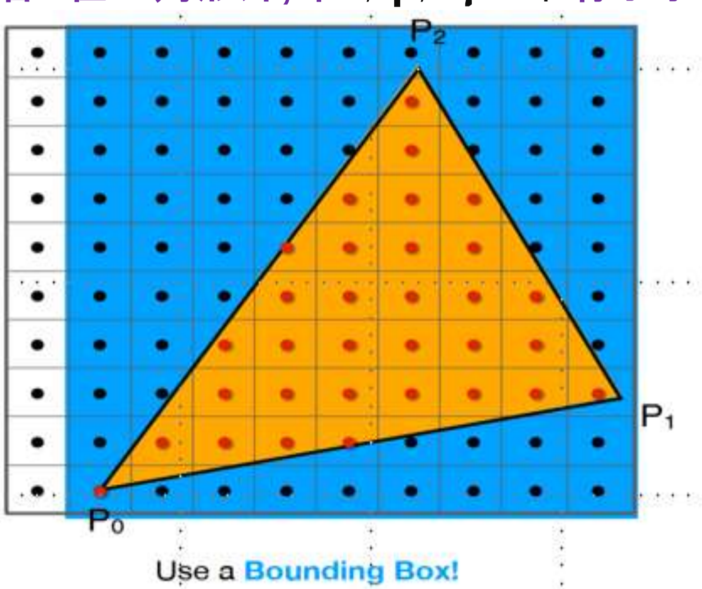
The University of New Mexico

Recap

➢三角形光栅化算法:

- 在三角形ABC的包围盒内测试每个像素点P(x,y)，判P是否在三角形内: Inside(P,A,B,C)
  - 若P在三角形内，则输出P(片元位置)，并插值计算出P的颜色(片元颜色)；
    - 根据三角形顶点ABC，计算P点的重心坐标(α，β， γ)。如果α，β， γ都大于0，则P在三角形内；并用P的重心坐标对三角形顶点的颜色进行插值得P点颜色。
  - 若P在三角形边界上，即 α，β， γ有等于0的情况，则....
  - 若P在三角形外,即α，β， γ之中有小于0的情况，则舍弃P。

$P_0(x_0,y_0)$
$P_1(x_1,y_1)$
$P_2(x_2,y_2)$

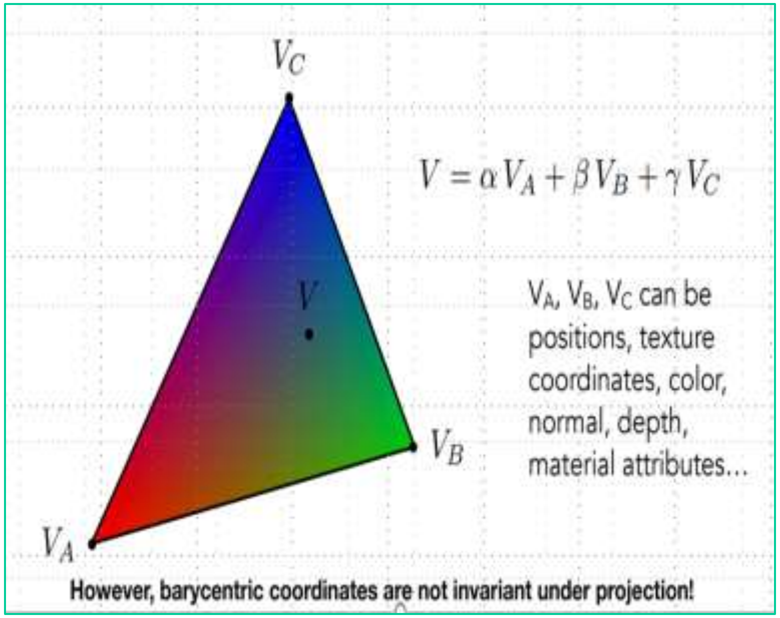


//Bouding Box（xmin,xmax,ymin,ymax）

$x_{min} = \text{floor}(x_i)$
$x_{max} = \text{ceiling}(x_i)$
$y_{min} = \text{floor}(y_i)$
$y_{max} = \text{ceiling}(y_i)$
for $y = y_{min}$ to y_{max} **do**
  for $x = x_{min}$ to x_{max} **do**
    $\alpha = f_{12}(x,y)/f_{12}(x_0, y_0)$
    $\beta = f_{20}(x,y)/f_{20}(x_1, y_1)$
    $\gamma = f_{01}(x,y)/f_{01}(x_2, y_2)$
    if $(\alpha > 0$ and $\beta > 0$ and $\gamma > 0)$ **then**
      $\mathbf{c} = \alpha \mathbf{c}_0 + \beta \mathbf{c}_1 + \gamma \mathbf{c}_2$
      drawpixel (x, y) with color $\mathbf{c}$

Today's theme

- ➢三角形“每顶点”的颜色/明暗如何计算得到？
- ➢或，插值后三角形内“每片元”的颜色/明暗如何计算得到？



每顶点着色
Gouraud Shading

每片元着色
Phong Shading

Shadedsphere1

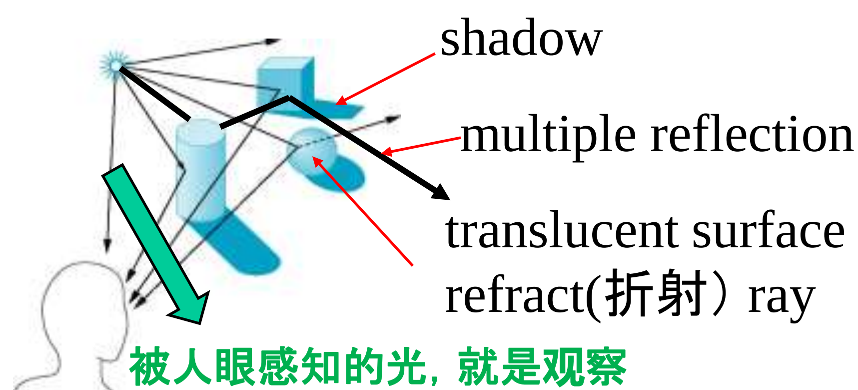
ShadedSphere2

ShadedSphere3

ShadedSphere4

Today's theme(cont.)

- 有光场景的明暗着色(shading)的思路
  - Light-Material Interaction cause each point to have a different color or shade（光与材质交互，导致物体表面的每个点具有不同颜色或明暗）
    - ➢ Light that strikes an object is partially <u>**absorbed（吸收）**</u> and partially <u>**scattered (散射)**</u>
      - ➢ **Scatter（散射）：reflect(反射）or refract（折射）**
    - ➢ A surface appears red under white light because the red component of the light is <u>reflected（反射）</u> and the rest is absorbed,（白光照射的表面呈现红色是因为光的红色成分被反射，而其它成分被物体吸收）
    - ➢ The reflected light（反射光）is scattered in a manner that depends **on the smoothness** and **orientation of the surface**,（反射光散射方式依赖于物体表面光滑度和朝向）
    - ➢ **The amount reflected determines the color and brightness of the object（被人眼/相机所感知的光主要是来自物体表面的散射光（反射光和折射光））**



被人眼感知的光，就是观察者看到的物体的颜色/明暗

The University of New Mexico

Today's theme(cont.)

- Shade： ***n.*** 阴凉处，背阴; 色度; 灯罩; （绘画）阴影部分;
- Shading: darkening明暗/coloring着色

Shading: Definition

- In Merriam-Webster Dictionary

  shad·ing, [ˈʃeɪdɪŋ], noun
  The darkening or coloring of an illustration or diagram with parallel lines or a block of color.

- In this course

  The process of applying a material to an object.

GAMES101 Lingqi Yan, UC Santa Barbara

Outline

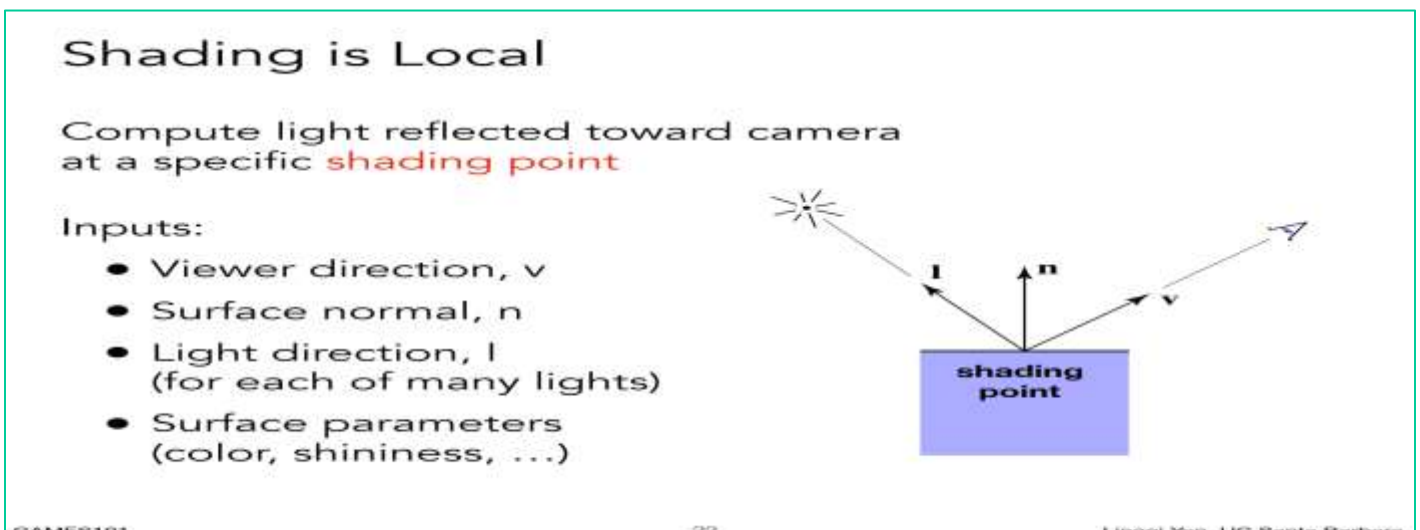
- **Lighting Models（光照模型）**
  - Phong Model
  - Blinn-Phong Model
- **Surface Shading Methods（着色方法）**
  - Flat shading: per face using lighting model
  - Gouraud shading: per vertex using lighting model
  - Phong shading: per fragment using lighting model
- **Shading Realization in Rasterization Pipeline**
  - Flat shading: mainly vertex shader program
  - Gouraud shading: mainly vertex shader program
  - Phong shading: both vertex and fragment shader program

Lighting Model

- **What is "lighting model"**
  - Compute light reflected toward a viewer **at a shading point**

Need to Consider: （注意:所有向量方向都是朝着外的！）

1. Light Direction: 光源方向l
2. Object Surface Orientation:法向量n
3. Viewer Direction :观察方向v
4. Object Surface Material properties:表面材质属性



The University of New Mexico

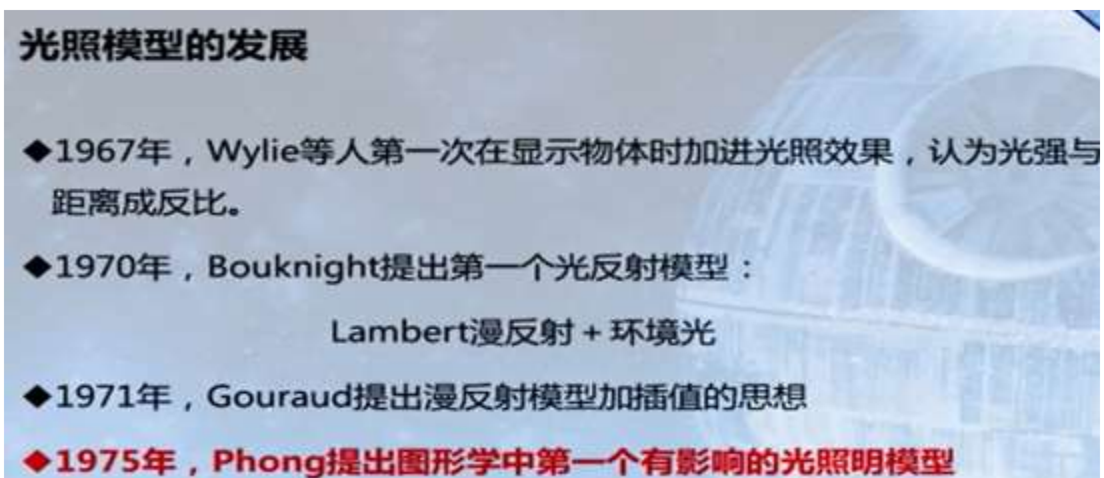
Lighting Model(cont.)

