Lighting and Shading

Hill – 8.2 (Upto 8.2.6), 8.3, 8.5 (Upto 8.5.3 excluding 8.5.2), 8.6

Basic Terms

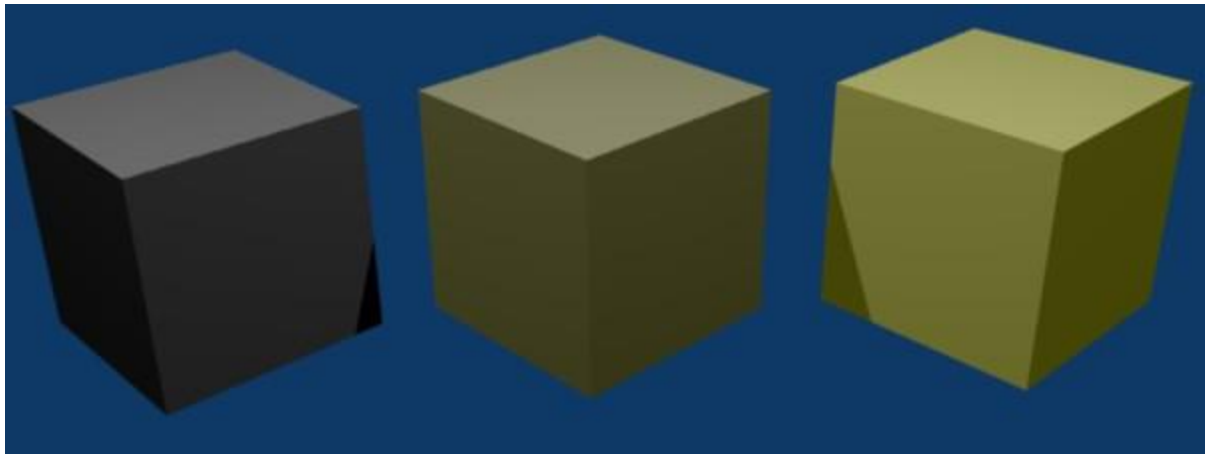
- Illumination: the transport of energy from light sources to surfaces & points
  - Local illumination
  - Global illumination
- Lighting model or Illumination model: Express the factors determining a surface's color or luminous intensity (outgoing or reflected light) at a particular 3D point

Components of Illumination

- Two components of illumination:
  1. Light sources
  2. Surface properties
- Light source described by a luminance/intensity ‘I’
  - Each color is described separately
  - I = [Ir Ig Ib]
- Types of Light Sources:
  1. Ambient Light
  2. Diffuse Light
  3. Spot Light

Ambient Light

- No identifiable source or direction
- Product of multiple reflections of light from the many surfaces present in the environment
- Computationally inexpensive



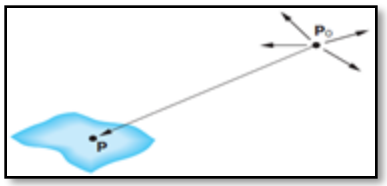
Ambient Light

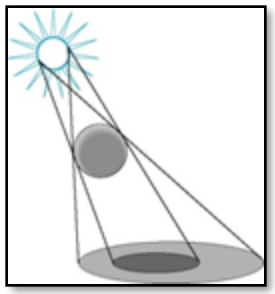
Categories:

1. Global ambient light
   - Independent of light source
   - Lights entire scene
   - Example: reflection of sunlight from several surfaces
2. Local ambient light
   - Contributed by additional light sources
   - Can be different for each light and primary color
   - Example: Reflection of fluorescent lamps from several surfaces

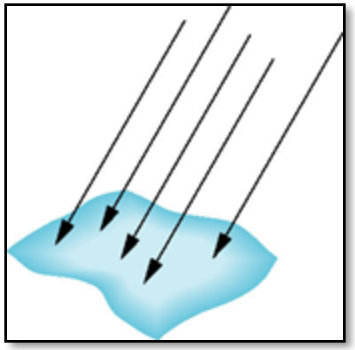
Diffuse Light

- **Point Source**
  - Given by a point
  - Light emitted equally in all directions
  - Intensity decreases with square of distance
  - Point source $[x\ y\ z\ 1]^T$
  - At point p, intensity received : $i(p, p_0) = \frac{1}{|p - p_0|^2} I(p_0).$

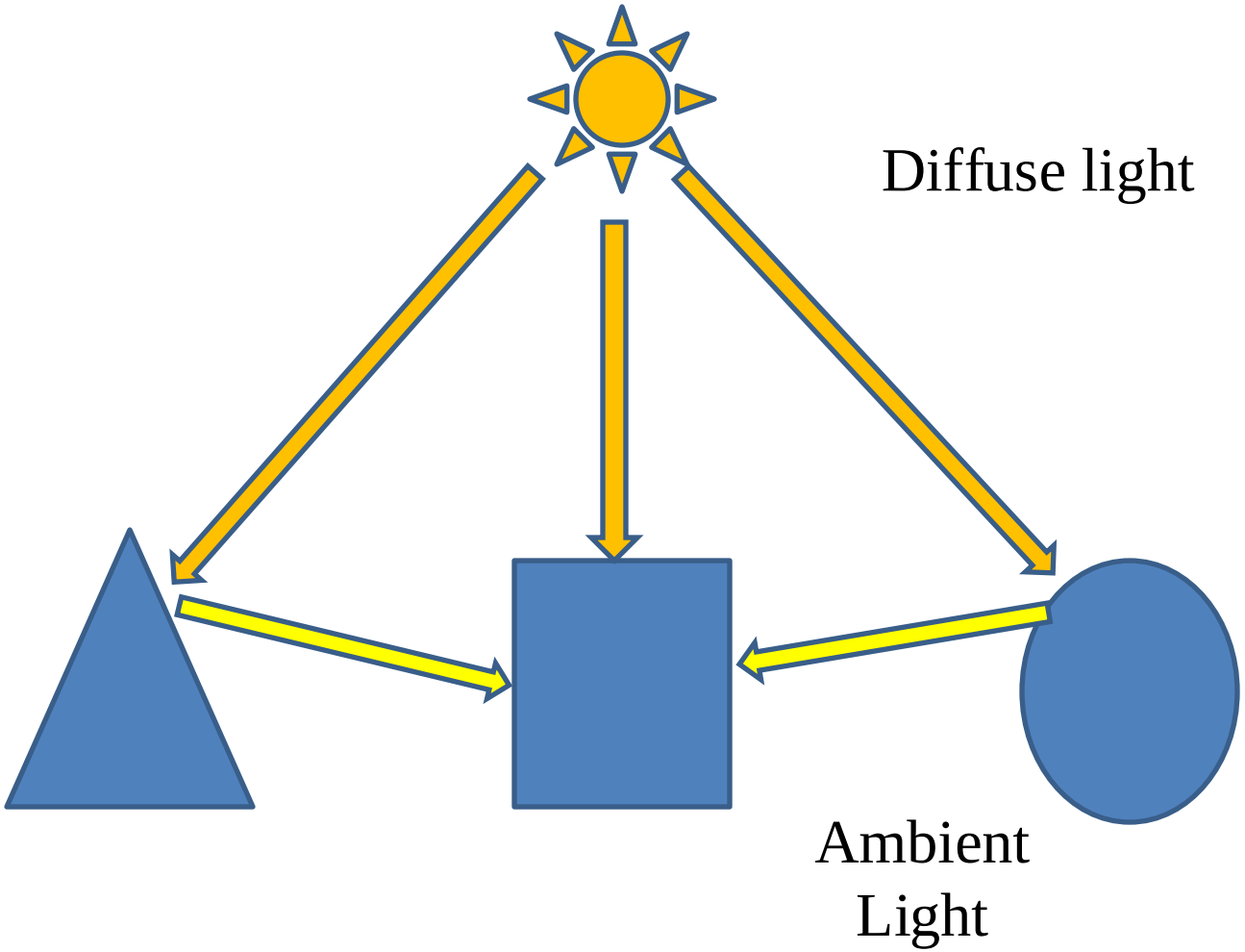




- **Directional Source**
  - Given by a direction
  - Simplifies some calculations
  - Intensity dependents on angle between surface normal and direction of light
  - Distant source $[x\ y\ z\ 0]^T$



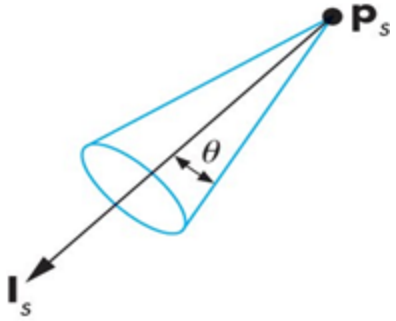
Ambient Light vs Diffuse Light



Spot Light

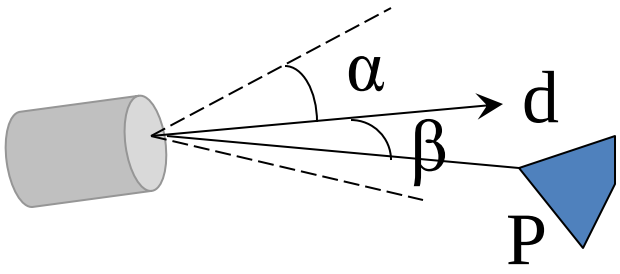
- *Spotlight* is a point source that emits light in restricted set of directions,
  - Requires color, point, direction, falloff parameter
  - usually direction boundary forms a cone shape.
  - Here θ is Cutoff Cone. No light is seen at points lying outside Cutoff angle.

- Intensity falls off directionally



Spot Light

- Consider, a spot light aimed at direction d.
  - $\boldsymbol{\beta}$ is the angle between aimed direction ***d*** and a line from source to object ***P***
  - Source has cut off angle $\boldsymbol{\alpha}$ and intensity ***I***
  - Intensity of spot light received at ***P*** is

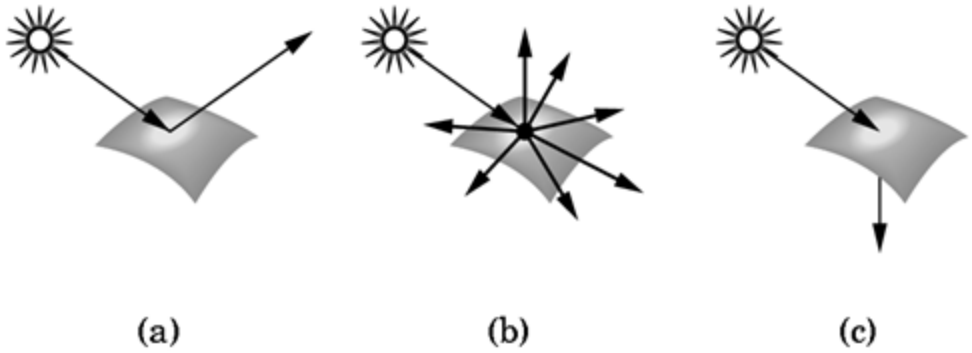


Intensity at P = $I \cos^{\varepsilon}(\beta)$

Types of Surface

Interaction between light and material can be classified as

- **Specular surfaces** – Ideal mirror
- **Diffuse surfaces** – Reflected light is ideally reflected to all directions uniformly
- **Translucent surfaces** – Allow some lights to penetrate the surface (e.g. refraction in glass, water)



