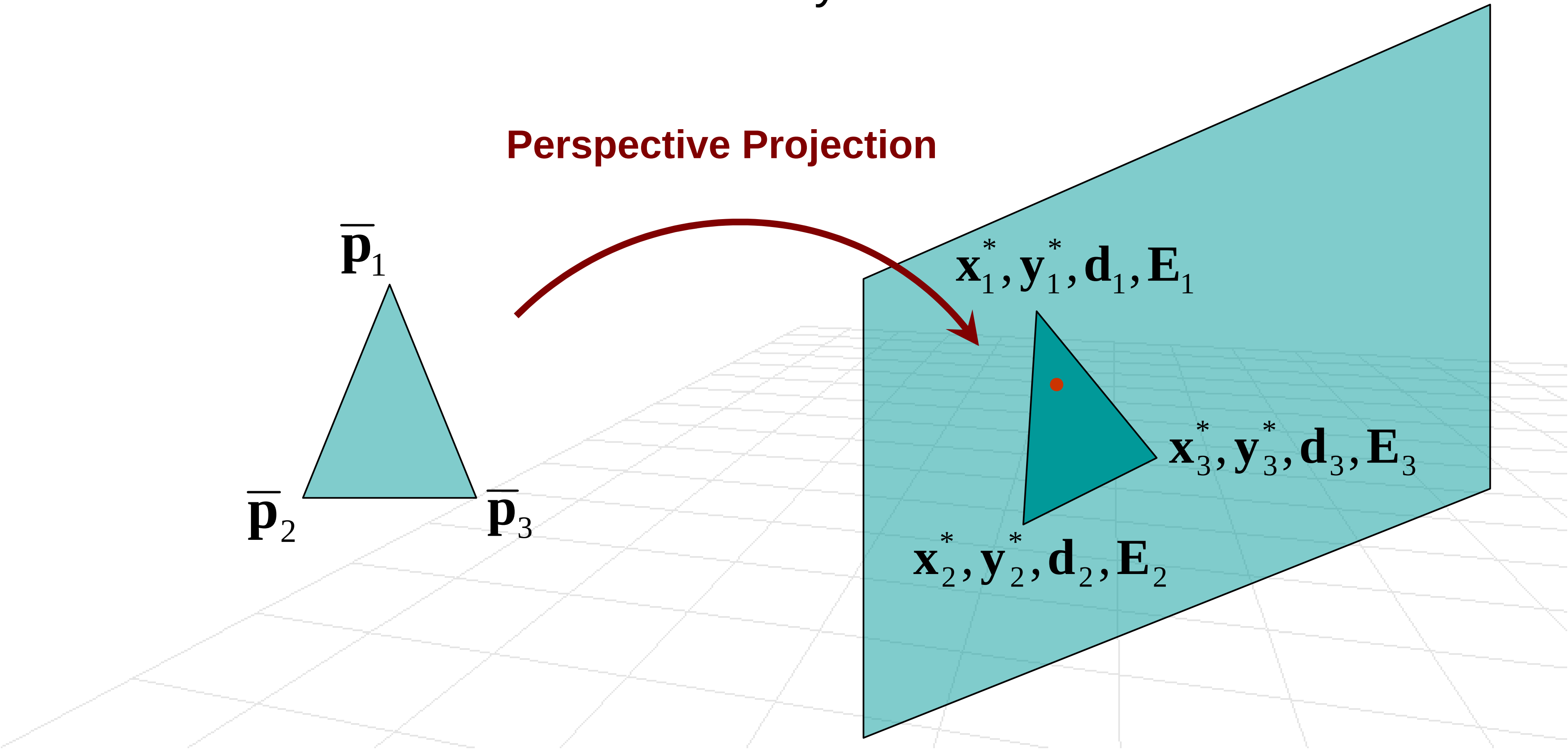
Interpolative Shading in Detail

- **Project onto image plane with pseudodepth**
  - So for each vertex we have $\mathbf{x}^*_j, \mathbf{y}^*_j, \mathbf{d}_j = \mathbf{z}^*_j, \mathbf{E}_j$



Interpolative Shading in Detail

- Scan conversion with linear interpolation of both pseudodepth ($\mathbf{d_j}$) and radiance values ($\mathbf{E_j}$)
  - use z-buffer to handle visibility



Algorithm (part 1)