➢Development of Lighting Model

➢Phong: 是一个经验模型，不是基于物理的模型

➢Only consider Reflection Light (three components)

1. Diffuse Light（漫反射光）
2. Specular（镜面反射光）
3. Ambient Light（环境光）



光照模型的发展

◆1967年，Wylie等人第一次在显示物体时加进光照效果，认为光强与距离成反比。

◆1970年，Bouknight提出第一个光反射模型：

Lambert漫反射 + 环境光

◆1971年，Gouraud提出漫反射模型加插值的思想

◆1975年，Phong提出图形学中第一个有影响的光照明模型

Outline

- **Lighting Models（光照模型）**
  - **Phong Model**
  - Blinn-Phong Model
- **Surface Shading Methods（着色方法）**
  - Flat shading: per face using lighting model
  - Gouraud shading: per vertex using lighting model
  - Phong shading: per fragment using lighting model
- **Shading Realization in Rasterization Pipeline**
  - Flat shading: mainly vertex shader program
  - Gouraud shading: mainly vertex shader program
  - Phong shading: both vertex and fragment shader program

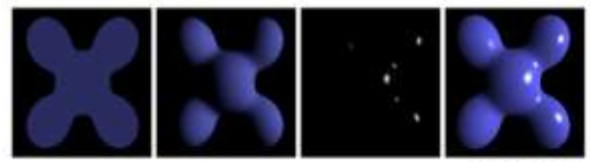
The University of New Mexico

Phong Model

➢Phong Model only consider “Refection Light”

➢Reflection Light has three components

1. Diffuse Light（漫反射光）
2. Specular（镜面反射光）
3. Ambient Light（环境光）



➢物体表面材质简单分成两种类型:光滑和粗糙

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflected the light(产生镜面反射光specular)
- A very rough ... rs ligl ... ons（产生漫反射光diffuse）

smooth surface　　rough surface

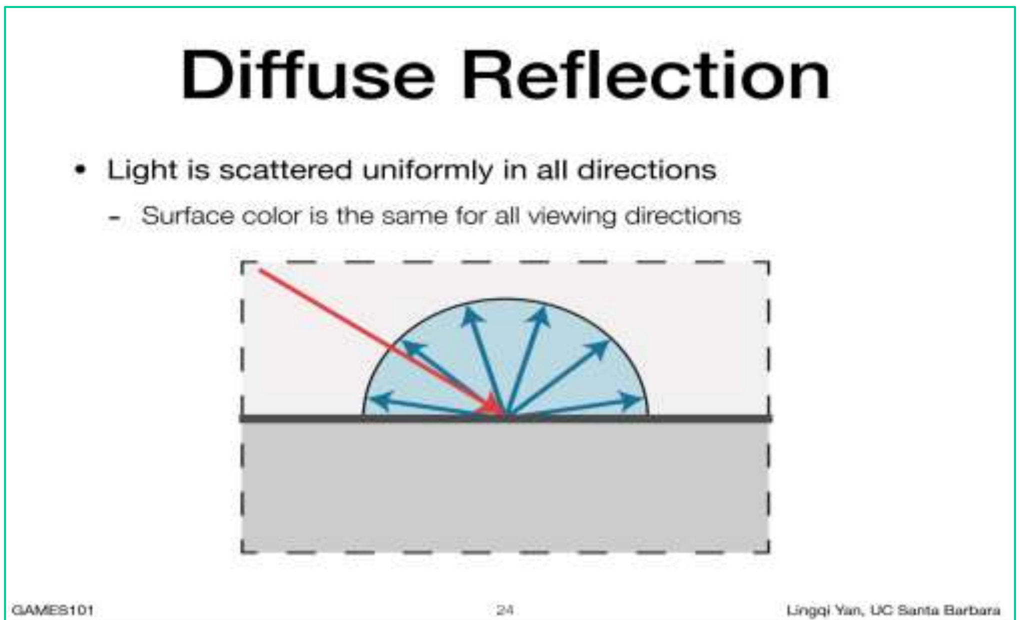
✓ ***物体表面材质一般都具这两种类型特点，只是程度不同！***

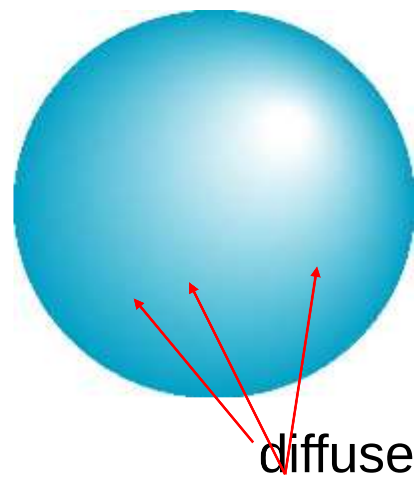
The University of New Mexico

Phong Model(cont.)

➢Diffuse Reflection Light（漫反射光）

- Perfectly diffuse reflector(全漫反射体)
- Light scattered equally in all directions（光在各个方向上均匀地散射）





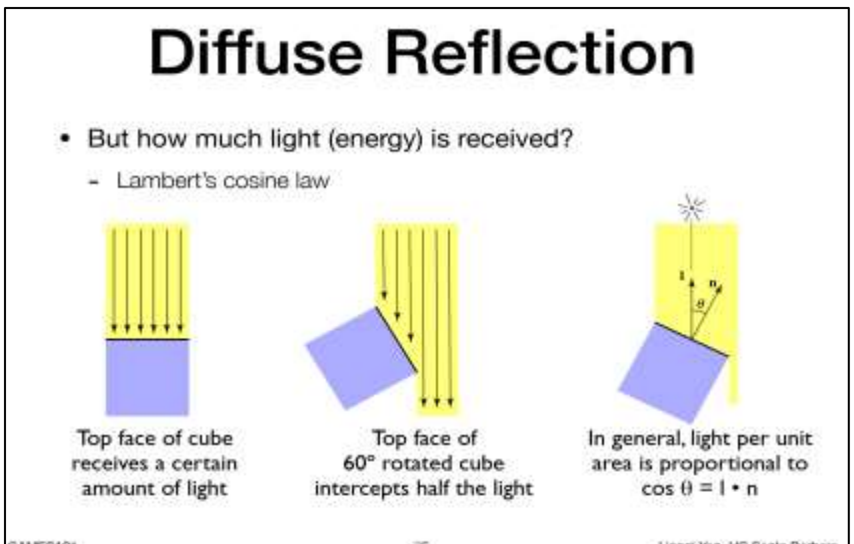
diffuse
reflection light

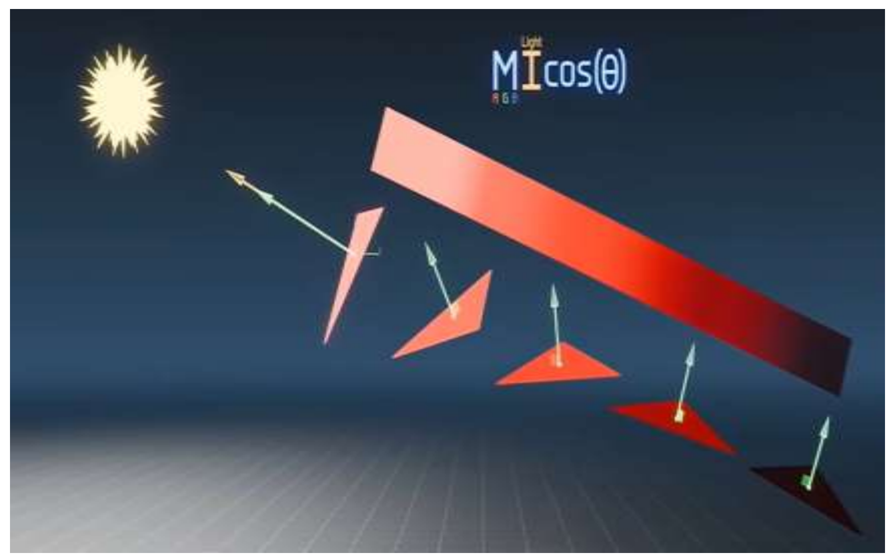
The University of New Mexico

Phong Model（cont.)

➢Diffuse Reflection Light（漫反射光）(cont.)

- ➢Amount of light reflected is proportional to the vertical component of incoming light **(“反射光光强” 与 “入射光的垂直分量”成 正比**
- ➢**入光线方向l和表面点的法向量n的夹角越小，则漫反射光越强，表示为**
  - ➢$\cos\theta_i = \mathbf{l}\cdot\mathbf{n}$ // if vectors normalized！
- ➢漫反射光光强与观察者方向V无关





Phong Model（cont.)

➢Diffuse Reflection Light（漫反射光）(cont.)