Reflection

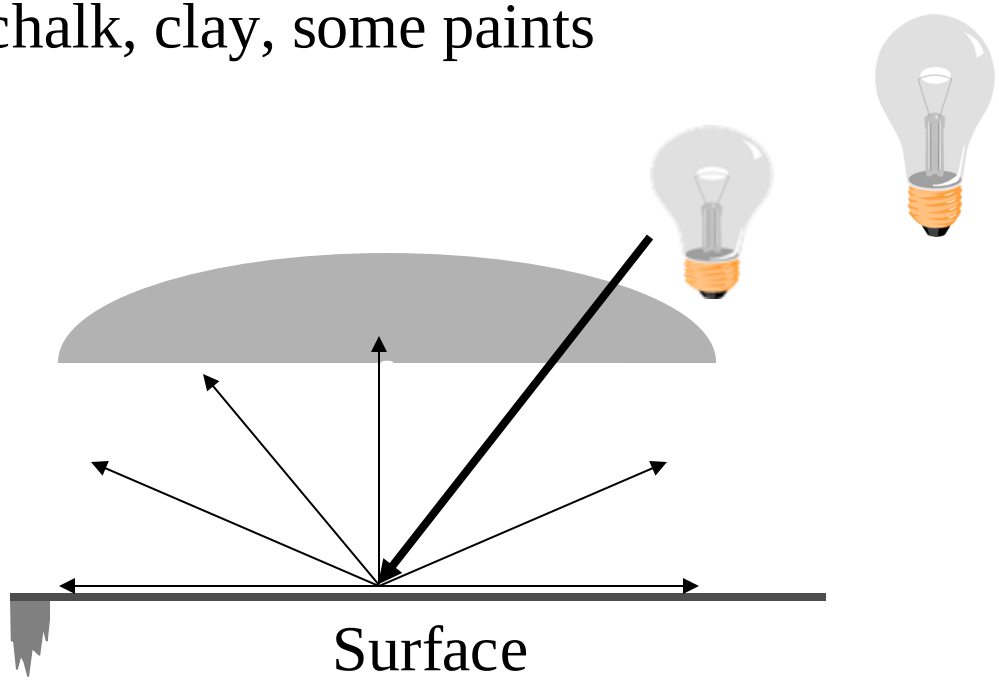
Interaction between light and material can be classified as

- **Specular reflection** – Ideal mirror
- **Diffuse reflection** – Reflected light is ideally reflected to all directions uniformly

* Usually both specular and diffuse reflection can take place at each of type of surface (e.g. mentioned as specular surface, diffuse surface or translucent surface), but their amounts depends on the surface property.

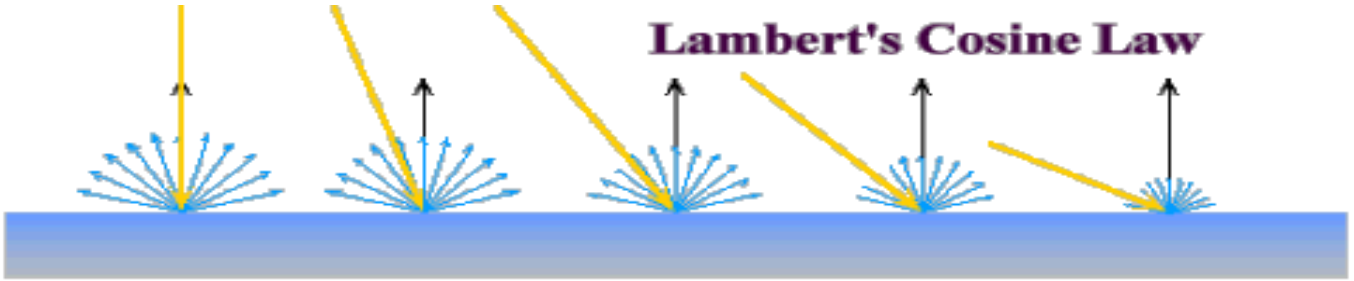
Ideal Diffuse Reflection

- Assumes surface reflects equally in all directions.
- An ideal diffuse surface is, at the microscopic level, a very rough surface.
  - Example: chalk, clay, some paints



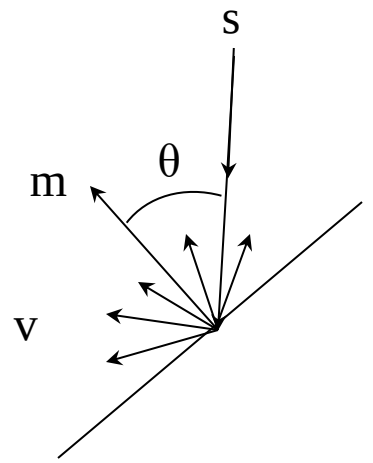
Ideal Diffuse Reflection

- Ideal diffuse reflectors reflect light according to Lambert's cosine law.
- Lambert's law determines how much of the incoming light energy is reflected. The reflected intensity is independent of the viewing direction.
- But reflected light intensity depends on incident angle of light.



Ideal Diffuse Reflection

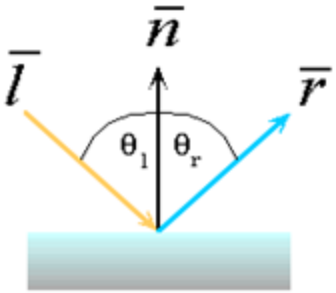
- Suppose light is incident on diffuse surface S.
  - m = direction of Normal of the surface
  - v = viewer's direction
  - s = direction of incident light
  - θ = incident angle
  - I_s = Intensity of Light Source
  - Intensity of reflected light, I_d



  - $I_d = I_s \rho_d (s.m/|s||m|)$, where ρ_d is *the diffuse reflectance coefficient*.
  - If θ is negative, then $I_d = \max(I_s \rho_d (s.m/|s||m|), 0)$

Ideal Specular Reflection

- Reflection is only at mirror angle. An ideal mirror is a purely specular reflector.
  - View dependent reflection. That is, reflected light's intensity varies with viewer's position.
  - Intensity of refelcted light is stronger near mirror angle and strongest at mirror angle.
  - An Ideal specular reflection follows Snell's Law.

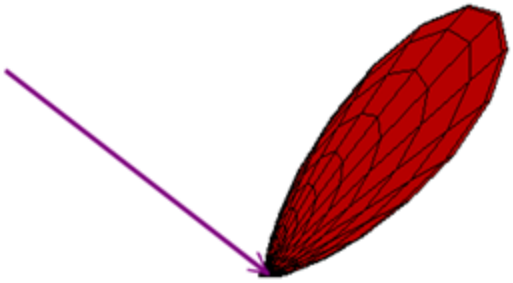


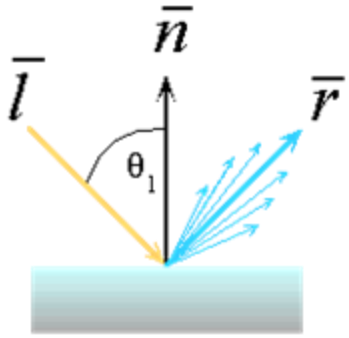
Snell's Laws:

- The incoming ray and reflected ray lie in a plane with the surface normal
- The angle that the reflected ray forms with the surface normal equals the angle formed by the incoming ray and the surface normal

Non-ideal Reflectors

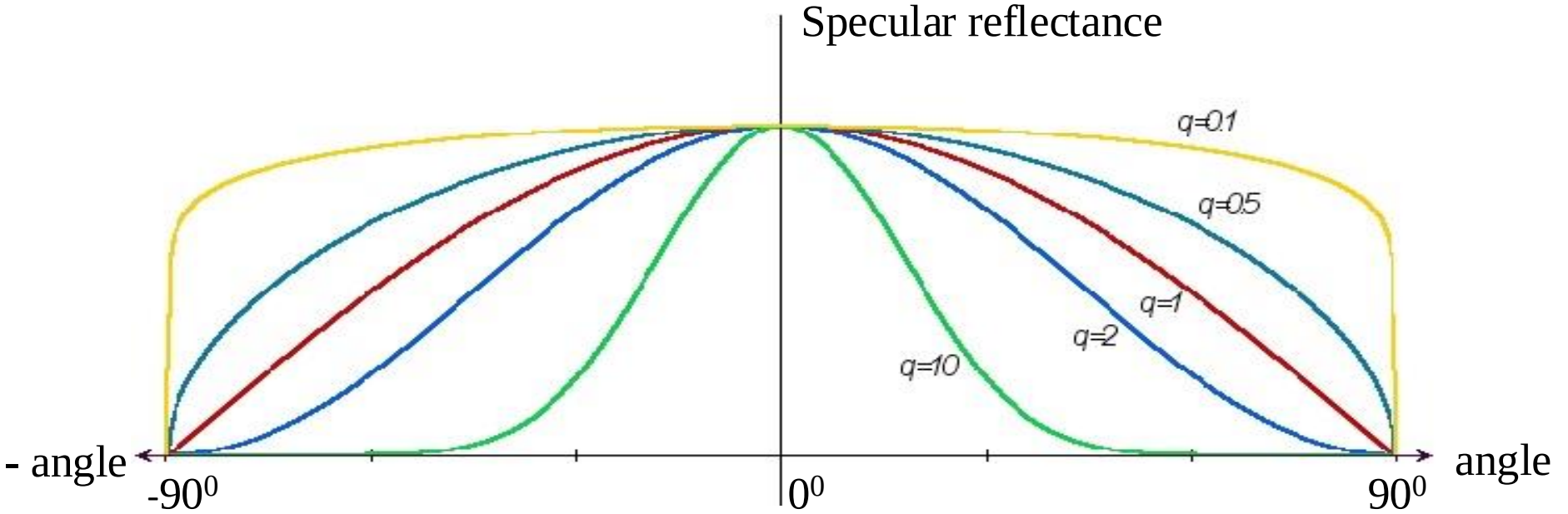
- Simple Empirical Model:
  - We expect most of the reflected light to travel in the direction of the ideal ray.
  - However, because of microscopic surface variations we might expect some of the light to be reflected just slightly offset from the ideal reflected ray.
  - As we move farther and farther, in the angular sense, from the reflected ray we expect to see less light reflected.





Non-ideal Reflectors

- Phong model approximates the fall off of specular reflection.
- The cosine term is maximum when the surface is viewed from the mirror direction and falls off to 0 when viewed at 90 degrees away from it.
- A scalar (shiny property, q in the figure below, which is the exponent term, k in the specular reflection formula of slide 20) controls the rate of this fall off

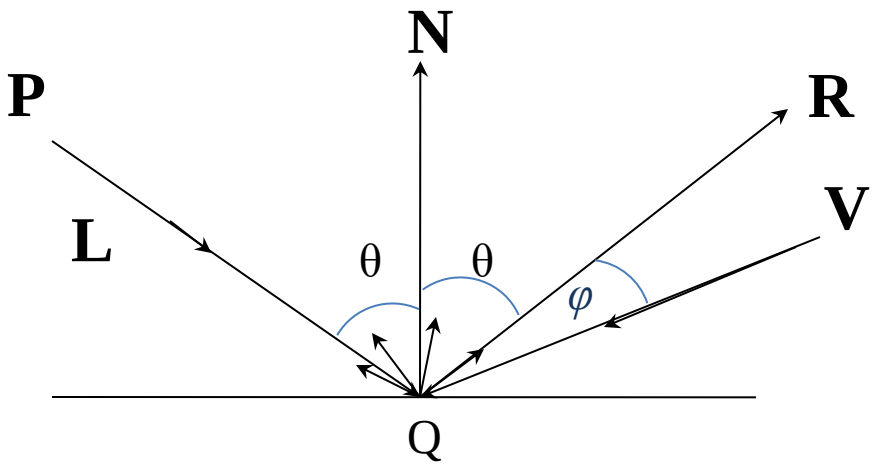


Phong Model

- Compute the combined impact of ambient light, diffuse and specular reflection at a point on surface
- Also called Local Illumination model as its main focus is on the direct impact of light coming from a source.
- This model has no physical basis, yet it is one of the most commonly used illumination models in computer graphics.

