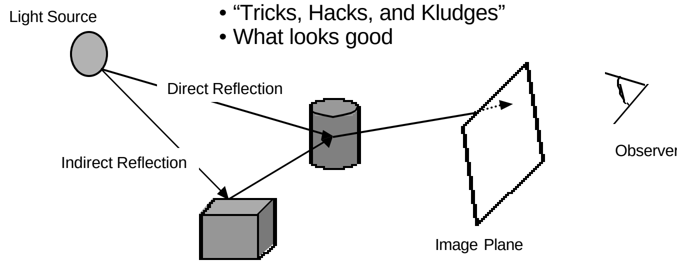
Illumination / Reflection Models

Lecture 9

- Rigorous Physics

$$I(x,y,z) = \int_{t=-\infty}^{+\infty}\int_{l=580}^{760}\int_{f=0}^{p/2}\int_{q=0}^{2p} L(t,x,y,z,f,q,l)R(t,x,y,z,f,q,l)\,dq\,df\,dl\,dt$$

- Simplified Physics
  - "Tricks, Hacks, and Kludges"
  - What looks good



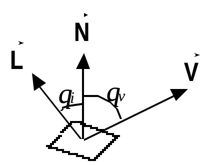
- Fidelity vs Speed Tradeoff
  - Application specific

1

Illumination / Reflection Models

Lecture 9

- Model Components
  - Number of light sources
  - Type of lights
  - Location of lights
  - Orientation of object
  - Object material/surface properties
  - Visibility
  - Location of observer



$\vec{N}$ = Surface Normal
$\vec{L}$ = Light Vector
$\vec{V}$ = Viewing Vector

q_v = Viewing Angle
q_i = IncidenceAngle

2

Illumination / Reflection Models

Lecture 9

Spheres Rendered with No Lights

Ambient Reflection

Lecture 9

- Ambient Reflection Term
  - Self-luminance?
  - Multiple reflections from multiple objects
  - Constant over entire surface

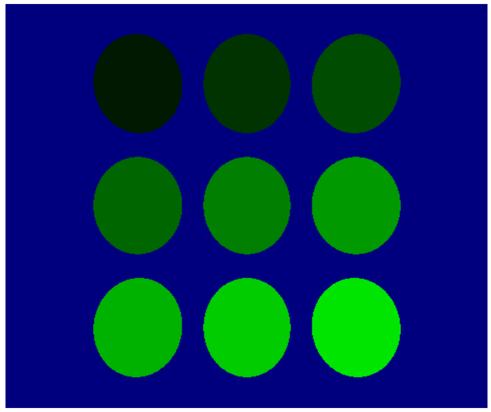
$$I = k_a I_a \quad \text{where} \quad \begin{aligned} I &= \text{Reflected intensity} \\ k_a &= \text{Ambient reflection coefficient} \\ I_a &= \text{Ambient light intensity} \end{aligned}$$

  - k_a is determined from material properties (no physical basis)
  - Ambient term is independent of :
    - Light vector
    - Surface normal
    - Viewing vector

Ambient Reflection

Lecture 9

$$I = k_a I_a$$



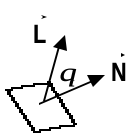
Spheres Rendered with Ambient Reflection Only

5

Diffuse Reflection

Lecture 9

- Diffuse Reflection Term
  - Reflects light equally in ALL directions
  - Caused by rough surface
  - Same intensity regardless of viewing direction
  - Lambertian surface



$$I_d = k_d I_l \cos(\theta_i)$$

$$I_d = k_d I_l (\hat{n} \cdot \hat{l})$$

I_d = Diffuse intensity

k_d = Diffuse reflection coefficient

I_l = Light source intensity

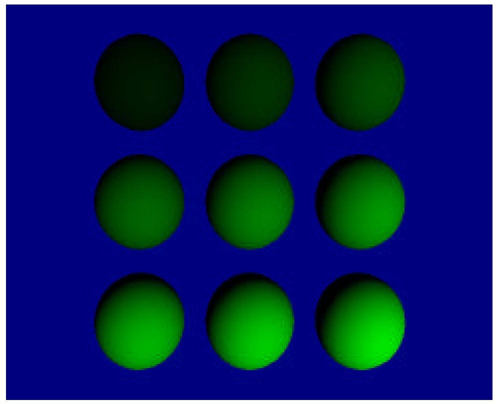
$$(\hat{n} \cdot \hat{l}) < 0, \quad \Rightarrow \quad \cos\theta_i < 0, \quad \Rightarrow \theta_i > 90^{\circ}$$ Light source is behind surface