$L_d = I_d\, k_d * \cos\theta_i \;=\; k_d\, I_d\, (l \cdot n)$ （ Lambertian Model）

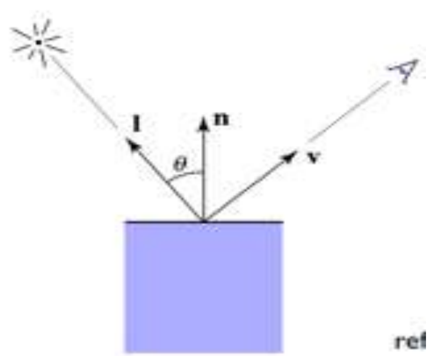
L_d：计算输出的漫反射光 $L_d\ (L_r, L_b, L_g)$

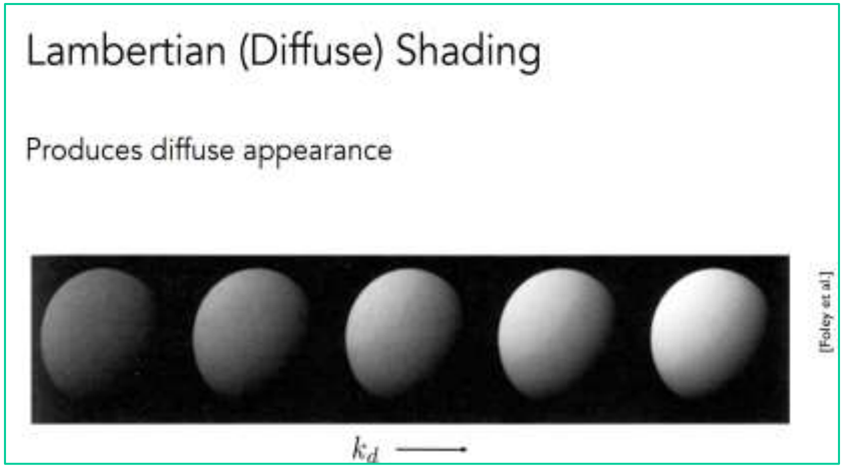
I_d：模拟的入射光中的漫反射分量（可就是入射光） $I_d(I_r, I_b, I_g)$

k_d：模拟的物体表面材质对漫反射光的反射率 $k_d(k_r, k_b, k_g)$

l: 单位化的入射光线的方向 （注：方向朝着光源）

n: 单位化的着色点的外法向量

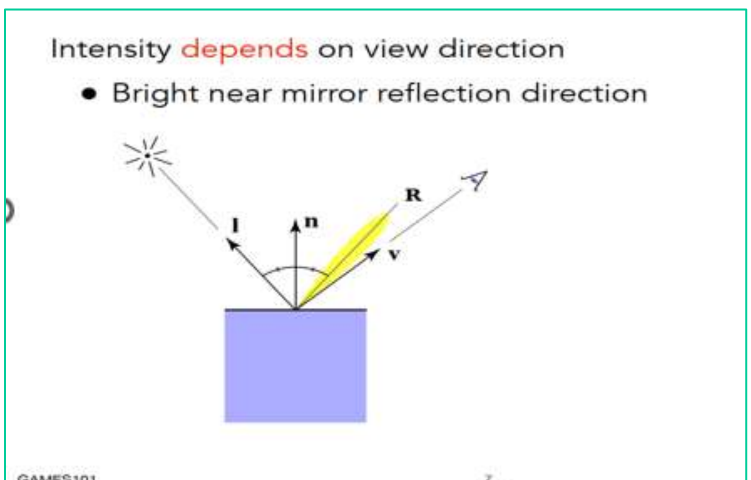


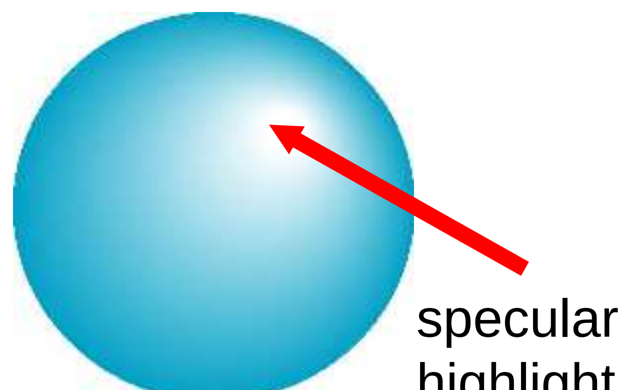


Lighting Model(cont.)

➢Specular Reflection Light（镜面反射光）

- ➢a perfect reflection（理想反射）:
  - ➢Normal is determined by local orientation;（法线由着色点的局部朝向确定）
    - – **Angle of incidence（入射角）= angle of reflection（反射角）**
    - – **The three vectors l, n, r must be coplanar**（共面的）
  - ➢**Smooth surfaces show specular highlights（光滑表面镜面亮光）**
    - – **due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection（镜面反射光方向集中在靠近理想反射方向的区域)**





Phong Model（cont.)

➢Specular Reflection Light（镜面反射光）（cont.)

➢Modeling Specular Reflections

- v和r的夹角ϕ越大， 则亮度值会越小，而且衰减很快，因此采用指数 α（称为高光系数shininess ）来加速衰减

$$L_s = k_s I_s \cos^{\alpha} \phi \quad = k_s I_s (v \cdot r)^{\alpha}$$

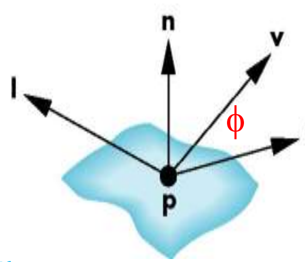
the angle between the viewer v
and the ideal reflection direction r

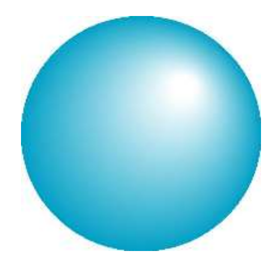
shininess coef.

Reflected intensity

incoming intensity

reflection/absorption coef.

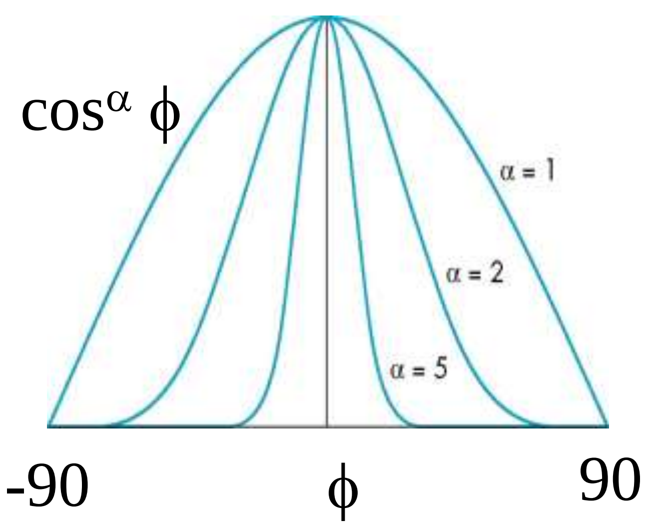


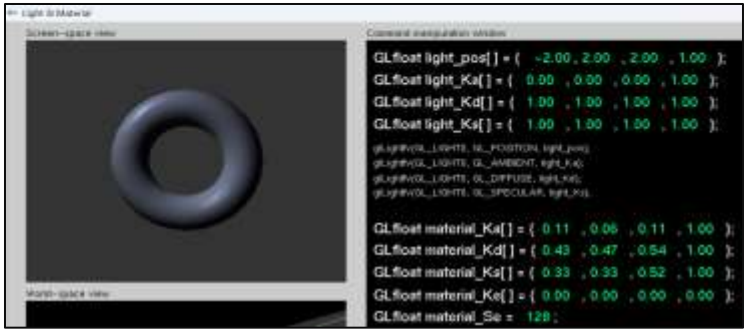


The University of New Mexico

Phong Model(cont.)

- **Specular Reflection Light（镜面反射光）(cont.)**
  - **The Shininess Coefficient α（高光系数）** $L_s = k_s I_s (v \cdot r)^{\alpha}$
    - **表征材质的属性，当材质越光滑则系数越大越抛光，物体表面渲染后的高亮区越小。**
    - Values between <u>100 and 200</u> correspond to **“metals金属”**
    - Values between <u>5 and 10</u> give surface that look like **“plastic塑料”**



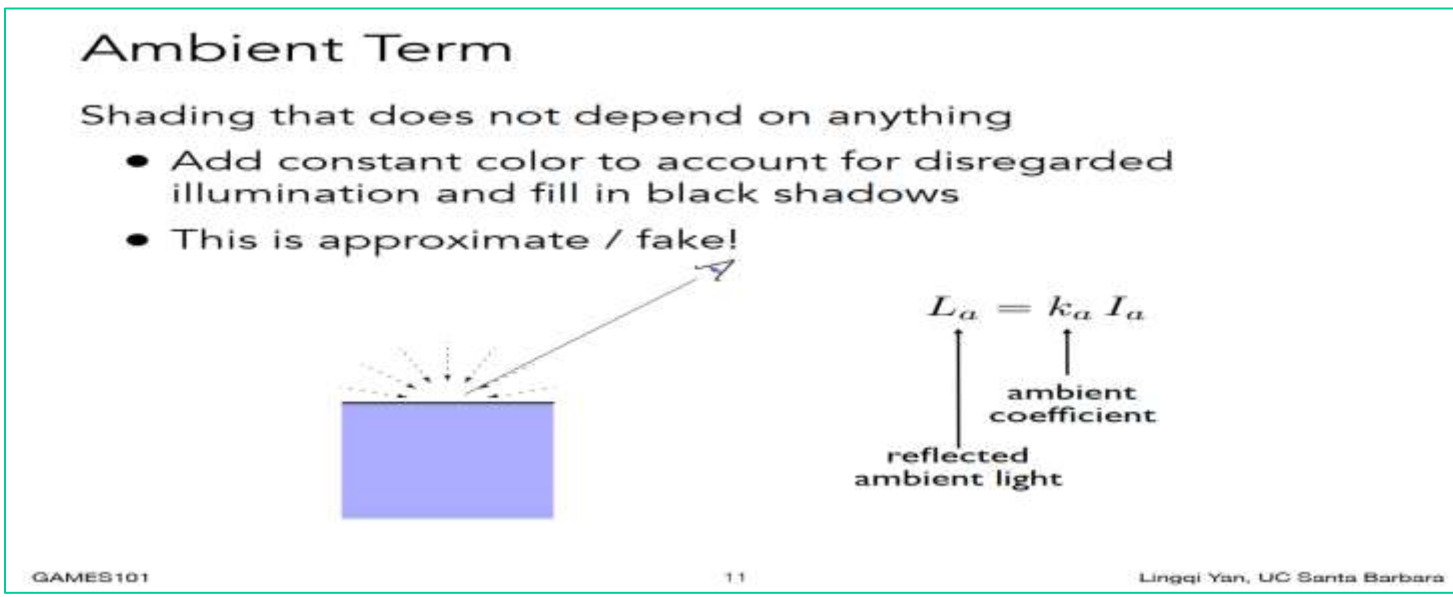


Phong Model（cont.)

➢Ambient Light Model（环境反射光）

- Ambient light is the result of multiple interactions between (large) light sources and the objects in the environment
- Amount and color depend on both the color of the light(s) and the material properties of the object

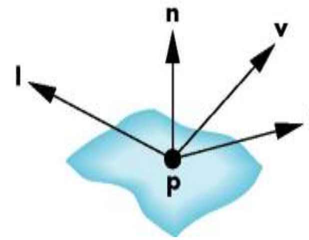
➢$L_a = k_a I_a$ （环境反射光=环境光反射系数 * 环境入射光）



Phong Model（cont.)