Phong Model

- Also models secondary effect of light.
- *Consider a point light source p and viewpoint v. What should be color of light reflected into viewer's eye from point Q of the surface?*
  - *Consider **diffuse** component, **specular** component of the incident light and **ambient** light present in the environment*



Mathematical Calculation of Phong Model

- First we consider, 2 extreme cases of light reflection.
  - Diffuse Reflection :

    $I_d \alpha \cos\theta$

    $= I_p k_d \cos\theta$ [Using k_d instead of previous ρ_d]

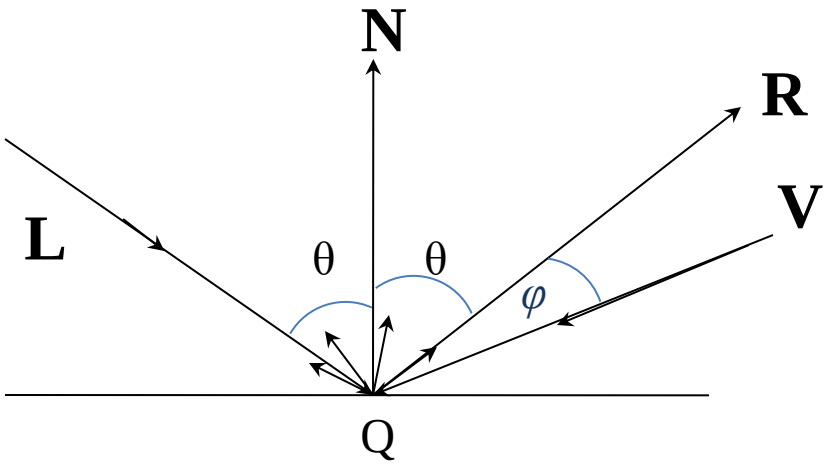
    $= I_p k_d$ (**L**.**N**)

  - Specular Reflection:

    $I_s \alpha (\cos\varphi)^k$

    $= I_p k_s (\cos\varphi)^k$

    $= I_p k_s$ (**R**.**V**)^k



* I_p is the intensity of the light source

Mathematical Calculation of Phong Model

- We need to ircorporate the effect light present in environment. Thus total refelcted light also includes ambient component.
  - Ambient Component = $I_a k_a$
- Total reflected light intensity from Q,

  I = Ambient Component + Diffuse Component + Specular Component

  $= I_a k_a + I_p k_d (\mathbf{L}.\mathbf{N}) + I_p k_s (\mathbf{R}.\mathbf{V})^k$

  More specifically,

  $I = I_a k_a + I_p [k_d \max \{(\mathbf{L}.\mathbf{N}),0\} + k_s \max\{(\mathbf{R}.\mathbf{V})^k ,0\}]$

Additional Issues

- When there are n light sources in the scene, their effects are cumulative: Intensity at Q,

  $I = I_a k_a + \sum_{(i=1 \text{ to } n)} I_{pi} \{k_d (\mathbf{L}.\mathbf{N}) + k_s (\mathbf{R}.\mathbf{V})^k \}$

- The intesnsity of red, green and blue component of reflected light,

  $I_r = I_a k_{ar} + I_p k_{dr} (\mathbf{L}.\mathbf{N}) + I_p k_s (\mathbf{R}.\mathbf{V})^k$

  $I_g = I_a k_{ag} + I_p k_{dg} (\mathbf{L}.\mathbf{N}) + I_p k_s (\mathbf{R}.\mathbf{V})^k$

  $I_b = I_a k_{ab} + I_p k_{db} (\mathbf{L}.\mathbf{N}) + I_p k_s (\mathbf{R}.\mathbf{V})^k$

  – k_s : coefficient for specular component which is same as the color of light source, not affected by surface color.

How to get vector **R ?**

- Find a formula to compute R, the reflection of vector L with respect to normal vector?

  $\mathbf{R} = 2\,(\mathbf{L}.\mathbf{N})\mathbf{N} - \mathbf{L}$

  (See Solved Problem 11.10 in Schaum(2^{nd} edition))

Blinn and Torrence Variation

- Calculation of **R** is computaionally expensive. So in phong model the term **R.V** is somtimes replaced by **N.H** , where **H** is a unit vector that bisect the angle between **L** and **V**.
  - angle between **N** and **H** measures the falloff of intensity.
  - Though calculation of **N.H** is is computaionally inexpensive relative to **R.V**, but **N.H** is not always equal to **R.V**. In that case calculation of specular component will be approximate.

    (See Solved Problem 11.11 in Schaum(2^{nd} edition)))

Effect of the Reflection Coefficients

- Diffuse Reflection Coefficient, k_d
- Ambient Reflection Coefficient, k_a
- Specular Reflection Coefficient, k_s
- Specular Component Exponent Term, k

* To observe the effects of different components, vary one component and keep the other two constant.

Diffuse Reflection Coefficient

- $I_d = \max \{I_s k_d \cos\theta, 0\}$
- Source intensity is 1.0
- Background intensity is 0.4
- Sphere reflecting diffuse light, for six reflection coefficients: 0, 0.2, 0.4, 0.6, 0.8, and 1.

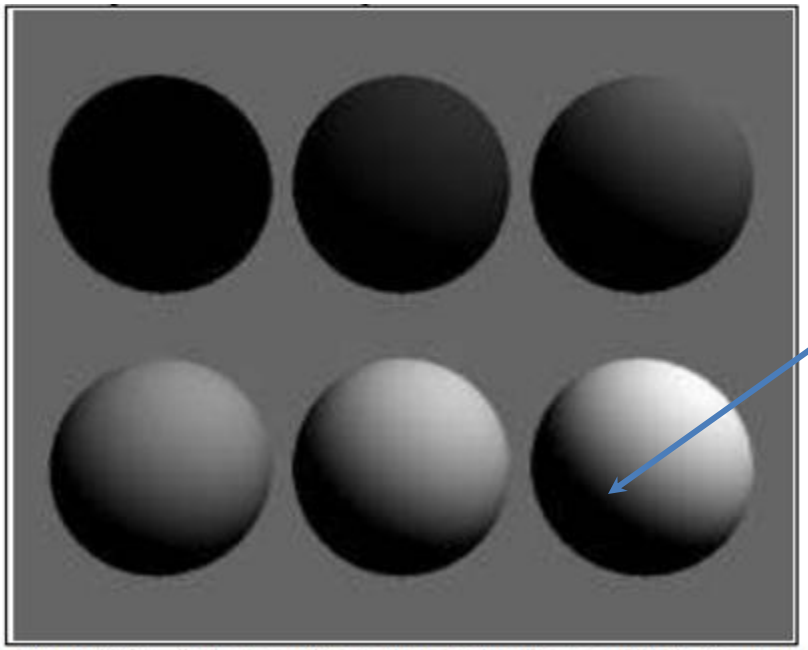


Figure 8.11. Spheres with various reflection coefficients shaded with diffuse light. (file: fig8.11.bmp)

Angle θ between surface normal and incident light is > 90^o

- What is the ambient component here?
- What is the specular component?

Ambient Reflection Coefficient

- Effect of adding ambient light to the diffuse light reflected by a sphere
- Diffuse source intensity is 1.0
  Diffuse reflection coefficient is 0.4
  Ambient source intensity is 1.0
- Moving from left to right the ambient reflection coefficient takes on values: 0.0, 0.1, 0.3, 0.5, and 0.7
  - Too little ambient light makes shadows too deep and harsh
  - Too much makes the picture look washed out and bland

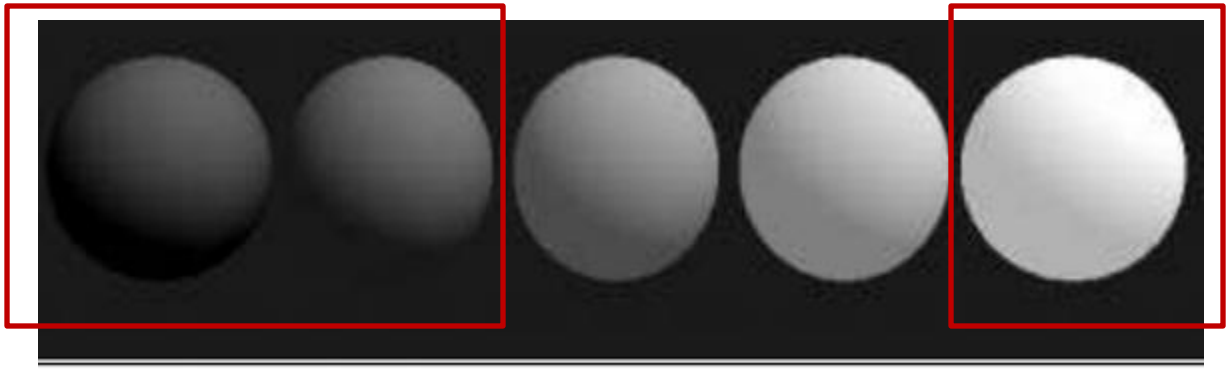


Figure 8.16. On the effect of ambient light.

Specular Reflection Coefficient

- The ambient and diffuse reflection coefficients are 0.1 and 0.4 for all spheres.
- $I_d = I_s\, k_s\, (\cos \varphi)^k$

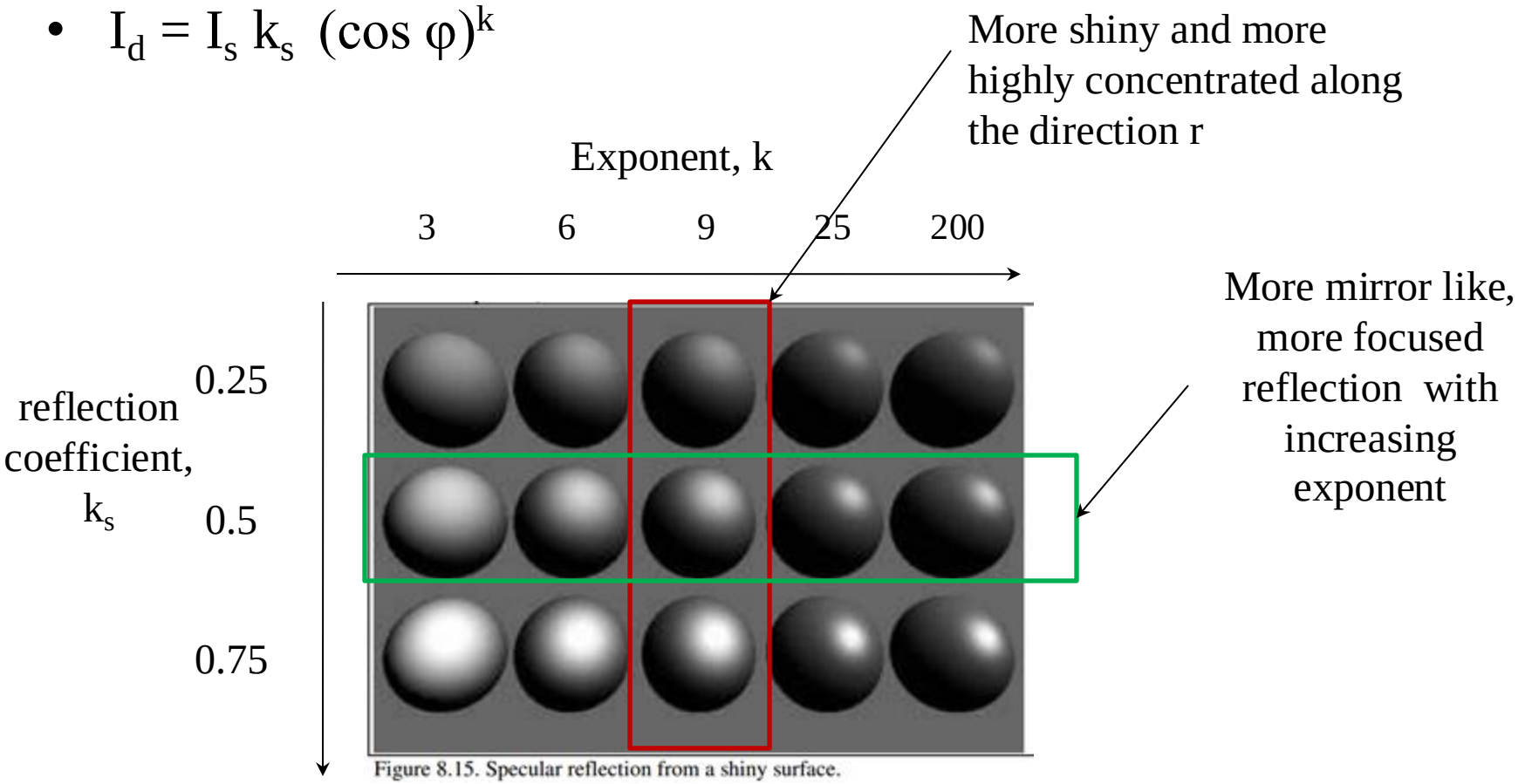


Figure 8.15. Specular reflection from a shiny surface.