- For each edge between $(\mathbf{x}_b^*, \mathbf{y}_b^*, \mathbf{d}_b, \mathbf{E}_b)$ and $(\mathbf{x}_a^*, \mathbf{y}_a^*, \mathbf{d}_a, \mathbf{E}_a)$ ordered such that $\mathbf{y}_a^* > \mathbf{y}_b^*$

$$\mathbf{x} = \mathbf{x}_b^*, \quad \Delta\mathbf{x} = (\mathbf{x}_a^* - \mathbf{x}_b^*)/(\mathbf{y}_a^* - \mathbf{y}_b^*)$$

$$\mathbf{d} = \mathbf{d}_b, \quad \Delta\mathbf{d} = (\mathbf{d}_a - \mathbf{d}_b)/(\mathbf{y}_a^* - \mathbf{y}_b^*)$$

$$\mathbf{E} = \mathbf{E}_b, \quad \Delta\mathbf{E} = (\mathbf{E}_a - \mathbf{E}_b)/(\mathbf{y}_a^* - \mathbf{y}_b^*)$$

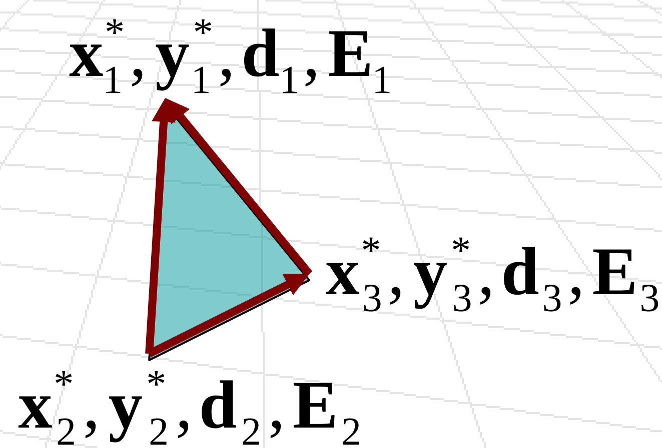
  - For ($\mathbf{y} = \mathbf{y}_b$; $\mathbf{y} < \mathbf{y}_a$; $\mathbf{y}$++)

    Place ($\mathbf{x}$, $\mathbf{d}$, $\mathbf{E}$) in active edge list (AEL) at scanline $\mathbf{y}$

$$\mathbf{x} = \mathbf{x} + \Delta\mathbf{x}$$

$$\mathbf{d} = \mathbf{d} + \Delta\mathbf{d}$$

$$\mathbf{E} = \mathbf{E} + \Delta\mathbf{E}$$



Algorithm (part 2)

- For each scanline between $\min(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3)$ and $\max(\mathbf{y}_1 . \mathbf{y}_2, \mathbf{y}_3)$
  - Extract ($\mathbf{x}_\mathbf{a}$, $\mathbf{d}_\mathbf{a}$, $\mathbf{E}_\mathbf{a}$) and ($\mathbf{x}_\mathbf{a}$, $\mathbf{d}_\mathbf{a}$, $\mathbf{E}_\mathbf{a}$) from AEL where $\mathbf{x}_\mathbf{a} > \mathbf{x}_\mathbf{b}$

$$\mathbf{d} = \mathbf{d}_\mathbf{b}, \quad \Delta\mathbf{d} = (\mathbf{d}_\mathbf{a} - \mathbf{d}_\mathbf{b})/(\mathbf{x}_\mathbf{a} - \mathbf{x}_\mathbf{b})$$

$$\mathbf{E} = \mathbf{E}_\mathbf{b}, \quad \Delta\mathbf{E} = (\mathbf{E}_\mathbf{a} - \mathbf{E}_\mathbf{b})/(\mathbf{x}_\mathbf{a} - \mathbf{x}_\mathbf{b})$$

  - For ($\mathbf{x} = \mathbf{x}_\mathbf{b}$; $\mathbf{x} < \mathbf{x}_\mathbf{a}$; $\mathbf{x}$++)

    if ($\mathbf{d}$ < z-buffer($\mathbf{x}$, $\mathbf{y}$))

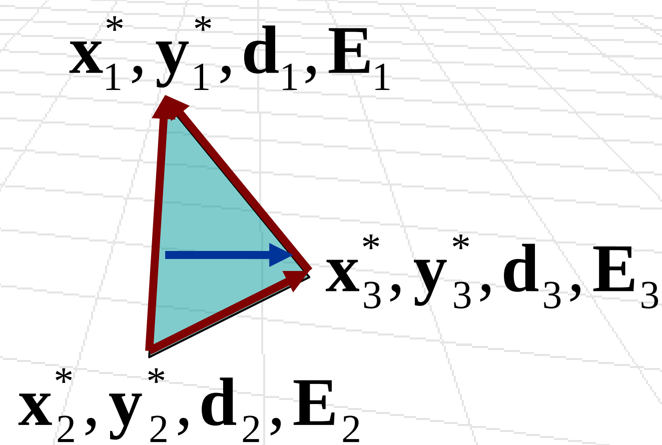
        putpixel($\mathbf{x}$, $\mathbf{y}$, $\mathbf{E}$)