➢**Phong model将三种反射分量加起来:**

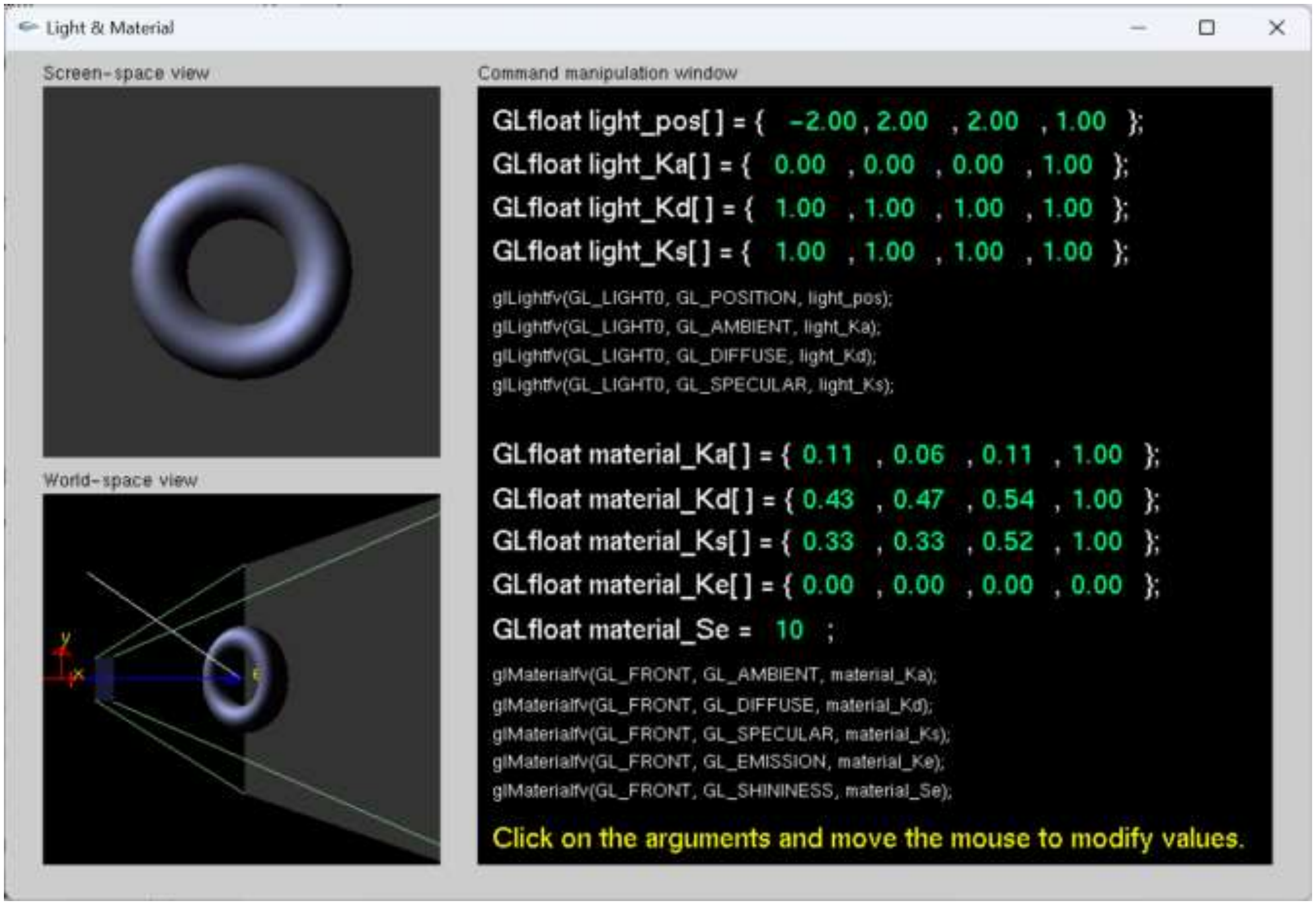
$$L = k_a I_a + k_d I_d \; l \cdot n + k_s I_s (v \cdot r)^{\alpha}$$



- l, n, v, r vectors are variable vectors based on the shading point.
- Constant Coefficients:
  - For each color component , separate red, green and blue components Hence, 9 intensity coefficients for each point source
    - I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}
  - Material properties match light source properties, 9 absorbtion coefficients
    - k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}
  - Only one Shininess coefficient
    - a

Phong Model(cont.)

- 展示简单光照模型



Outline

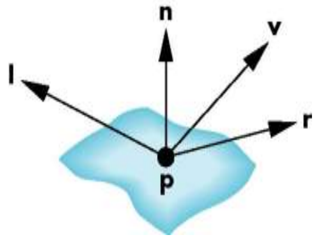
- **Lighting Models（光照模型）**
  - Phong Model
  - Blinn-Phong Model
- **Surface Shading Methods（着色方法）**
  - Flat shading: per face using lighting model
  - Gouraud shading: per vertex using lighting model
  - Phong shading: per fragment using lighting model
- **Shading Realization in Rasterization Pipeline**
  - Flat shading: mainly vertex shader program
  - Gouraud shading: mainly vertex shader program
  - Phong shading: both vertex and fragment shader program

Blinn-Phong Model

➢Why Blinn-Phong?

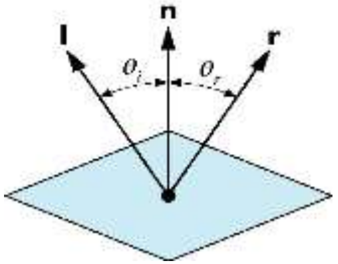
- The specular term in the Phong model is problematic , because it requires the calculation of a new "reflection vector r " and "view vector v " for each vertex

  - $L = k_a I_a + k_d I_d \; l \cdot n + k_s I_s (v \cdot r)^{\alpha}$



➢根据Ideal Reflector理想反射器公式进行r的计算太复杂:

$r = 2\,(l \cdot n)\,n - l$



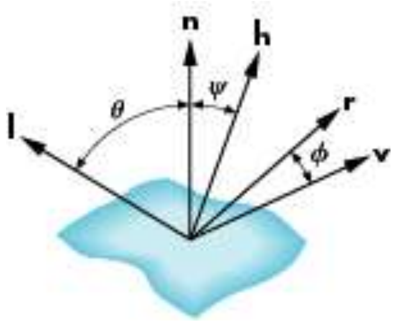
The University of New Mexico

Blinn-Phong Model(cont.)

- **Why Blinn-Phong?(cont.)**
  - **Blinn suggested an approximation using “ the halfway vector h ” that is more efficient**
    - the halfway angle Ψ is half of angle ϕ which between **r** and **v** if vectors are coplanar.
    - The Halfway Vector **h** is normalized vector halfway between **l** and **v;**

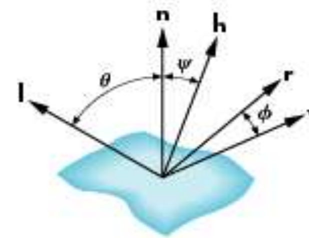
h = (l + v)/ | l + v |

$$\psi = \phi/2$$



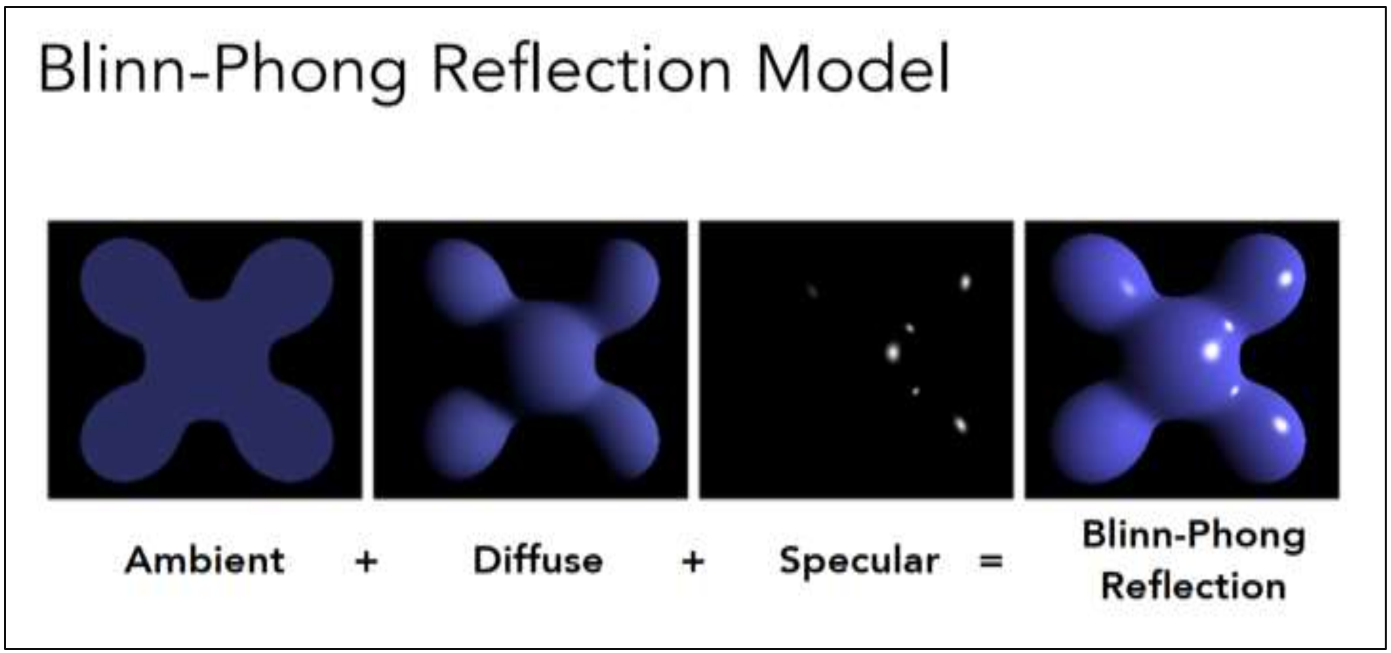
  - Replace $(v \cdot r)^{\alpha}$ by $(n \cdot h)^{\beta}$, and β is chosen to match shininess

Blinn-Phong Model(cont.)



- Blinn-Phong Model

$$\mathbf{L = La + Ld + Ls = k_a\, I_a + k_d\, I_d\ \ l \cdot n + k_s\, I_s\, (n \cdot h\,)^{\beta}}$$