Color

- Color is constructed by adding certain amounts of red, green, and blue light
- Light sources have three "types" of color:

  ambient = (I_{ar}, I_{ag}, I_{ab}), diffuse = (I_{dr}, I_{dg}, I_{db}), and specular = $(I_{spr}, I_{spg}, I_{spb})$
- Surface has two sets of reflection co-efficient
  - Ambient reflection coefficient : k_{ar} , k_{ag} , k_{ab}
  - Diffuse reflection coefficient : k_{dr} , k_{dg} , k_{db}
- Color of the specular component is often the same as that of the light source – mirror like, so the specular reflection coefficient: k_{sr} , k_{sg} , $k_{sb} = k_s$

  e.g. ***Glossy*** red apple when illuminated by a yellow light is yellow rather than red

Color of an Object

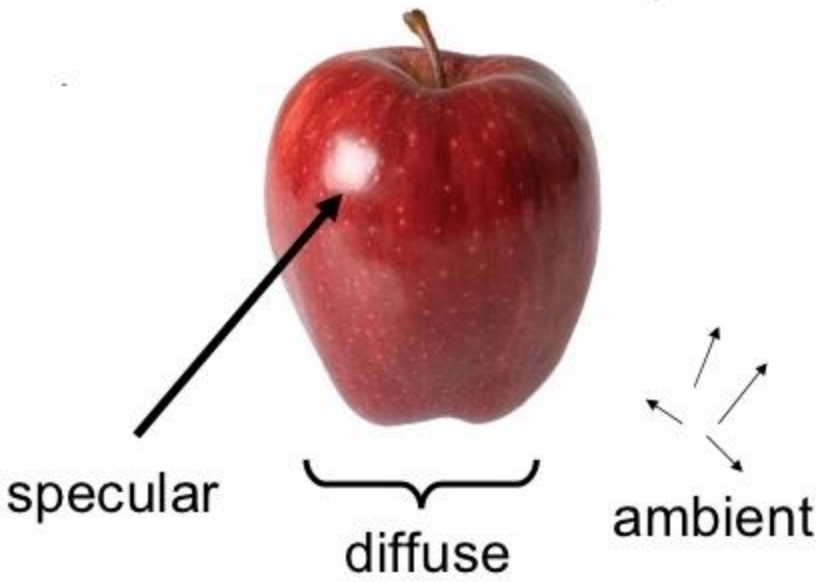


Image synthesis view:

"light, surface, and material interact to reflect light perceived as color, modeled via simplifying assumptions"

Color of an Object

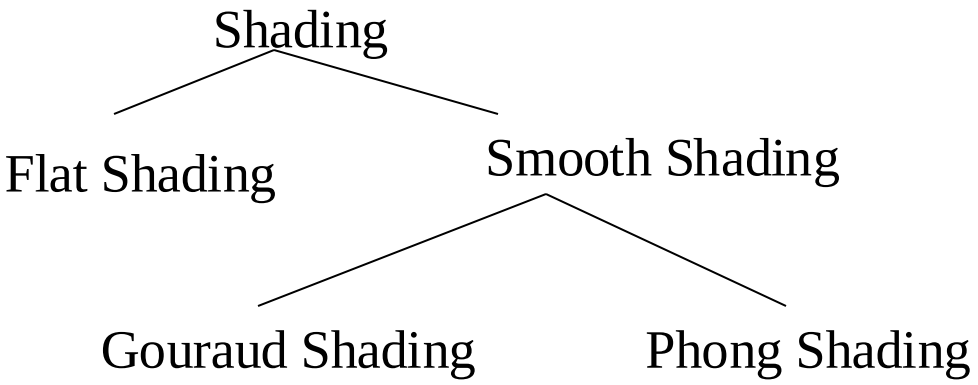
- By “color” of a surface we mean the color that is reflected from it when the illuminated by white light
- The ambient and diffuse reflection coefficients are based on the color of the surface itself.
- The specular reflection coefficient is based on the color of the light source (and the roughness property of the surface)

Color of an Object

- Color of a sphere is 30% red, 45% green, and 25% blue
- Light source in environment:
  White light (Intensity : $I_{sr} = I_{sg} = I_{sb} = I_s$)
- Sphere's ambient and diffuse reflection coefficients:
  $k_{ar} = k_{dr} = 0.30k$
  $k_{ag} = k_{dg} = 0.45k$
  $k_{ab} = k_{db} = 0.25k$
  here, k is some scaling value
- The individual ambient and diffuse components have intensities
  $I_{ar} = 0.30k\, I_a$, $I_{dr} = 0.30k\, I_s \cos\theta$
  $I_{ag} = 0.45k\, I_a$, $I_{dg} = 0.45k\, I_s \cos\theta$
  $I_{ab} = 0.25k\, I_a$, $I_{ab} = 0.25k\, I_s \cos\theta$
- If the environment is uniformly lighted, $\theta = 0^0$ for sphere!!

Shading

- The process of assigning colors to pixels.



Shading Model

- Flat Shading
  - Compute Phong lighting once for entire polygon
- Gouraud Shading
  - Compute Phong lighting at the vertices and interpolate lighting values across polygon
- Phong Shading
  - Interpolate normals across polygon and perform Phong lighting across polygon

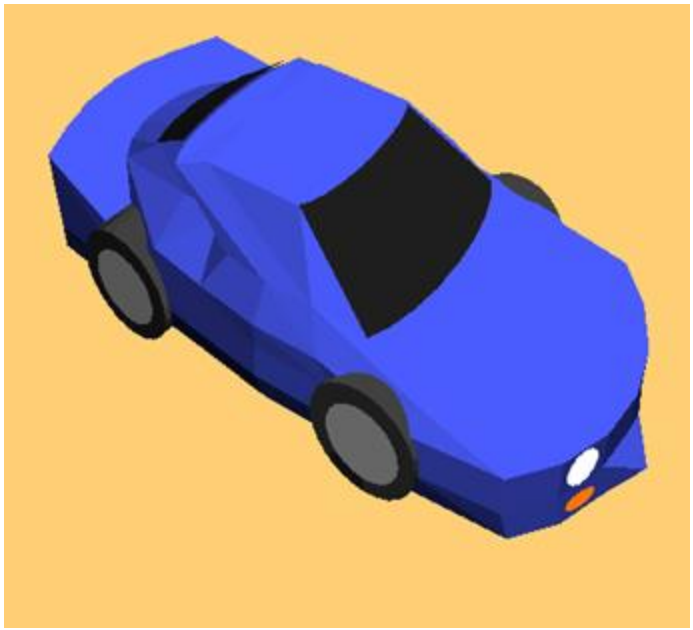
Flat Shading

- For each polygon
  - Determines a single intensity value at a chosen point on the polygon
  - Uses that value to shade the entire polygon

- Assumptions
  - Light source at infinity
  - Viewer at infinity
  - The polygon represents the actual surface being modeled

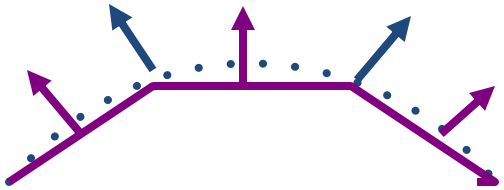
Problems of Flat Shading

- Specular highlights tends to get lost
- If chosen point on polygon is at location of the light soure, then color of the polygon will be significantly distorted.



Smooth Shading

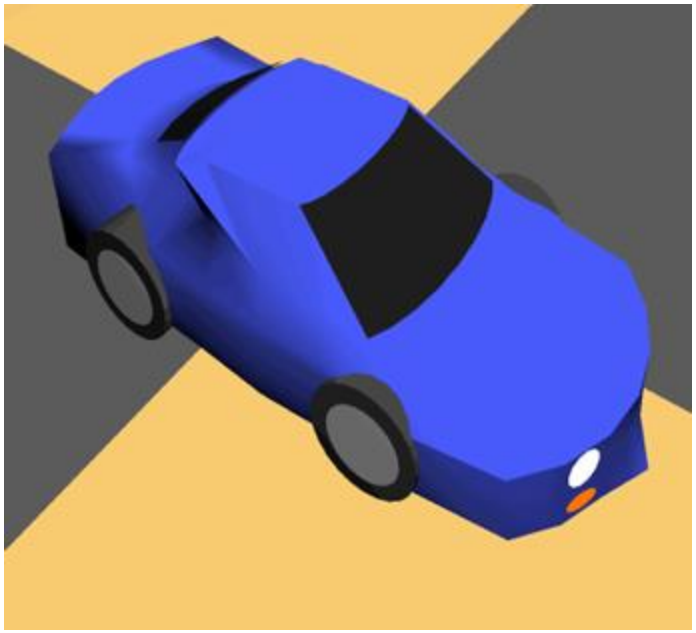
- Introduce vertex normals at each vertex
  - Used only for shading
  - Think of as a better approximation of the real surface that the polygons approximate
  - Finds color value for each point in the polygon individually



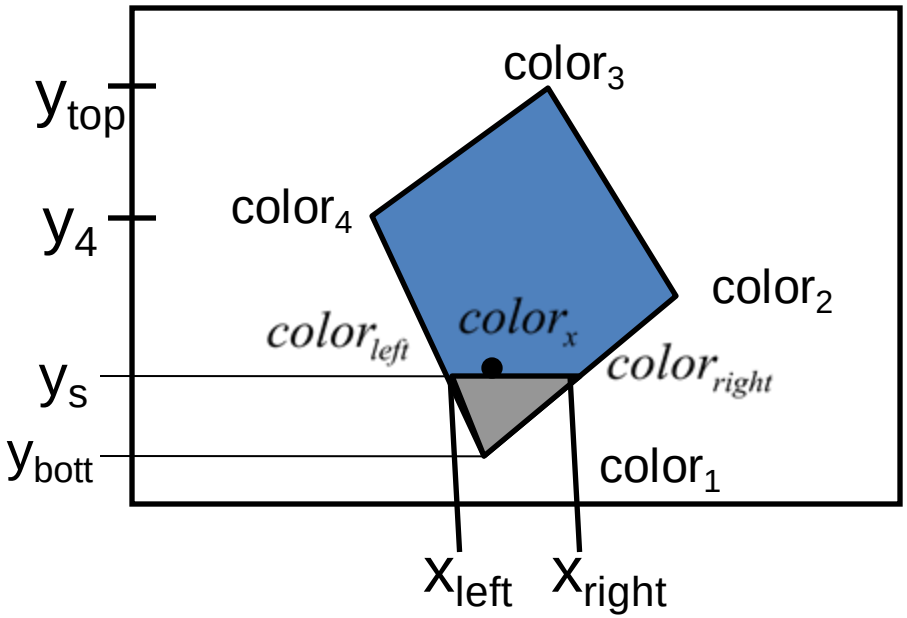
- Two types
  - Gouraud Shading
  - Phong Shading (do not confuse with Phong Lighting Model)

Gouraud Shading

- Most common approach
- Perform Phong lighting at the vertices
- Linearly interpolate the resulting colors over faces
  - Along edges
  - Along scanline