        z-buffer($\mathbf{x}$, $\mathbf{y}$) = $\mathbf{d}$

    end

$$\mathbf{d} = \mathbf{d} + \Delta\mathbf{d}$$

$$\mathbf{E} = \mathbf{E} + \Delta\mathbf{E}$$



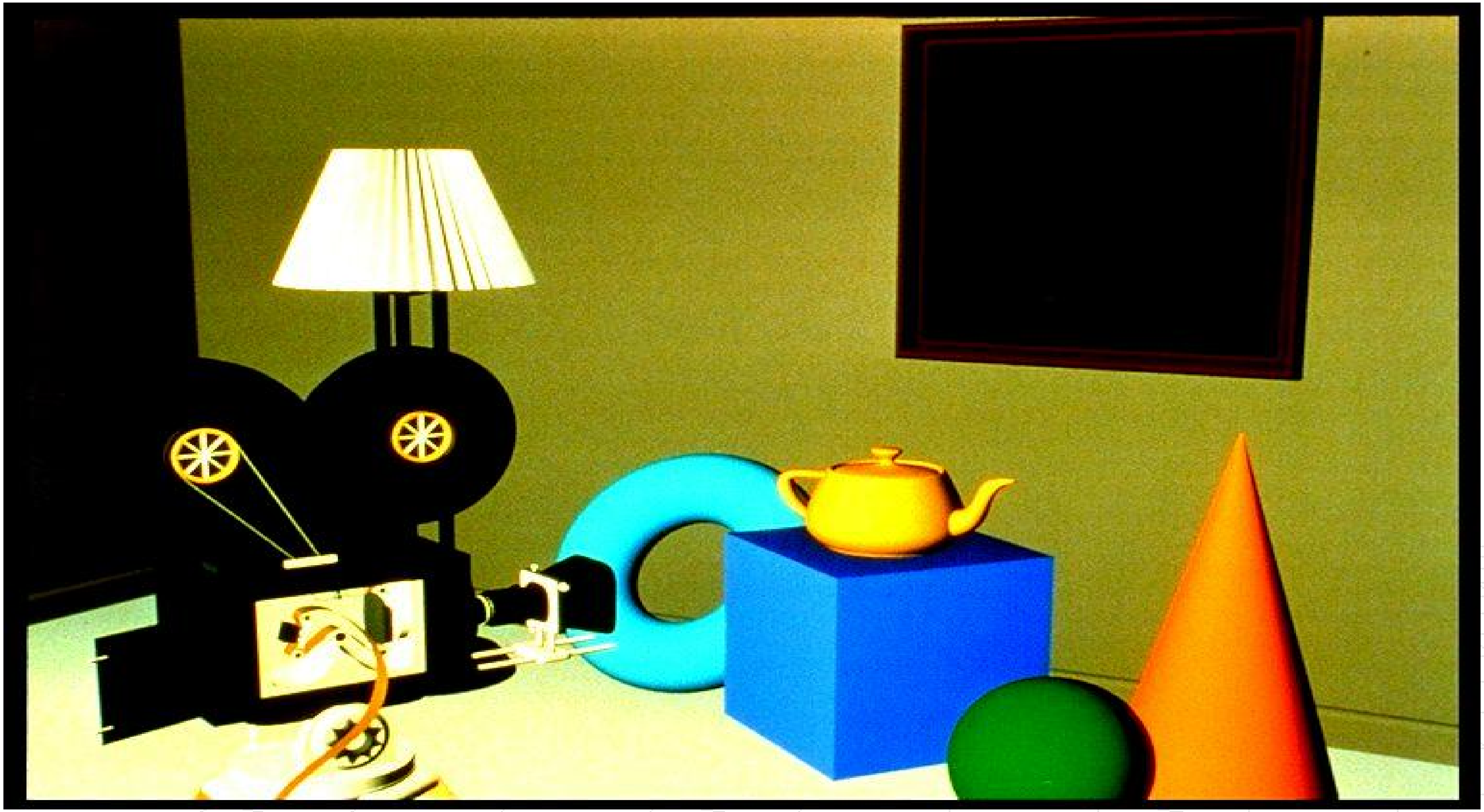
Interpolative Shading in Detail

- What we just described is so called **Gouraud Shading**

- **Advantages**
  - Does not produce artifacts at face boundaries (i.e. better then flat shading)