6

Diffuse Reflection

Lecture 9

$$I_d = k_d \, I_l \cos(q_i)$$



Spheres Rendered with Diffuse Reflection Only

7

Specular Reflection

Lecture 9

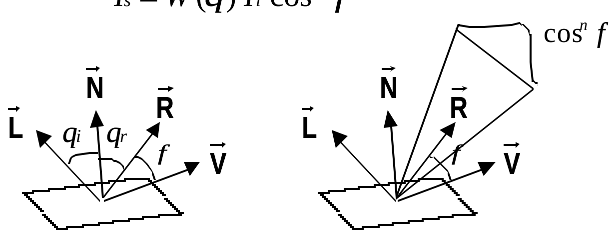
- Specular Reflection Term
  - Highlights
  - Reflection angle = Incidence angle
  - Perfect specular reflection :

$$I_s = 0 \text{ everywhere except at } \vec{V} = \vec{R}$$

  - Phong reflection model

$$I_s = W(q) \, I_l \cos^n f$$

$q_i = q_r$

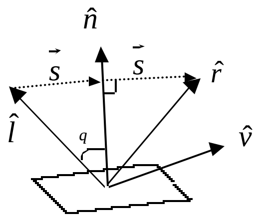


8

Specular Reflection

Lecture 9

- Specular Reflection Term
  - As *n* increases, the size of the highlight decreases
    - Dull surfaces have small *n*
    - Polished surfaces large *n*
  - W(θ) described by Fresnel Reflection Coefficients
    - Transparent materials have High W(θ) at large θ
    - Opaque materials have relatively constant W(θ)



$$I_s = W(\theta)\, I_l \cos^n \phi = k_s\, I_l\, (\hat{r}\cdot\hat{v})^n$$

$$\hat{l} + \vec{s} = \hat{n}\cos\theta$$
$$\vec{s} = \hat{n}\cos\theta - \hat{l}$$

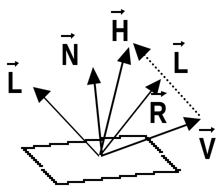
$$\vec{R} = \hat{n}\cos\theta + \vec{s} = \hat{n}\,2\cos\theta - \hat{l} = 2(\hat{n}\cdot\hat{l})\hat{n} - \hat{l}$$

$$\hat{r} = \frac{\vec{R}}{|\vec{R}|}$$

Specular Reflection

Lecture 9

- Specular Reflection Term
  - Some formulations use the “Halfway” vector
    - Orientation of surface for maximum specular reflection in the viewing direction



$$I_s = k_s\, I_l\, (\vec{N}\cdot\vec{H})^n$$

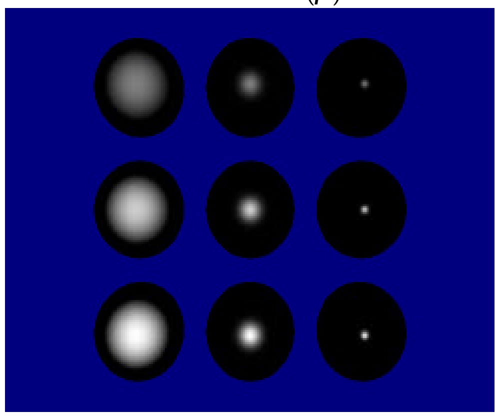
$$\vec{H} = \vec{L} + \vec{V}$$

If $\vec{L}$ and $\vec{V}$ are constant over the surface, then $\vec{H}$ is constant.
So, $(\vec{N}\cdot\vec{H})$ requires fewer calculations than $(\vec{R}\cdot\vec{V})$ since $\vec{R}$ must be re-calculated for every value of $\vec{N}$ across the surface

Specular Reflection

Lecture 9

$$I_s = k_s I_l \cos^n(f)$$



Spheres Rendered with Diffuse Reflection Only

11

Simple Reflection Model

Lecture 9

$$I = Ia + Id + Is$$
$$= k_a I_a + k_d I_l (\vec{\mathbf{N}} \cdot \vec{\mathbf{L}}) + k_s I_l (\vec{\mathbf{R}} \cdot \vec{\mathbf{V}})^n$$

- Reflection Coefficients
  - Based on:
    - Material type
    - Surface finish
    - Texture maps
  - What looks good
    - Artistic license
    - Trial and error
    - Personal library