Gouraud Shading

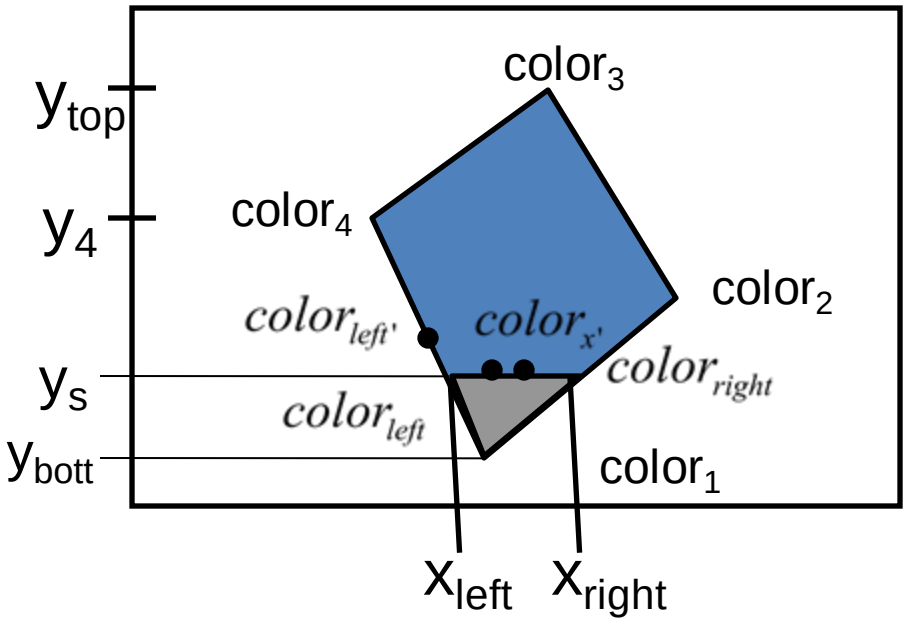


$$color_{left} = color_1 + (color_4 - color_1)\frac{y_s - y_{bott}}{y_4 - y_{bott}}$$

$$color_{right} = color_1 + (color_2 - color_1)\frac{y_s - y_{bott}}{y_2 - y_{bott}}$$

$$color_x = color_{left} + (color_{right} - color_{left})\frac{x - x_{left}}{x_{left} - x_{right}}$$

Gouraud Shading



$$color_x = color_{left} + \left(color_{right} - color_{left}\right)\frac{x - x_{left}}{x_{left} - x_{right}}$$

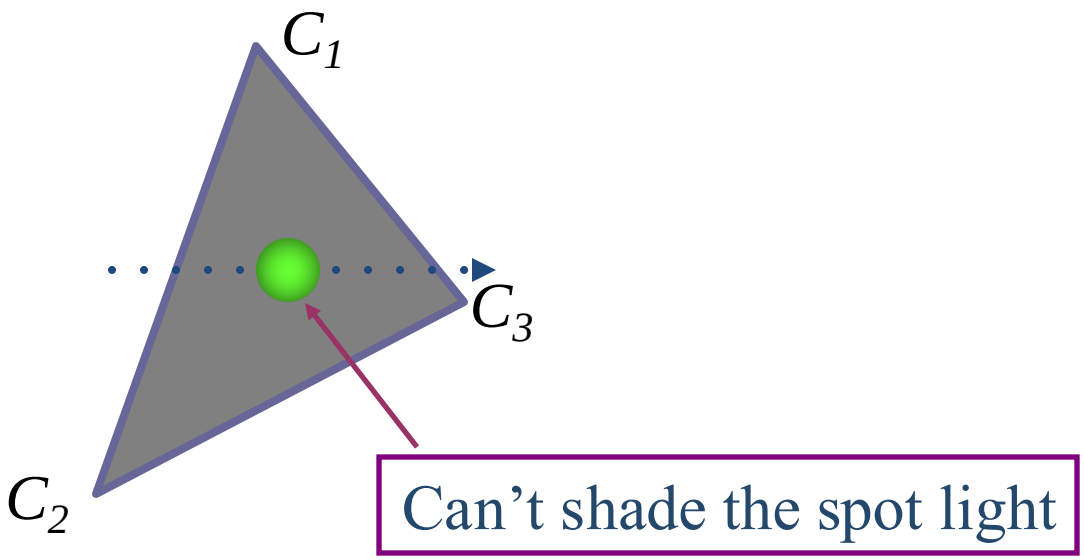
$$color_{x'} = color_x + K\Delta x$$

$$color_{left'} = color_{left} + K'\Delta y$$

Calculate the surface normals along the scan line and the edge using incremental approach

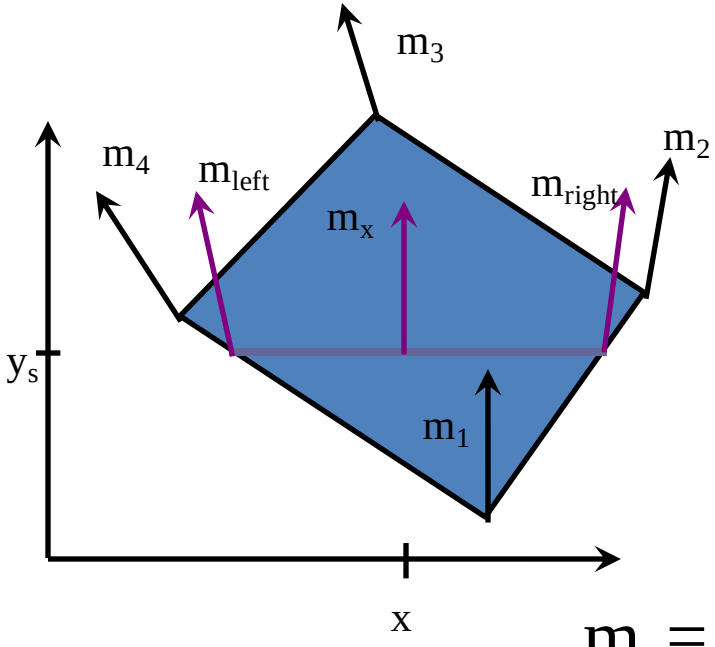
Problem of Gouraud Shading

- Often appears dull
- Lacks accurate specular component



Phong Shading

Interpolate normal vectors of face vertices at each pixel, then perform Phong lighting at each pixel.



$$m_{left} = m_1 + (m_4 - m_1)\frac{y_s - y_1}{Y_4 - y_1}$$

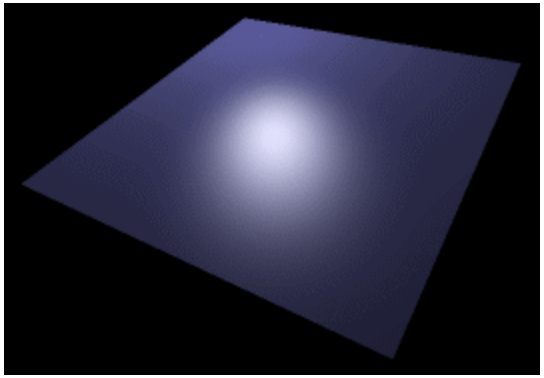
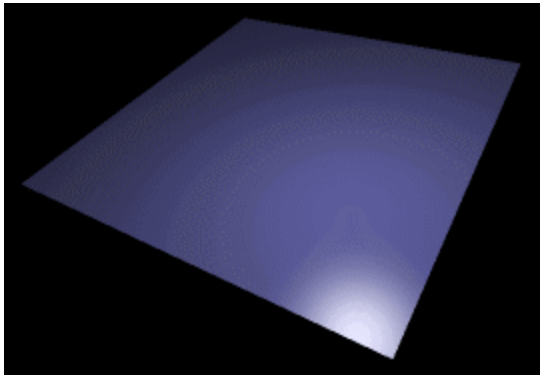
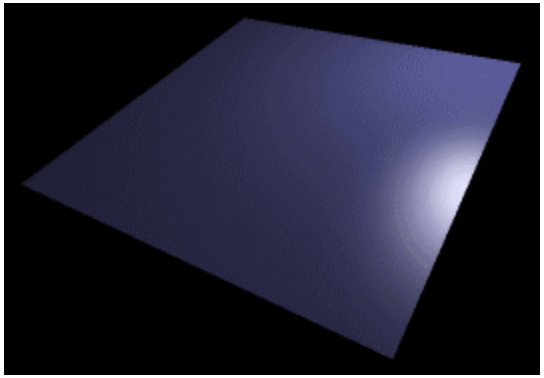
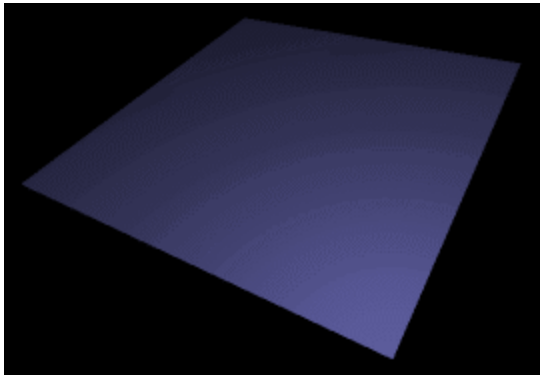
$$m_{right} = m_1 + (m_2 - m_1)\frac{y_s - y_1}{Y_2 - y_1}$$

$$m_x = m_{left} + (m_{right} - m_{left})\frac{x - x_{left}}{x_{right} - x_{left}}$$

Calculate the surface normals along the scan line and the edge using incremental approach

Phong vs Gouraud Shading

- Phong shading is more smooth
- If a highlight does not fall on a vertex, Gouraud shading may miss it completely, but Phong shading does not.

Texture

- Texture Types
  - Bitmap textures: Using some image file
  - Procedural textures: Changing pixel intensity in some controlled fashion without using external image source

- Texture (s, t) produces a color or intensity value for each value of s and t between 0 and 1

Texture

- Bitmap textures: Using some image file having dimension CxR presented as txtr[c][r]
  - 0<=c<=C-1 and 0<=r<=R-1

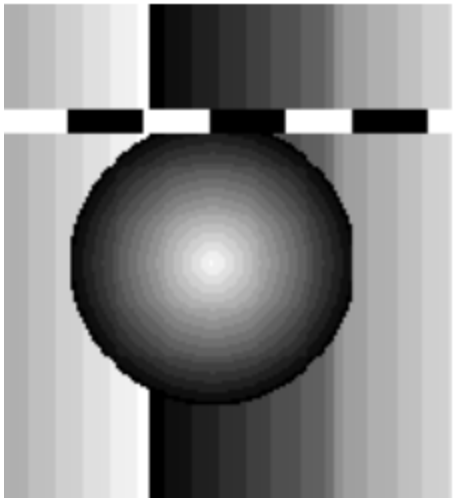
```
Color3 texture(float s, float t)
{
      return txtr[(int)(s * C)][(int)(t * R)];
}
```

- Example: C= 600 and R= 400
  texture(0.261,0.783) evaluates to txtr[156][313]
  texture(1,1) evaluates to txtr[600][400]

Texture

- Procedural textures: Changing pixel intensity in some controlled fashion without using external image source

```
float fakeSphere(float s, float t)
{
  float r = sqrt((s-0.5)*(s-0.5)+(t-0.5)*(t-0.5));
  if(r < 0.3) return 1 - r/0.3; // sphere intensity
  else return 0.2;  // dark background
}
```



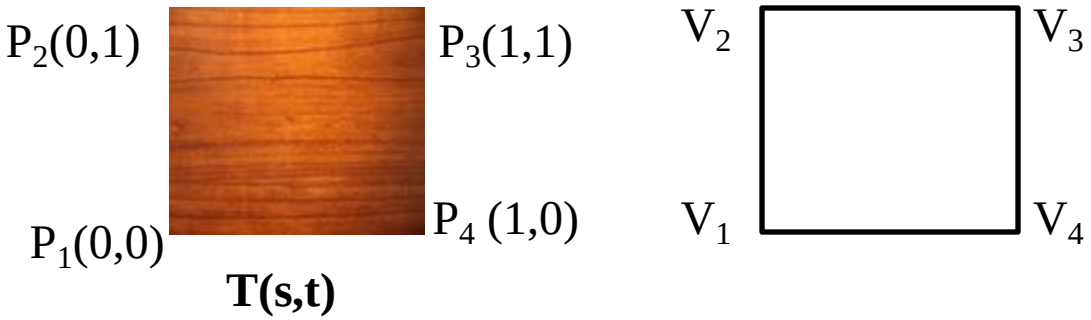
- Not used frequently

(Bitmap textures that is image files are commonly used)