- **Disadvantages**
  - Still hard to handle secular highlights. **Why?**

Gouraud Shading



Foley, van Dam, Feiner, Hughes, Plate II.30

Phong Shading

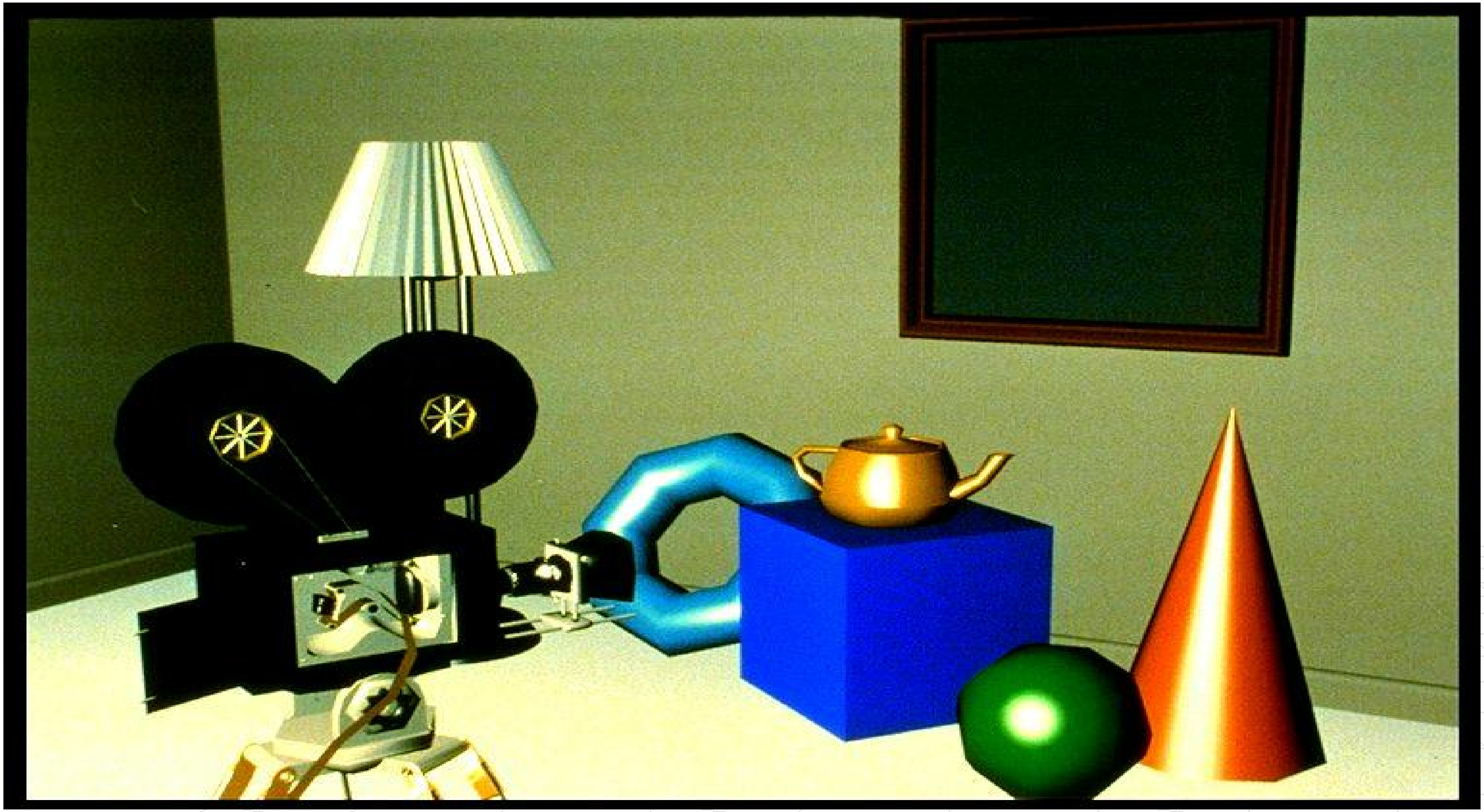
- **Idea:** Slightly modify the Gouraud shading algorithm to correctly shade every pixel (with secularities)

  (Note that **phong shading** and **phong lighting** are not one and the same)

- **Algorithm**, for a triangular face with vertices $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$
  - Compute normals at each vertex
  - For each point on a triangle that corresponds to a pixel location interpolate the normal
  - Compute radiance $\mathbf{E}_j$ for each pixel in the projected triangle that corresponds to point within the world triangle
  - Project vertices onto image plane

- **Why is this batter then just doing Phong lighting?**

Phong Shading



Foley, van Dam, Feiner, Hughes, Plate II.32

Phong Shading

- Advantages
  - Produces very accurate shading with specular highlights (better then flat shading and Gouraud shading)
- Disadvantages
  - It's computationally expensive (but not on current graphics hardware)