12

Attenuation Model

Lecture 9

- Intensity Attenuation
  - Distinguish overlapping surfaces with the same reflection parameters
  - Radiant energy disperses as $1/r^2$
  - Attenuation function f(d)
    - User defined constants for a_0, a_1, a_2
    - *d* is the distance from the light source to the reflection point

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

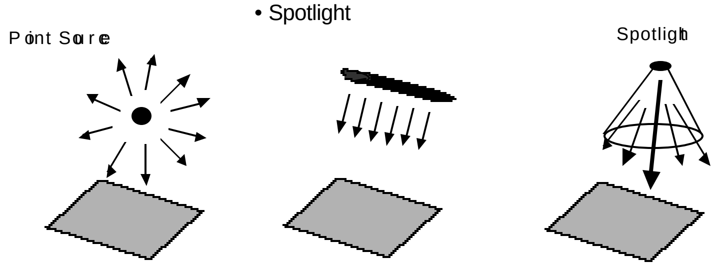
$$I = k_a I_a + f(d)\left\{k_d I_l (\vec{\mathbf{N}} \cdot \vec{\mathbf{L}}) + k_s I_l \left(\vec{\mathbf{R}} \cdot \vec{\mathbf{V}}\right)^n\right\}$$

13

Light Sources

Lecture 9

- Types of Light Sources
  - Point source
  - Directional source
  - Spotlight



- Sum Intensities Over All Sources

$$I = k_a I_a + \sum_{i=1}^{n} f(d_i)\left\{k_d I_{l_i} (\vec{\mathbf{N}} \cdot \vec{\mathbf{L}}_i) + k_s I_{l_i} \left(\vec{\mathbf{R}}_i \cdot \vec{\mathbf{V}}\right)^n\right\}$$

14

Write Reflection Model in RGB components

- Treat reflection coefficients as vectors

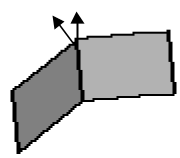
$$k_a = \langle k_{aR}, k_{aG}, k_{aB} \rangle$$
$$k_d = \langle k_{dR}, k_{dG}, k_{dB} \rangle$$
$$k_s = \langle k_{sR}, k_{sG}, k_{sB} \rangle$$

- A "green" object has nonzero green coeffs

$$I_R = k_{aR}\, I_{aR} + \sum_{i=1}^{n} f(d_i) \left\{ k_{dR}\, I_{l_i} (\vec{\mathbf{N}} \cdot \vec{\mathbf{L}}_i) + k_{sR}\, I_{l_i} \left(\vec{\mathbf{R}}_i \cdot \vec{\mathbf{V}} \right)^n \right\}$$

- Modify surface color by changing reflection coeffs
  - Plastic surfaces have white highlights
  - Metallic surfaces have colored highlights

- Entire Polygon Displayed with the Same Intensity
  - Calculate intensity from the reflection model
  - Use the facet normal (for 3 vertex polygons)
  - Compute an average normal (for >3 vertex polygons)



- Valid Shading Model when
  - Object is truly planar (not an approximation of a curved surface)
  - All light sources are far enough away that $(\vec{\mathbf{N}} \cdot \vec{\mathbf{L}}_i)$ is constant over the polygon surface
  - Viewing position is far enough away that $(\vec{\mathbf{R}}_i \cdot \vec{\mathbf{V}})$

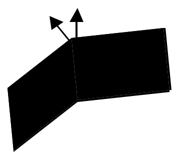
Gouraud Shading

Lecture 9

- Linear Interpolation of Intensity Across Polygon Surface

Match Intensity Across Polygon Edges

- Gouraud Algorithm
  - Determine average normal at each vertex



$$\vec{N}_{avg} = \frac{\sum_{i=1}^{n} \vec{N}_i}{\left|\sum_{i=1}^{n} \vec{N}_i\right|} \qquad n = \text{all polygons that share the vertex}$$

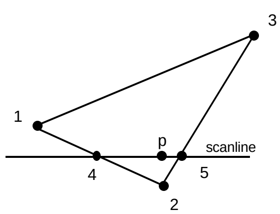
  - Compute intensity at each vertex from reflection model and average normal
  - Linearly interpolate intensity across surface of polygon

17

Gouraud Shading

Lecture 9

- Linear Interpolation of Intensity Across Polygon Surface



- Compute average normal at each vertex (1, 2, 3)
- Compute intensities I_1, I_2, I_3
- Compute intensities I_4, I_5

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

- Compute intensity I_p

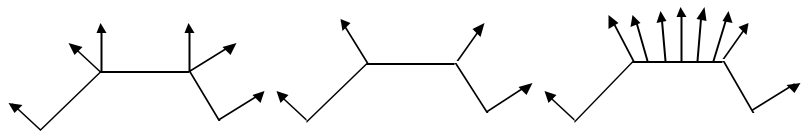
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