Texture

Adding Texture to Flat Surface

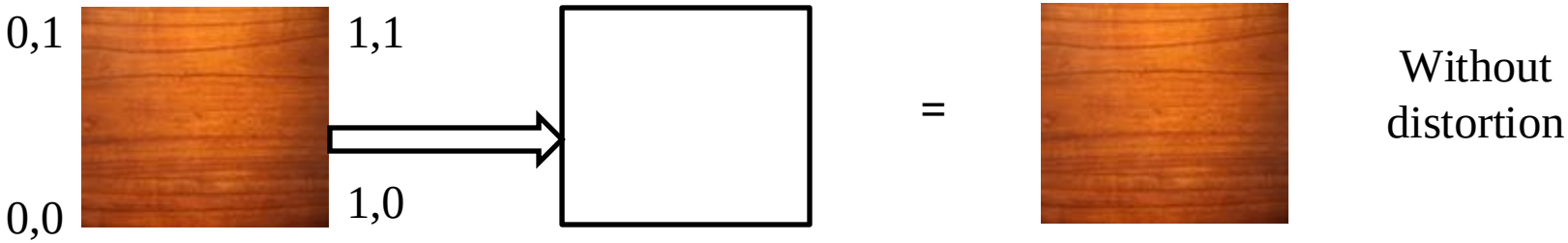
- Texture P and Surface S has
  - same shape
  - Same number of vertices
- Attaching a P_i to each V_i



Texture

Adding Texture to Flat Surface

- Texture P and Surface S has same shape

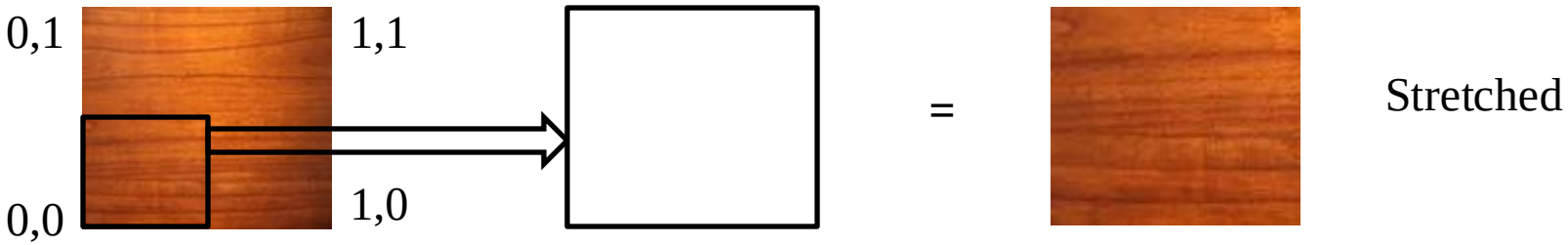



```
glBegin(GL_QUADS);
   glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.0, 1.5);
   glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 3.0, 1.5);
   glTexCoord2f(1.0, 1.0); glVertex3f(2.0, 3.0, 1.5);
   glTexCoord2f(1.0, 0.0); glVertex3f(2.0, 2.0, 1.5);
glEnd();
```

Texture

Adding Texture to Flat Surface

- Texture P and Surface S has same shape

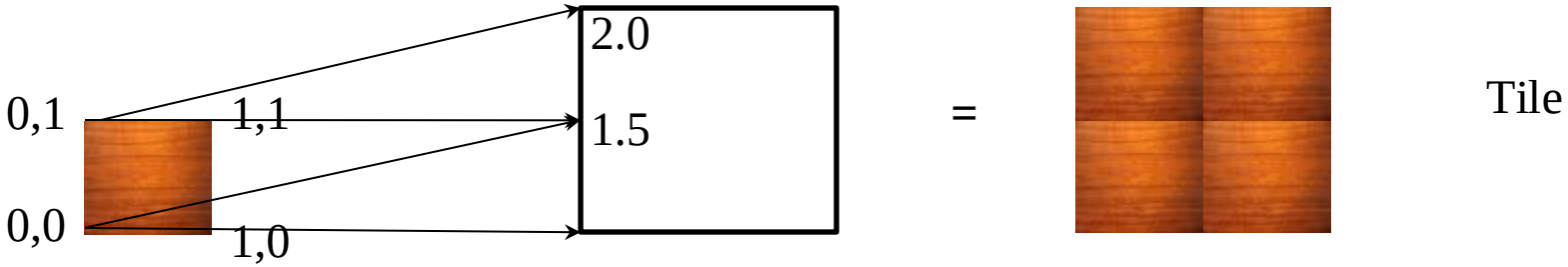



```
glBegin(GL_QUADS);
   glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.0, 1.5);
   glTexCoord2f(0.0, 0.5); glVertex3f(1.0, 3.0, 1.5);
   glTexCoord2f(0.5, 0.5); glVertex3f(2.0, 3.0, 1.5);
   glTexCoord2f(0.5, 0.0); glVertex3f(2.0, 2.0, 1.5);
glEnd();
```

Texture

Adding Texture to Flat Surface

- Texture P and Surface S has same shape

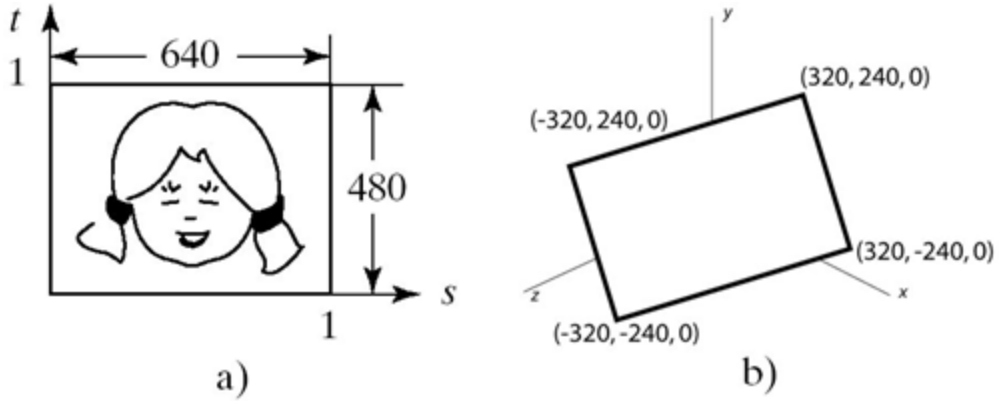



```
glBegin(GL_QUADS);
   glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.0, 1.5);
   glTexCoord2f(0.0, 2.0); glVertex3f(1.0, 3.0, 1.5);
   glTexCoord2f(2.0, 2.0); glVertex3f(2.0, 3.0, 1.5);
   glTexCoord2f(2.0, 0.0); glVertex3f(2.0, 2.0, 1.5);
glEnd();
```

Texture

Adding Texture to Flat Surface

- Texture P and Surface S has same shape



- Mapping is clearly affine
  - Scaling
  - Rotation
  - Translation

Creating Visual Effect Using Texture

1. Creating a glowing object

- I = texture(s, t)


```
gl.glEnable(GL.GL_TEXTURE_2D);
gl.glTexEnvf(GL.GL_TEXTURE_ENV, GL.GL_TEXTURE_ENV_MODE,
    GL.GL_REPLACE);
gl.glBindTexture(GL.GL_TEXTURE_2D, texName);
glBegin(GL_QUADS);
  glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.0, 1.5);
  glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 3.0, 1.5);
  glTexCoord2f(1.0, 1.0); glVertex3f(2.0, 3.0, 1.5);
  glTexCoord2f(1.0, 0.0); glVertex3f(2.0, 2.0, 1.5);
glEnd();
```

Creating Visual Effect Using Texture

2. Modulating the reflection coefficient

$$I = texture(s,t)[I_a\rho_a + I_d\rho_d \times lambert] + I_{sp}\rho_s \times phong^f$$

- Object is the color of
  - Reflected diffuse light component
  - Reflected ambient light component

```
gl.glEnable(GL.GL_TEXTURE_2D);
glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE, GL_MODULATE);
gl.glBindTexture(GL.GL_TEXTURE_2D, texName);
glBegin(GL_QUADS);
  glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.0, 1.5);
  glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 3.0, 1.5);
  glTexCoord2f(1.0, 1.0); glVertex3f(2.0, 3.0, 1.5);
  glTexCoord2f(1.0, 0.0); glVertex3f(2.0, 2.0, 1.5);
glEnd();
```

Creating Visual Effect Using Texture

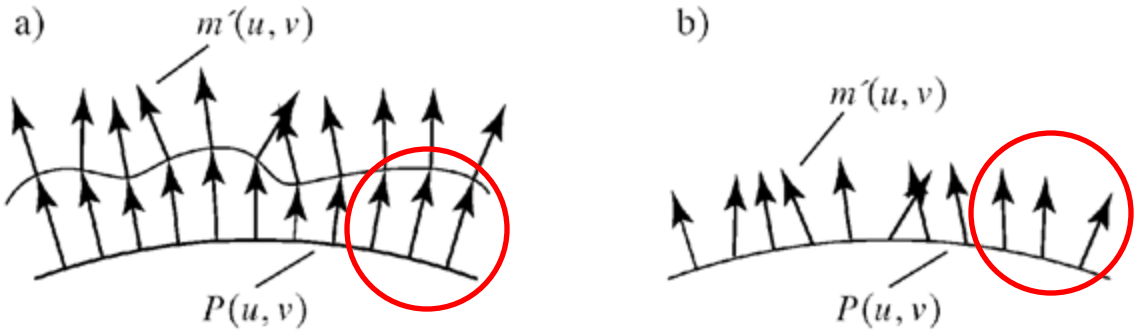
3. Simulating Roughness by Bump Mapping

– Technique to give surface wrinkled or dimpled appearance without having to model each individual dimple

– Independent of the viewing angle and object orientation

Creating Visual Effect Using Texture

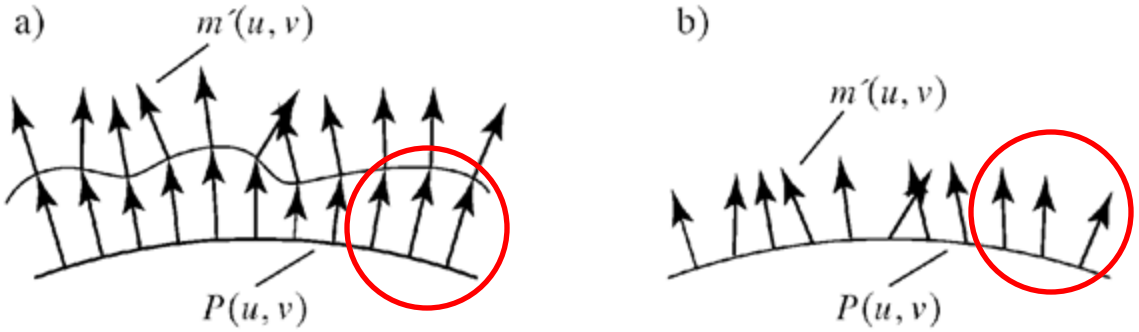
3. Simulating Roughness by Bump Mapping



- Scalar function *texture(s, t) perturb the normal vector at each spot in a* controlled fashion. It causes perturbations in the amount of diffuse and specular lights.