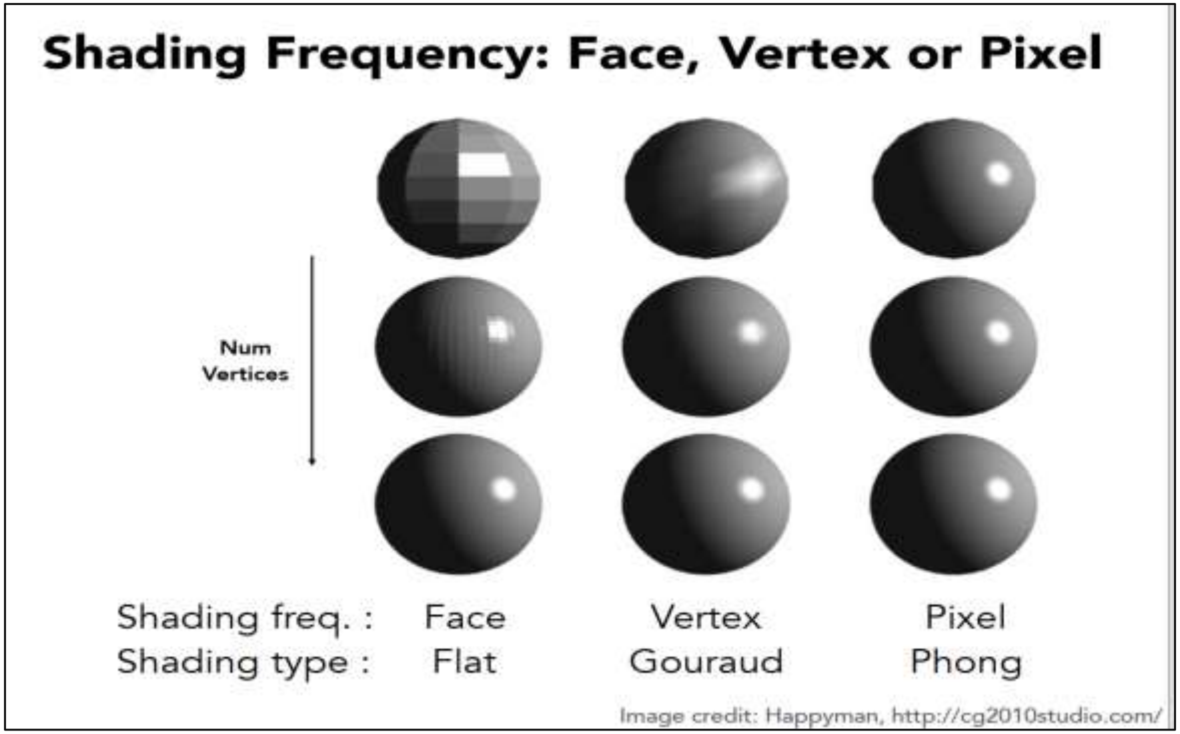
Outline

- **Lighting Models（光照模型）**
  - Phong Model
  - Blinn-Phong Model
- **Surface Shading Methods（着色方法）**
  - Flat shading: per face using lighting model
  - Gouraud shading: per vertex using lighting model
  - Phong shading: per fragment using lighting model
- **Shading Realization in Rasterization Pipeline**
  - Flat shading: mainly vertex shader program
  - Gouraud shading: mainly vertex shader program
  - Phong shading: both vertex and fragment shader program

Surface Shading Methods

➢对于多边形(三角形)表面主要有三种着色方法

1. Flat Shading（平面着色）per face using lighting model
2. Gouraud Shading（高洛德着色）per vertex using lighting model
3. Phong shading（冯 着色）per fragment using lighting model



Surface Shading Methods(cont.)

The University of New Mexico

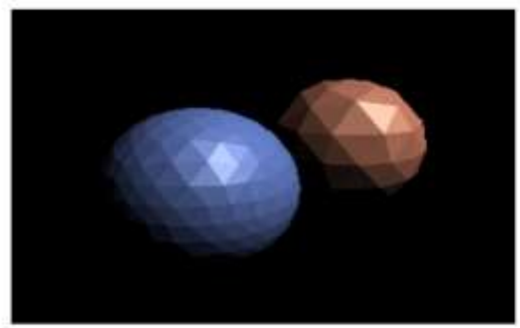
➢1.Flat Shading

- One color Per Triangle Surface,光照模型计算颜色
- 如Blinn-Phong model：$L = La + Ld + Ls = k_a I_a + k_d I_d\ l \cdot n + k_s I_s (n \cdot h)^{\beta}$

Shade each triangle (flat shading)

Flat shading

- Triangle face is flat — one normal vector
- Not good for smooth surfaces



GAMES101 16 Lingqi Yan, UC Santa Barbara

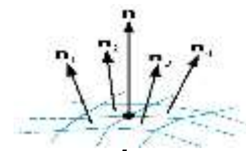
The University of New Mexico

Surface Shading Methods(cont.)

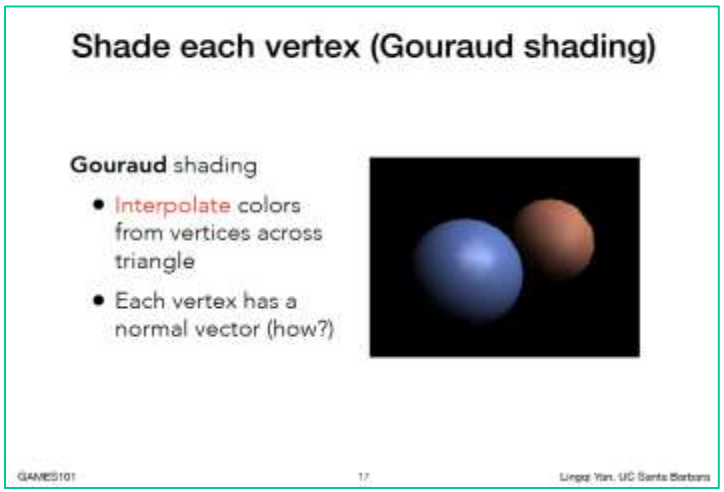
➢2. Gouraud Shading / Smooth Shading

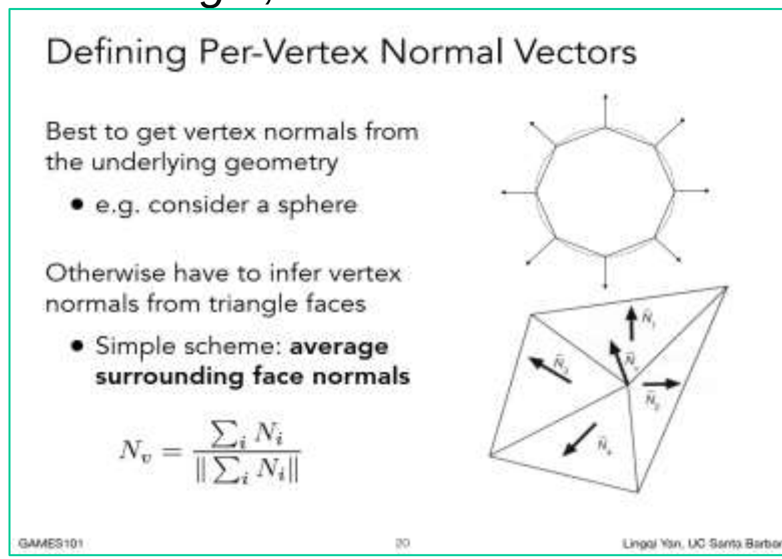
Step1: set a new normal at each vertex

- ***General Mesh: eg:*** $\boldsymbol{n} = (\boldsymbol{n}_1+\boldsymbol{n}_2+\boldsymbol{n}_3+\boldsymbol{n}_4)/\,|\boldsymbol{n}_1+\boldsymbol{n}_2+\boldsymbol{n}_3+\boldsymbol{n}_4|$



- *Sphere model: for if it centered at origin, each vertex normal* $\mathbf{n} = \mathbf{p}$





Step2: Apply Lighting Model at each vertex to get color

- 如Blinn-Phong model： $\mathbf{L = La + Ld + Ls = k_a\, I_a + k_d\, I_d\ \ l \cdot n + k_s\, I_s\, (n \cdot h)^{\beta}}$

Step3: Interpolate “colors from vertices” across triangle!(见下页)

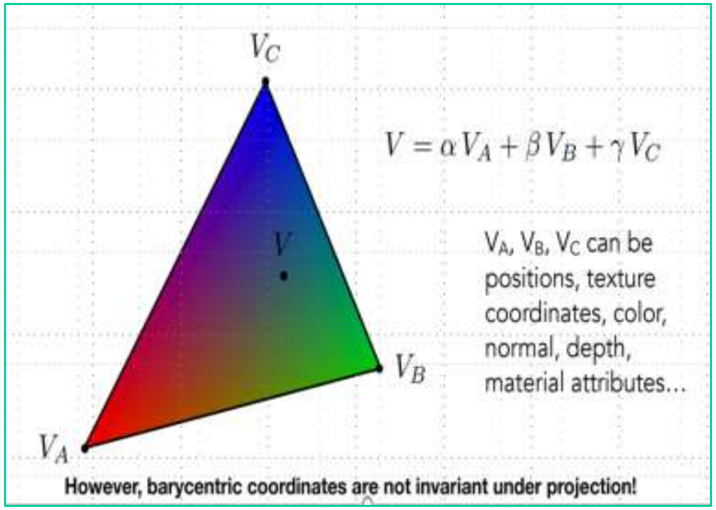
Polygonal Shading(cont.)

- 2. Gouraud Shading / Smooth Shading（cont.)

step3.Color Interpolation:

Linearly Interpolate Values at Vertices using Barycentric Coordinates(采用重心坐标进行线性插值)

➢ $C= \alpha\ C_A + \beta\ C_B + \gamma\ C_c$



Surface Shading Methods(cont.)

The University of New Mexico

➢3.Phong shading

More smooth shading , but more computation !

Step1: Set **a normal** at each vertex（*like gauroud shading)*

Step2: Interpolate “normal vector from vertices” across each triangle(见下页)

Step3: Apply Lighting Model at each fragment to get color

- **如**Blinn-Phong model：$\mathbf{L = La + Ld + Ls = k_a\, I_a + k_d\, I_d\ \ l \cdot n + k_s\, I_s\, (n \cdot h)^{\beta}}$

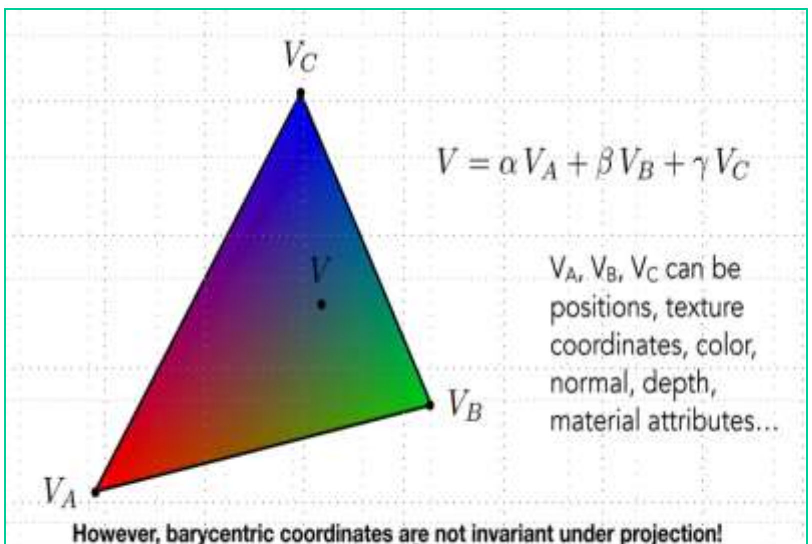
Polygonal Shading(cont.)

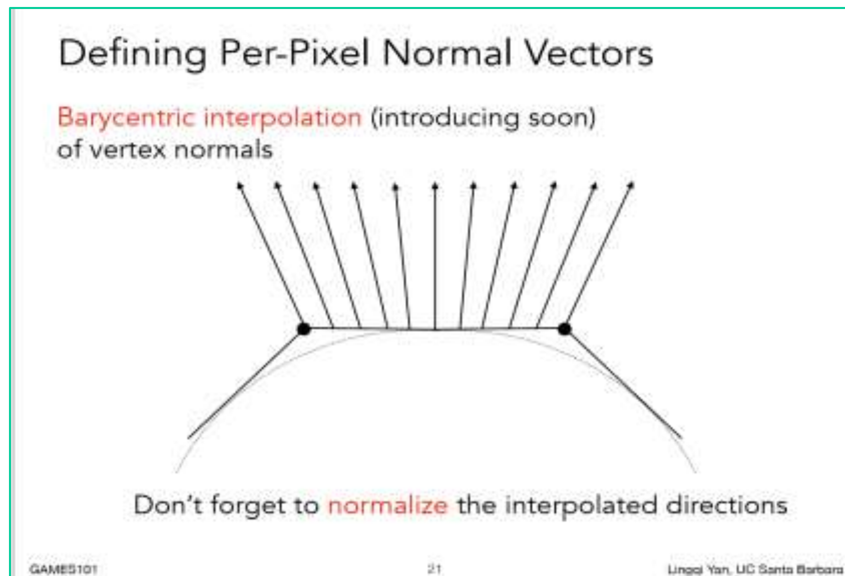
- 3.Phong shading（cont.)