18

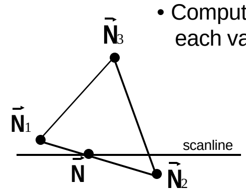
Phong Shading

Lecture 9

- Linear Interpolation of Normal Across Polygon Surface



- Phong Algorithm
  - Determine average normal at each vertex
  - Linearly interpolate normal across surface
  - Compute intensity from reflection model for each value of the normal



$$\vec{N} = \frac{y - y_2}{y_1 - y_2}\vec{N}_1 + \frac{y_1 - y}{y_1 - y_2}\vec{N}_2$$

Shading

Lecture 9

Flat Shaded

Phong Shaded

ERROR!

Rendering A Lit Sphere

```
/*  Initialize material property, light source, lighting model,  and depth buffer. */
void init(void)
{
   GLfloat mat_           [] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
           light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}
```

OpenGL Programming

Lecture 9

- Light Sources
  - glEnable(GL_LIGHTING)

  - glLightfv(lightname, param, value)
    - Parameters
      - GL_AMBIENT
      - GL_DIFFUSE
      - GL_SPECULAR
      - GL_POSITION

      - GL_SPOT_DIRECTION
      - GL_SPOT_EXPONENT
      - GL_SPOT_CUTOFF

      - GL_CONSTANT_ATTENUATION
      - GL_LINEAR_ATTENUATION
      - GL_QUADRATIC_ATTENUATION

  - glEnable(GL_LIGHT0)

OpenGL Programming

Lecture 9

- Shading Models

  - glShadeModel(mode)
    - Mode
      - GL_FLAT
      - GL_SMOOTH (Gouraud)

OpenGL Programming

- Material Properties
  - glMaterial(face, param, value)
    - Face
      - GL_FRONT
      - GL_BACK
      - GL_FRONT_AND_BACK
    - Parameters
      - GL_AMBIENT
      - GL_DIFFUSE
      - GL_AMBIENT_AND_DIFFUSE
      - GL_SPECULAR
      - GL_SHININESS
      - GL_EMISSION

See Chapter 5 for other techniques to minimize performance costs associated with changing material properties

OpenGL Programming

$$\text{Vertex Color} = (\text{Emission})_{\text{material}} + (\text{Ambient})_{\text{light}} * (\text{Ambient})_{\text{material}} + \sum_{i=1}^{num\_lights} \left\{ \left(\frac{1}{k_c + k_l d + k_q d^2} \right)_i * (\text{spotlight\_effect}) * \left[(\text{Ambient})_{\text{light}} * (\text{Ambient})_{\text{material}} + (\hat{\mathbf{L}} \bullet \hat{\mathbf{N}}) * (\text{Diffuse})_{\text{light}} * (\text{Diffuse})_{\text{material}} + (\hat{\mathbf{H}} \bullet \hat{\mathbf{N}})^{\text{shininess}} * (\text{Specular})_{\text{light}} * (\text{Specular})_{\text{material}} \right]_i \right\}$$

OpenGL Code

Lecture 9

```
GLfloat diffuseMaterial[4] = { 0.5, 0.5, 0.5, 1.0 };

/*  Initialize values for material property, light source,
 *  lighting model, and depth buffer. */
void myinit(void)
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseMaterial);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 25.0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glDepthFunc(GL_LESS);
    glEnable(GL_DEPTH_TEST);

    glColorMaterial(GL_FRONT, GL_DIFFUSE);
    glEnable(GL_COLOR_MATERIAL);
}
```

27

OpenGL Code

Lecture 9

```
void changeRedDiffuse (AUX_EVENTREC *event)
{
    diffuseMaterial[0] += 0.1;
    if (diffuseMaterial[0] > 1.0)
        diffuseMaterial[0] = 0.0;
    glColor4fv(diffuseMaterial);
}
void changeGreenDiffuse (AUX_EVENTREC *event)
{
    diffuseMaterial[1] += 0.1;
    if (diffuseMaterial[1] > 1.0)
        diffuseMaterial[1] = 0.0;
    glColor4fv(diffuseMaterial);
}
void changeBlueDiffuse (AUX_EVENTREC *event)
{
    diffuseMaterial[2] += 0.1;
    if (diffuseMaterial[2] > 1.0)
        diffuseMaterial[2] = 0.0;
    glColor4fv(diffuseMaterial);
}
```

28