Creating Visual Effect Using Texture

3. Simulating Roughness by Bump Mapping



- Surface is represented parametrically by the function *P(u, v)*
- Surface has unit normal vector **m(*u, v*)**
- 3D point at (*u*, v*) corresponds to the texture at (u*, v*)*
- *New normals can be found by:*

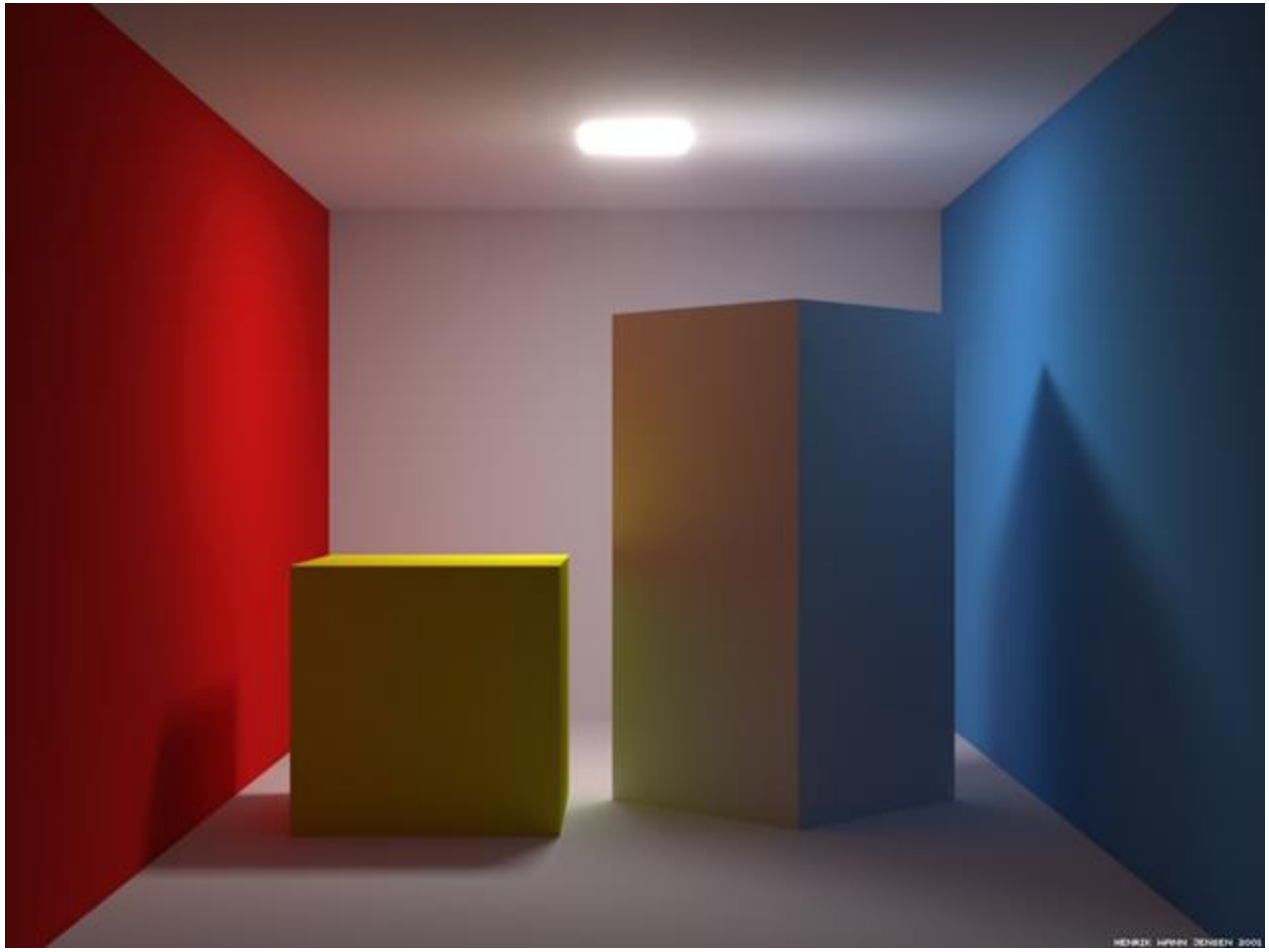
$$\mathbf{m}'(u^*, v^*) = \mathbf{m}(u^*, v^*) + \mathbf{d}(u^*, v^*)$$

where the perturbation vector **d** is given by

$$(u^*, v^*) = (\mathbf{m} \times P_v)texture_u - (\mathbf{m} \times P_u)texture_v$$

$texture_u$ and $texture_v$ are partial derivatives of the texture function wrt u and v

Shadows

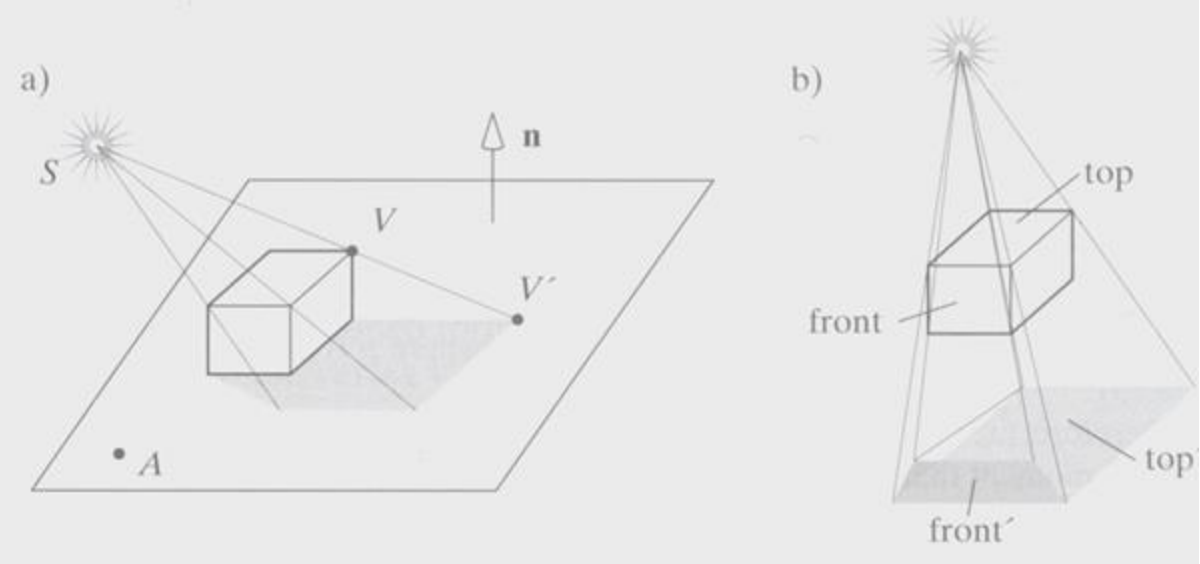


Adding Shadows of Objects

1. Shadows as Texture
2. Shadows Using a Shadow Buffer

Shadows as Texture

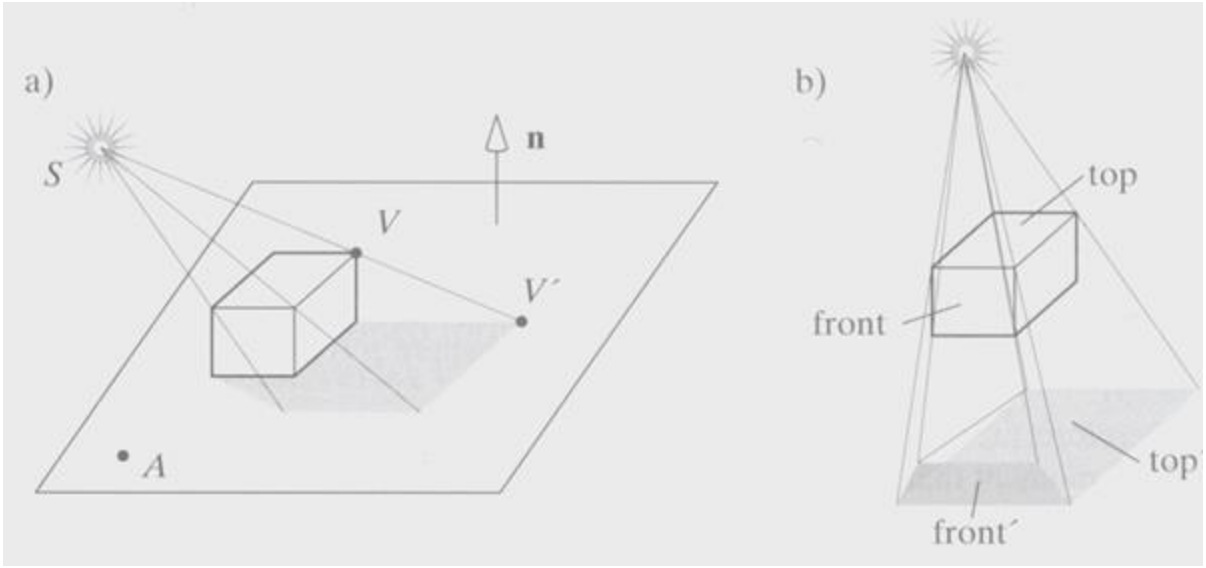
- Paint shadows as a texture
- Works for flat surfaces illuminated by point light source



- Problem: compute shape of shadow

Shadows as Texture

* Union of projections of individual faces = projection of entire object

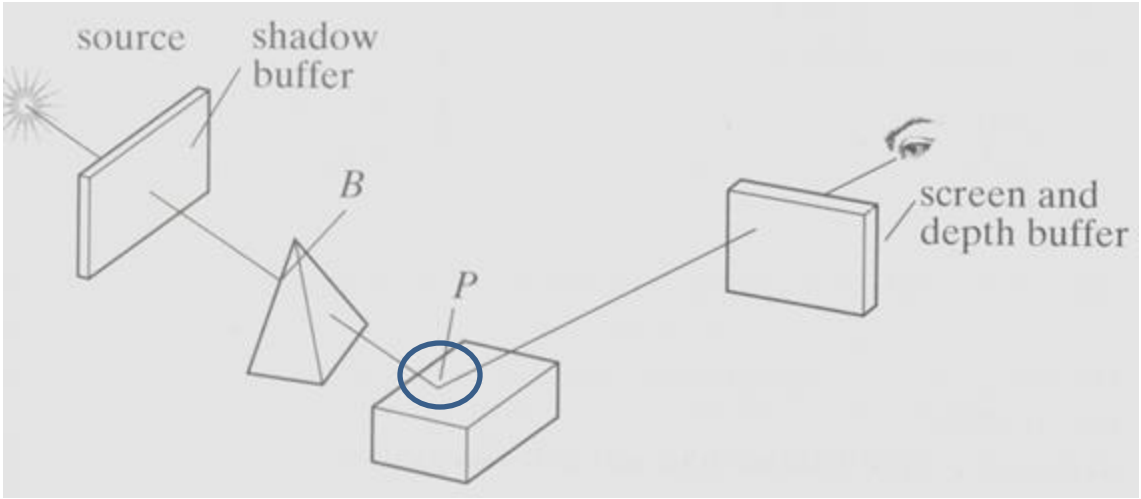


1. First, draw plane using specular-diffuse-ambient components
2. Then, draw shadow projections (face by face) using only ambient component

Shadows Using a Shadow Buffer

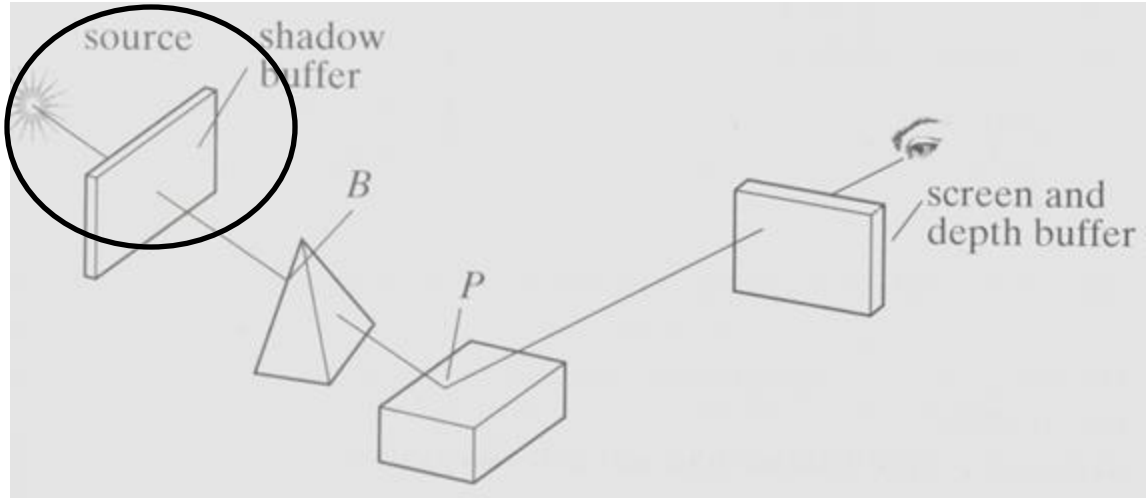
Shadow Buffer: An auxiliary second depth buffer

- Any points in the scene that are “hidden” from the light source must be in shadow.
- If no object lies between a point and the light source the point is not in shadow.