Step2: Normal Interpolation:

Linearly Interpolate Values at Vertices using Barycentric Coordinates(采用重心坐标进行线性插值）

- $N = \alpha N_A + \beta N_B + \gamma N_c$





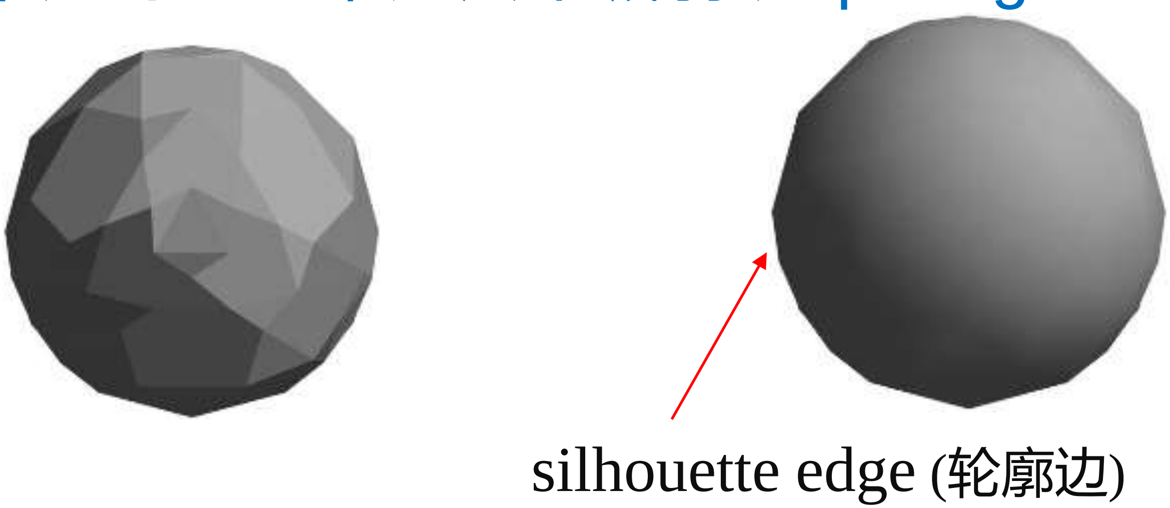
Surface Shading Methods(cont.)

- Comparation **Gourand shading** and **Phong Shading**
  - Both need data structures to represent meshes so we can obtain vertex normal
  - Phong shading may look more smooth while Gouraud shading may show edges(马赫带效应~参见下页）,
  - Phong shading requires much more work than Gouraud shading
    - *Until recently not available in real time systems,*
    - *Now can be done using fragment shaders*

Gourand shading(Per Vertex) **Phong Shading(Per Fragment)**

Surface Shading Methods(cont.)

- Mach band（马赫带效应）
  - 人类视觉系统具有“侧抑制(lateral inhibitioin)性质”：在亮度变化的边界，在较亮一侧看见一条更亮的线，在较暗一侧看一条更暗的线。
  - 没有办法能够避免，但可通过三角形细分使其减弱，或采用好的渲染方法来减弱(如phong shading).

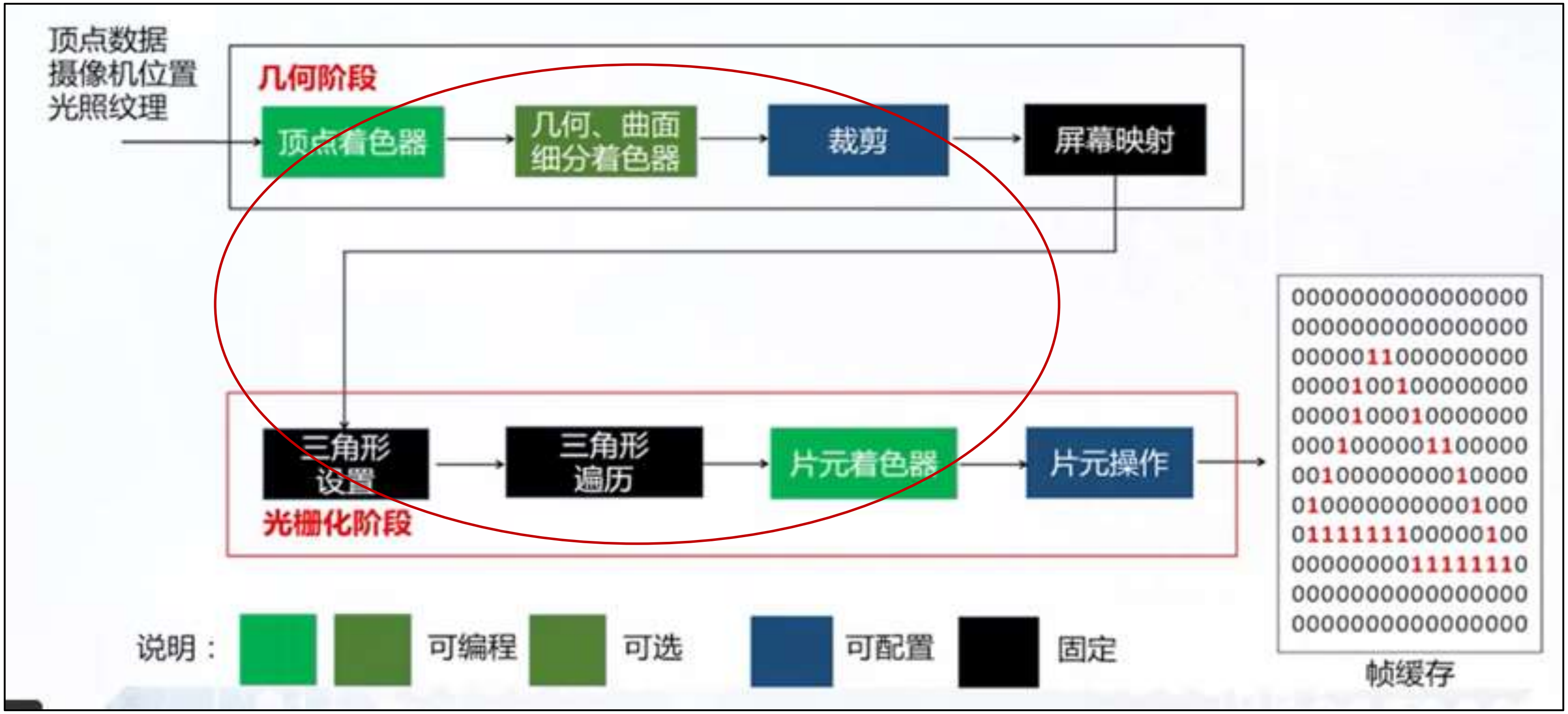


Outline

- **Lighting Models（光照模型）**
  - Phong Model
  - Blinn-Phong Model
- **Surface Shading Methods（着色方法）**
  - Flat shading: per face using lighting model
  - Gouraud shading: per vertex using lighting model
  - Phong shading: per fragment using lighting model
- **Shading Realization in Pipeline**
  - Flat shading: mainly vertex shader program
  - Gouraud shading: mainly vertex shader program
  - Phong shading: both vertex and fragment shader program

Shading Realization

- 光栅化管线中实现光照着色渲染：
  - 涉及诸多模块共同完成



The University of New Mexico

Shading Realization(cont.)

- Flat恒定强度明暗处理(Flat Surface Rendering)
  - 顶点着色器中, 对三角面片的一个顶点用光照模型计算颜色, 作为整个三角面片表面的颜色
- Gouraud明暗处理(Gouraud surface rendering)
  - 在顶点着色器中, 对三角面片的每顶点用光照模型计算颜色,
  - 光栅化时, 各片元的颜色是由三角形顶点的颜色插值得到。
- Phong明暗处理(Phong surface rendering)
  - 在顶点着色器中, 计算各项参数并传递输出给管线
  - 在片元着色器中, 对每片元用光照模型计算颜色。

//注意:这里的光照模型采用后页面介绍的

//Blinn-Phong Model with Distance Terms and Non-Negative

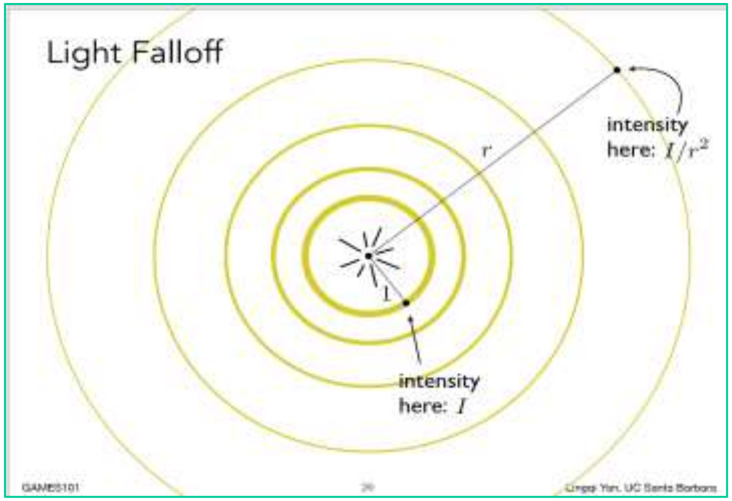
Shading Realization(cont.)

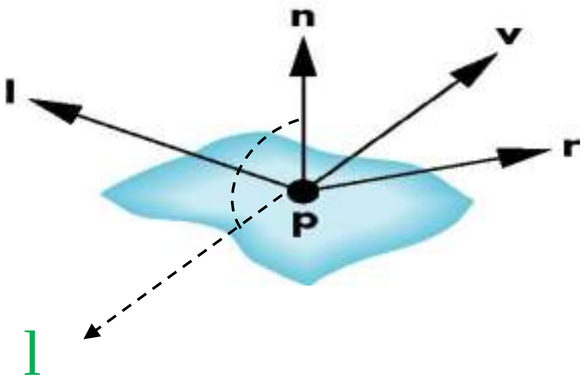
The University of New Mexico

- **Blinn-Phong Model with Distance Terms and Non-Negative**
  - 考略了“距离衰减”和“非背面入射（即非负）”，漫反射光计算模型 Ld = kd Id（l · n）可转换为：

$$\boxed{L_d = f(r)\, I_d\ k_d\, max(0, l \cdot n\,)}$$

- 注意：一般f(r)= 1 /r2　或采用 f(r)=1 /(a+b*r+c*r2)；注：图中I/r2 = f(r) Id
- 注：因为是衰减因子，如果f(r)值应该小于1，所以，f(r)=min(1,1/ /r2)





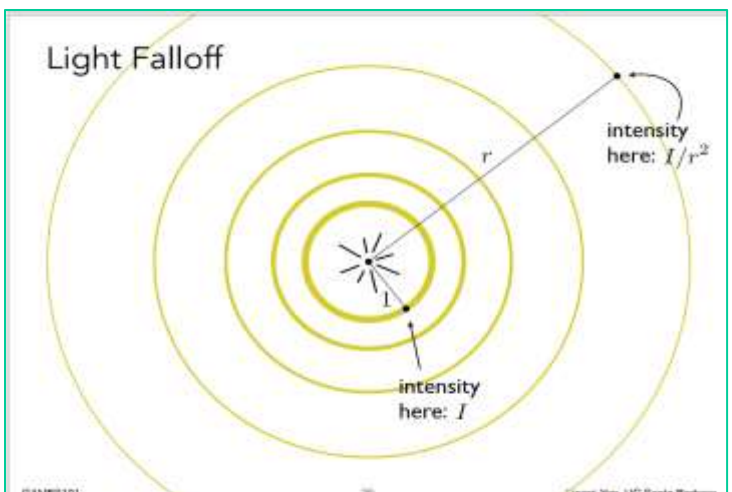
The University of New Mexico

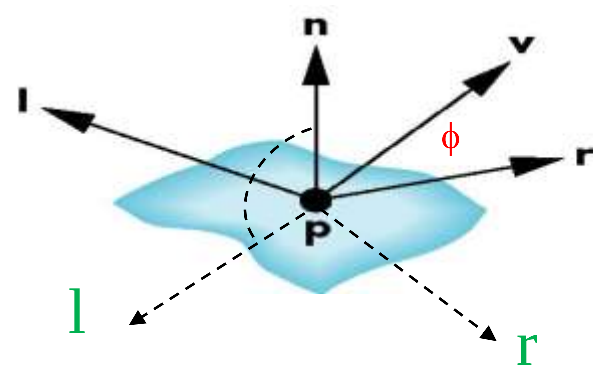
Shading Realization(cont.)

- Blinn-Phong Model with Distance Terms and Non-Negative(cont.)
  - 考略了“距离衰减”和“非负”，镜面反射光的计算模型$L_s = k_s\ I_s\ (v \cdot r)^{\alpha}$ 可转换为：

$$\boxed{L_s = f(r)\ k_s\ I_s\ exp\ (max(0, v \cdot r),\alpha)}$$

- 注意：一般 **f(r)= 1** $/r^2$ 或采用 **f**(r)=1/(a+b*r+c*r^2) 注：图中$I/r^2 = f(r)\ I_s$
- 注：因为是衰减因子，如果f(r)值应该小于1，所以，f(r)=min(1,1/ /r^2)



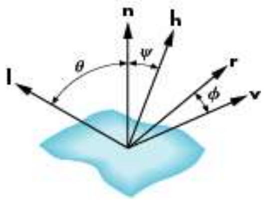


Shading Realization(cont.)

➢Blinn-Phong Model with Distance Terms and Non-Negative(cont.)

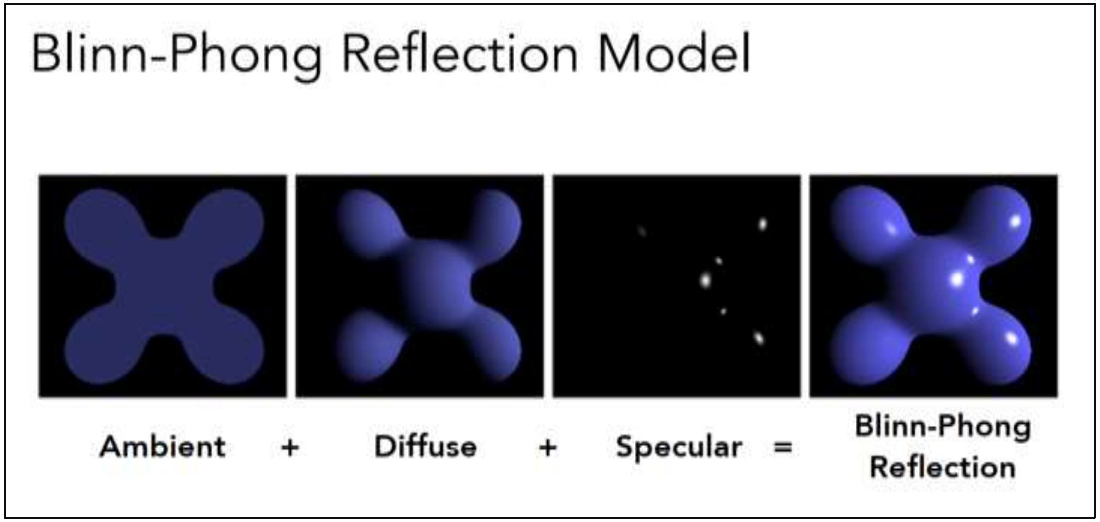
- 最后的光照模型计算公式如下：

$$\boxed{\mathrm{L} = f(d)\left[k_d \mathrm{I}_d \max(\vec{L} \bullet \vec{N}, 0) + k_s \mathrm{I}_s \max((\vec{N} \bullet \vec{H})^{\mathrm{e}}, 0)\right] + k_a \mathrm{I}_a}$$



//注：f(d)= $1/d^2$ or f(d)=min（1，$1/(a+bd+cd^2)$）；主要对入射光进行衰减

//实现用：f(d)= min(1, $1/d^2$) or f(d)=min（1，$1/(a+bd+cd^2)$），因为是衰减因子，其值不能大于1



Shading Realization(cont.)

➢Examples //ref: AngelCode /06/ShadedSphere*

- shadingshadedSphere1: shaded sphere using true normals and per vertex lighting
- shadingshadedSphere2: shaded sphere using true normals and per fragment lighting
- shadingshadedSphere3: shaded sphere using vertex normals and per vertex lighting
- shadingshadedSphere4: shaded sphere using vertex normals and per fragment lighting

	每顶点着色 Gouraud Shading	每片元着色 Phong shading
Using true normals **n=p**	 Shadedsphere1	 ShadedSphere2
using vertex normals	 ShadedSphere3	 ShadedSphere4

Shading Realization(cont.)

- 结合ShadedSphere3代码，讲解实现 Gouraud Shading
  - var lightPosition = vec4(1.0, 1.0, 1.0, 0.0);   //采用平行光
  - 在顶点着色器中应用光照模型，计算每个顶点的颜色，后续交给光栅化模块进行重心坐标插值得到每片元颜色

```
main()
{
    vec3 pos = (uModelViewMatrix * aPosition).xyz;
    vec3 light = uLightPosition.xyz;
    vec3 L;

    // check for directional light
    if(uLightPosition.w == 0.0) L = normalize(uLightPosition.xyz);
    else L = normalize(uLightPosition.xyz - pos);
    vec3 E = -normalize(pos);
    vec3 H = normalize(L + E);

    // Transform vertex normal into eye coordinates
    vec3 N = normalize(uNormalMatrix*aNormal.xyz);

    // Compute terms in the illumination equation
    vec4 ambient = uAmbientProduct;

    float Kd = max( dot(L, N), 0.0 );
    vec4  diffuse = Kd*uDiffuseProduct;

    float Ks = pow( max(dot(N, H), 0.0), uShininess );
    vec4  specular = Ks * uSpecularProduct;

    if( dot(L, N) < 0.0 ) {
        specular = vec4(0.0, 0.0, 0.0, 1.0);
    }

    gl_Position = uProjectionMatrix * uModelViewMatrix * aPosition;

    vColor = ambient + diffuse +specular;
    vColor.a = 1.0;
}
```

```
<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es

precision mediump float;

in vec4 vColor;
out vec4 fColor;

void
main()
{

    fColor = vColor;

}
</script>
```

The University of New Mexico

Shading Realization(cont.)

- 结合ShadedSphere4代码，讲解实现 Phong Shading
  - var lightPosition = vec4(1.0, 1.0, 1.0, 0.0);　//采用平行光
  - 在顶点着色器中,计算每个顶点的相关向量N,L,E ，再out给光栅化模块进行插值
  - 在片元着色器中,根据in光栅化插值得到的每片元的N,L,E,应用光照模型计算每片元的颜色

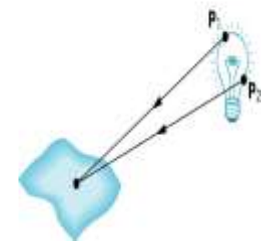
```
<script id="vertex-shader" type="x-shader/x-vertex">
#version 300 es

in vec4 aPosition;
in vec4 aNormal;
out vec3 N, L, E;

uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
uniform vec4 uLightPosition;
uniform mat3 uNormalMatrix;

void main()
{
    vec3 light;
    vec3 pos = (uModelViewMatrix * aPosition).xyz;
    if(uLightPosition.z == 0.0)  L = normalize(uLightPosition.xyz);
    else  L = normalize(uLightPosition).xyz - pos;

    E =  -normalize(pos);
    N = normalize(uNormalMatrix*aNormal.xyz);

    gl_Position = uProjectionMatrix * uModelViewMatrix * aPosition;

}
</script>
```

```
<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es
precision mediump float;

uniform vec4 uAmbientProduct;
uniform vec4 uDiffuseProduct;
uniform vec4 uSpecularProduct;
uniform float uShininess;

in vec3 N, L, E;
out vec4 fColor;

void main()
{
    vec3 H = normalize( L + E );
    vec4 ambient = uAmbientProduct;

    float Kd = max( dot(L, N), 0.0 );
    vec4  diffuse = Kd*uDiffuseProduct;

    float Ks = pow( max(dot(N, H), 0.0), uShininess );
    vec4  specular = Ks * uSpecularProduct;

    if( dot(L, N) < 0.0 ) specular = vec4(0.0, 0.0, 0.0, 1.0);

    fColor = ambient + diffuse +specular;
    fColor.a = 1.0;
}
</script>
```

Shading Realization(cont.)

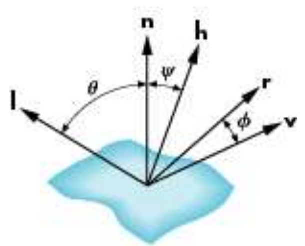
➢Light Sources Type光源类型

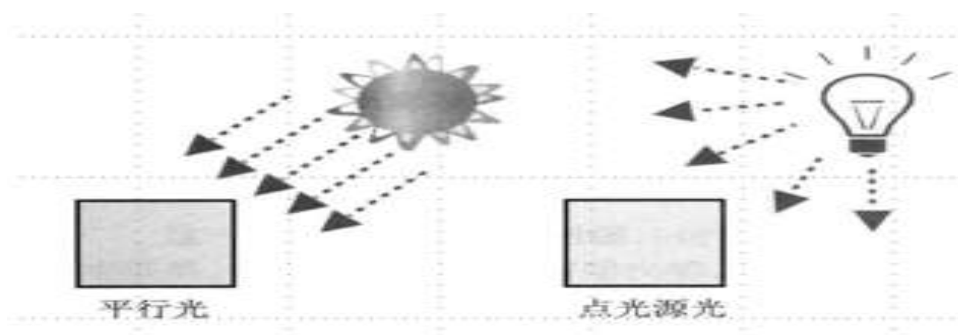
- General Light Sources are difficult to work with , because we must integrate light coming from all points on the source



➢Common Using Simple Light Sources Type

- ➢**Point source Model点光源** with position and color,
- ➢**Parallel source平行光源** is infinite distance away Model with light source direction vector
- ➢I(x,y,z,w)：如果w=1表示点光源，这时(x,y,z)表示点光源的位置
  如果W=0表示平行光，这时(x,y,z)表示入射光的方向





The University of New Mexico

Shading Realization(cont.)