Shadows Using a Shadow Buffer

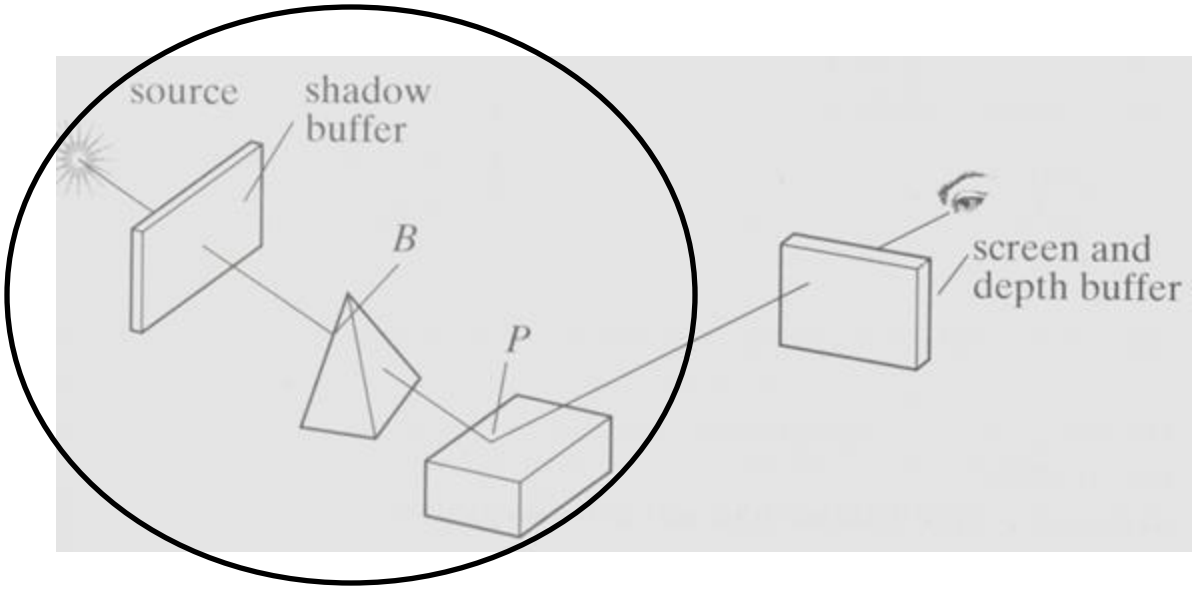
- Contains a “depth picture” of the scene from the point of view of the light source
- Each of its index (i, j) records the distance from the source to the closest object in the associated direction.



Shadows Using a Shadow Buffer

Step 1: Loading Shadow Buffer:

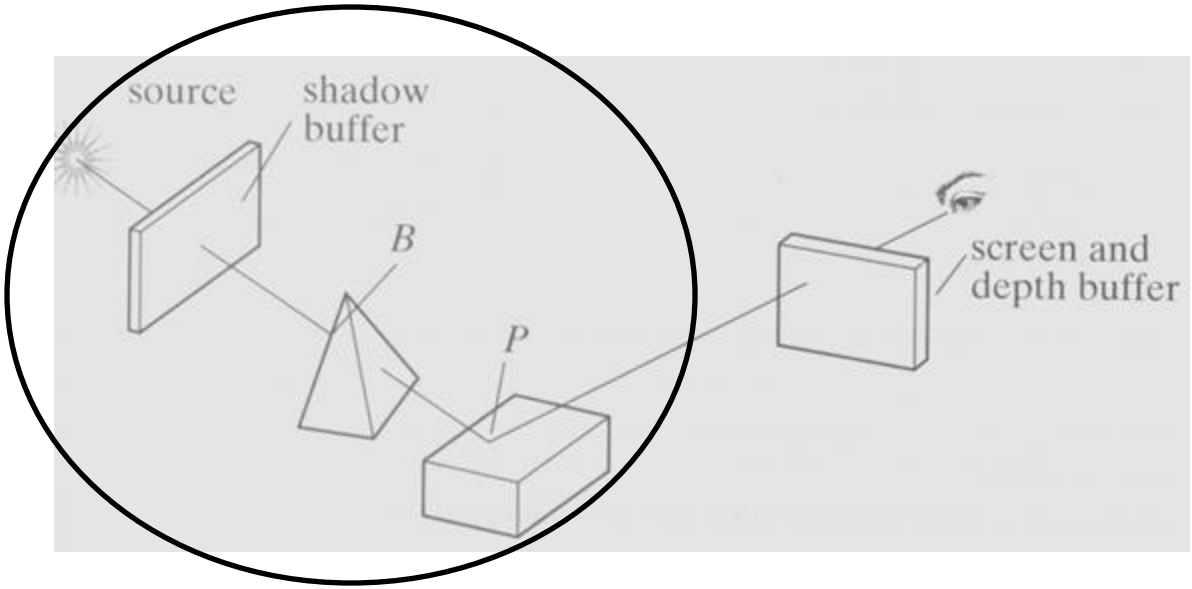
- Initialize each index value d[i][j] to 1.0
- Position a camera at light source
- Rasterize each face in scene updating pseudo-depth
- Shadow buffer tracks smallest pseudo-depth so far



Shadows Using a Shadow Buffer

Step 1: Loading Shadow Buffer:

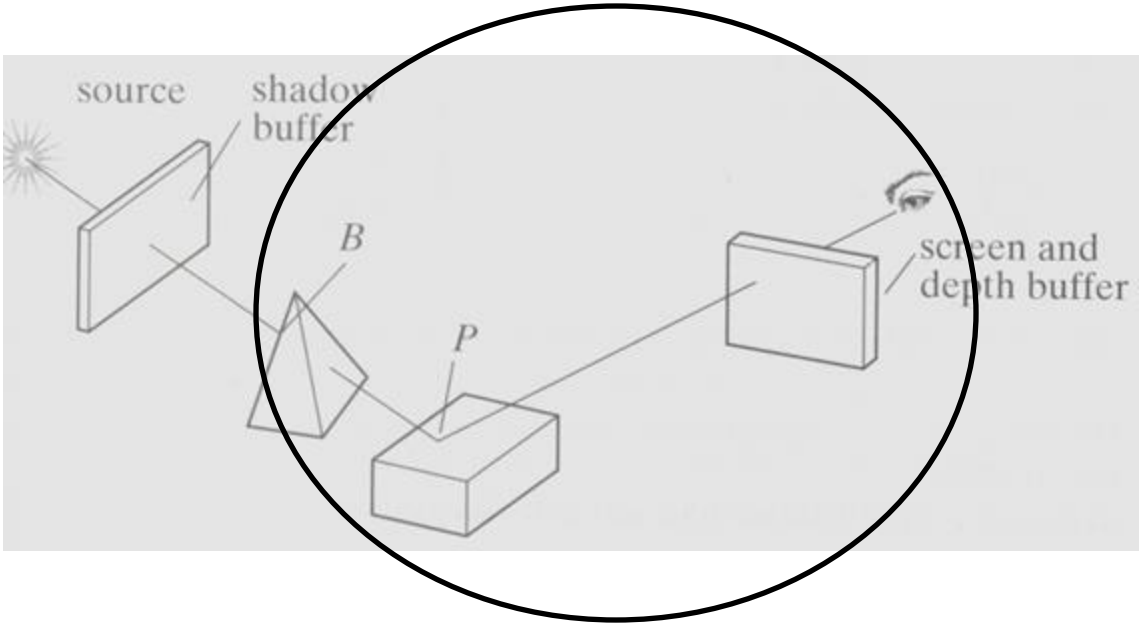
- Shadow buffer calculation is independent of eye position
- In animations, shadow buffer loaded once
- If eye moves, no need for recalculation
- If objects move, recalculation required



Shadows Using a Shadow Buffer

Step 2: Rendering Scene:

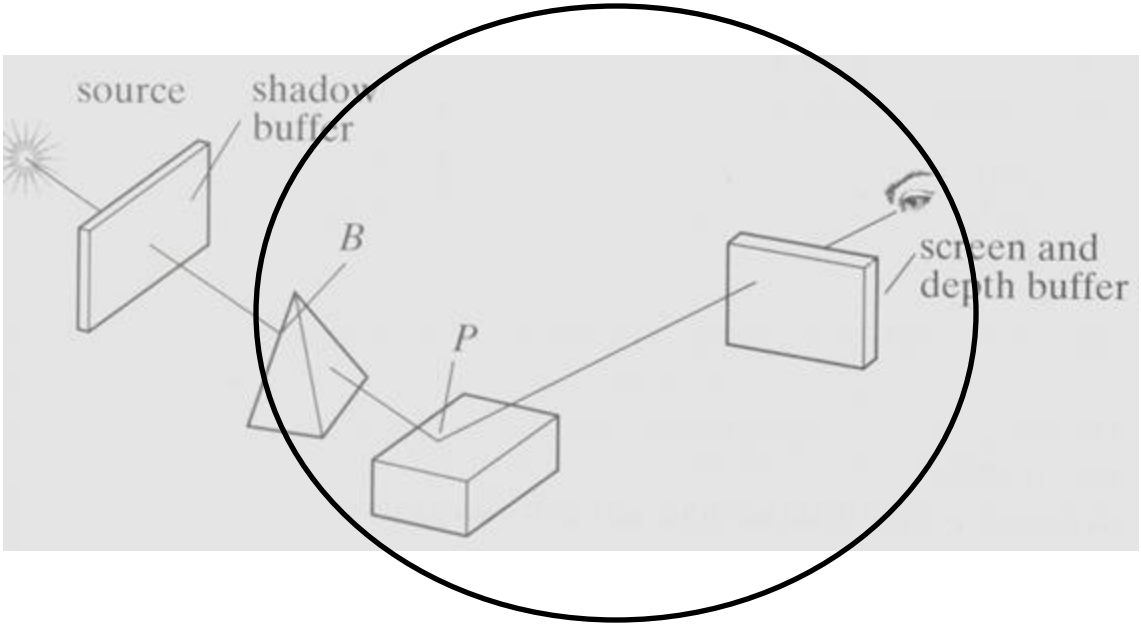
- Render scene using camera as usual
- While rendering a pixel [c][r], find:
  - pseudo-depth D from light source to P
  - Index location [i][j] in shadow buffer, to be tested
  - Value d[i][j] stored in shadow buffer



Shadows Using a Shadow Buffer

Step 2: Rendering Scene:

- If d[i][j] < D (other object on this path closer to light)
  - point P is in shadow
  - set lighting using only ambient
- Otherwise, not in shadow



Thank you ☺