➢法向量变换Normal Transformation

$$I = f(d)\left[k_d L_d \max(\vec{L} \bullet \vec{N}, 0) + k_s L_s \max((\vec{N} \bullet \vec{H})^e, 0)\right] + k_a L_a$$

问题引入：作光照计算如blinn phong模型，需要具有以下参数：

- 标量参数：一般设定好不变，当作常量 uniform
  - 如入射光强分量Ls,Ld,La，对应的反射系数Ks,kd,ka; 和高光系数e等
- 向量参数：一般根据实际情况计算得到，每顶点/每片元不同
  - 根据光源位置，顶点位置，相机位置，需计算出三个单位向量：L,V,N，(半角向量可H根据L,V得到：H=(L+V)/|L+V|;)
  - 入射光向量L=光源位置-顶点位置;
  - 观察方向向量V=相机位置-顶点位置;
  - 物体表面点的法向量N=?

思考：法向量的变换是否等同于物体顶点的“模型变换M”？

The University of New Mexico

Shading Realization(cont.)

➢Normal Transformation法向量变换(cont.)

➢Why 法向量变换矩阵(魔法矩阵)

- 法向量只有方向没有大小，平移变换不会改变，旋转变换和均匀缩放也不会改变法向量，可以直接用模型变换矩阵作用于顶点法向量，但是变换是非均匀的缩放操作时就不正确了，所以还是需要找到更通用正确的法向量变换矩阵
  - **参见ref1《wegGL编程指南》附录E 逆转置矩阵**

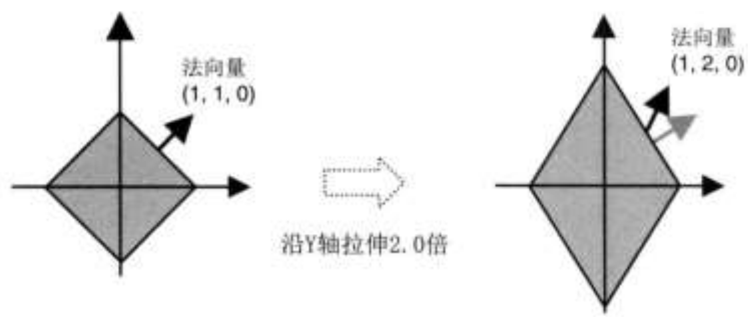


图 E.3 使用模型矩阵子矩阵乘以法向量获得的新向量与缩放后的法向量不同

上图使一个矩形物体（左图）沿 Y 轴拉伸 2.0 倍，拉伸后的形状如右图所示。这里，为了获得变换后的法向量，我们试图使用模型矩阵乘以原先的法向量 (1, 1, 0)。但是，得到的结果是 (1, 2, 0)，并不垂直于拉伸后的物体表面了。

The University of New Mexico

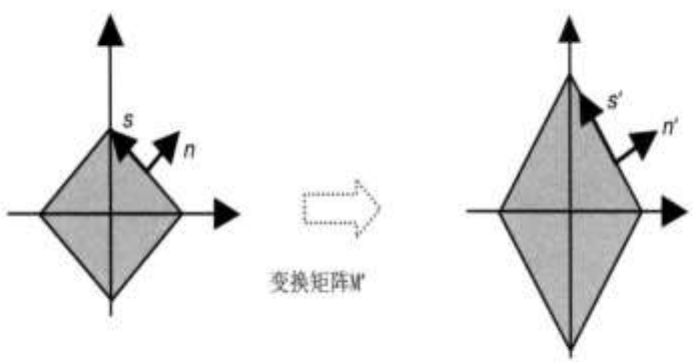
Shading Realization(cont.)

➢Normal Transformation法向量变换(cont.)

➢法向量变换矩阵M’表示为物体变换矩阵M的“**逆转置矩阵**”,即$M'=(M^{-1})^T$

推导思路:找不变量，物体边向量s和向量n是垂直的，变换后仍保持垂直关系，即:

s’ • n’ =0　 , s • n=0　//**参见ref1《wegGL编程指南》附录E 逆转置矩阵**



图E.4 n和s，以及n'和s'的关系

$$s' = M \times s \qquad n' = M' \times n$$

$$(M' \times n) \cdot (M \times s) = 0$$

$$(M' \times n)^T \times (M \times s) = 0 \quad (\text{因为} A.B = A^T \times B)$$

$$n^T \times M'^T \times M \times s = 0 \quad (\text{因为} (A \times B)^T = B^T \times A^T)$$

$$n \cdot s = n^T \times s = 0$$

$$M'^T \times M^T = I$$

$$M' = (M^{-1})^T$$

Summary

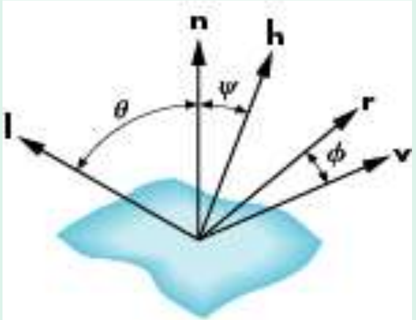
Lighting Models

Phong Model

$$L = k_a I_a + k_d I_d \; l \cdot n + k_s I_s (v \cdot r)^{\alpha}$$

Blinn-Phong Model

$$L = k_a I_a + k_d I_d \; l \cdot n + k_s I_s (n \cdot h)^{\beta}$$

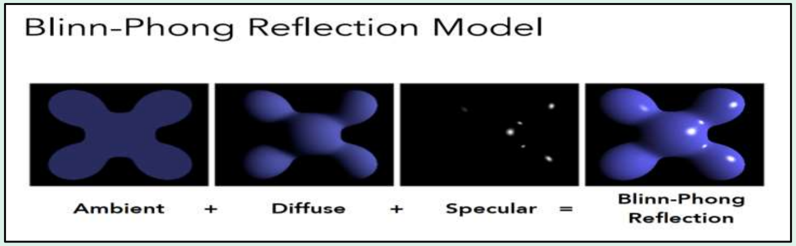


✓ **Blinn-Phong Model with Distance Terms and Non-Negative**

$$\boxed{\mathrm{L} = f(d)\left[k_d \mathrm{I}_d \max(\vec{L} \bullet \vec{N}, 0) + k_s \mathrm{I}_s \max((\vec{N} \bullet \vec{H})^{e}, 0)\right] + k_a \mathrm{I}_a}$$

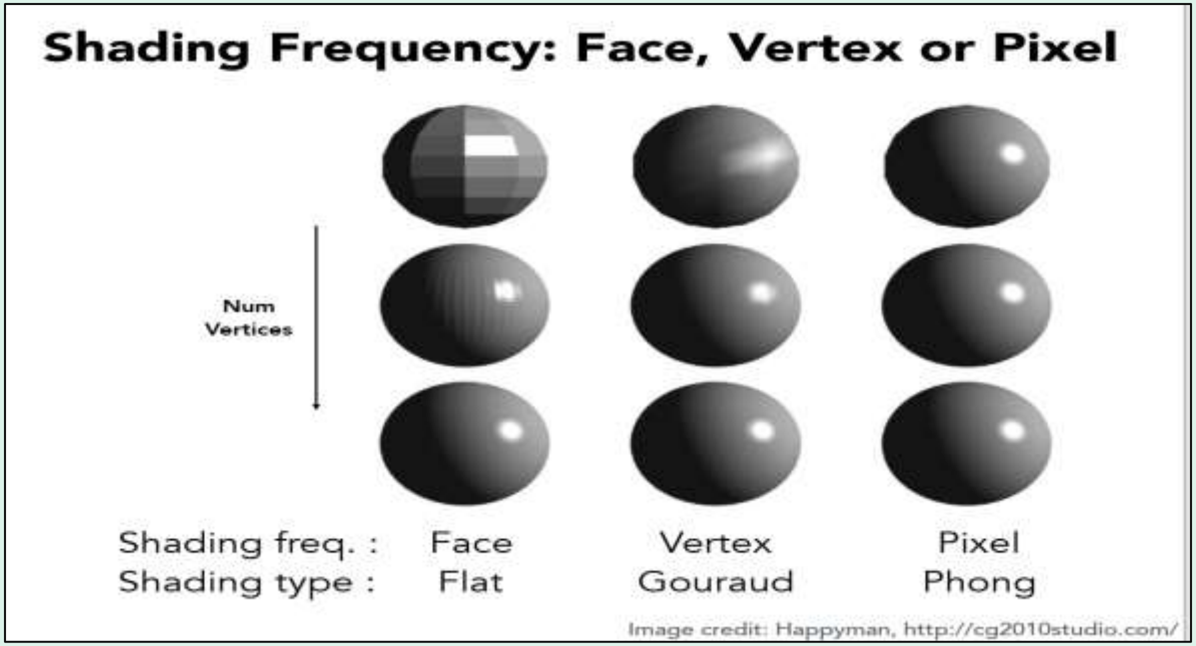
//注： $f(d)= 1/d^2$ or $f(d)=\min(1, 1/(a+bd+cd^2))$ ，一般是对入射光的衰减因子

//实用： $f(d)= \min(1, 1/d^2)$ or $f(d)=\min(1, 1/(a+bd+cd^2))$ 因为是衰减因子，其值不能大于1



Summary(cont.)

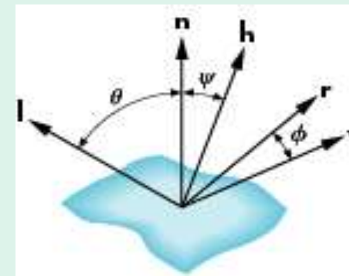
- Shading Methods:
  - Flat shading: per face using lighting model
  - Gouraud shading: per vertex using lighting model
  - Phong shading: per fragment using lighting model



Summary(cont.)

➢实现中需要注意的问题:

- 光照模型中的向量方向
  - 光线方向L 是从物体表面点到光源 Lpos-Ppos
  - 观察方向V是从物体表面点到相机文职 Vpos-Ppos



- 光源表示：l(x,y,z,w)向量的含义
  - ➢如果w=1表示点光源，这时(x,y,z)表示点光源的位置
  - ➢如果W=0表示平行光，这时(x,y,z)表示入射光的方向

- 法向量变换矩阵M'
  - 采用模型变换矩阵M的“**<u>逆转置矩阵</u>**”，即$M'=(M^{-1})^T$

Summary (cont.)

➢Shading Merits and Demerits

- ➢优点：简单快速，适合光栅化管线实现，适合实时要求高的应用
- ➢缺点：采用局部local光照模型 ,真实感比全局global光照模型弱
  - ➢ Only one scattering of light on the surface of the object is considered（Direct illumination）

  只考虑光线从光源发出，直接在物体表面的一次散射后进入观察者视线。
  - ➢ No shadows will be generated

  因为没有考虑物体之间的光线弹射，光线不会被物体遮挡，所以不会产生阴影效果。