Illumination and Shading

Jian Huang, CS456

also edited by Scott Simmerman

Illumination Vs. Shading

- Illumination (lighting) model: determine the color of a surface point by simulating some light attributes.

- Shading model: applies the illumination models at a set of points and colors the whole image.

Illumination Vs. Shading

- Illumination (lighting) model: determine the color of a surface point by simulating some light attributes.
  - Per vertex
  - Consider geometry, light sources, material properties
- Shading model: applies the illumination models at a set of points and colors the whole image.
  - Per fragment
  - Consider interpolation method

"…many of the illumination and shading models traditionally used in computer graphics include a multitude of kludges, 'hacks', and simplifications that have no firm grounding in theory, but that work well in practice."

"It should be mentioned that this sort of lighting model is not based on much physical theory, but the result is still fairly good and relatively easy to control."

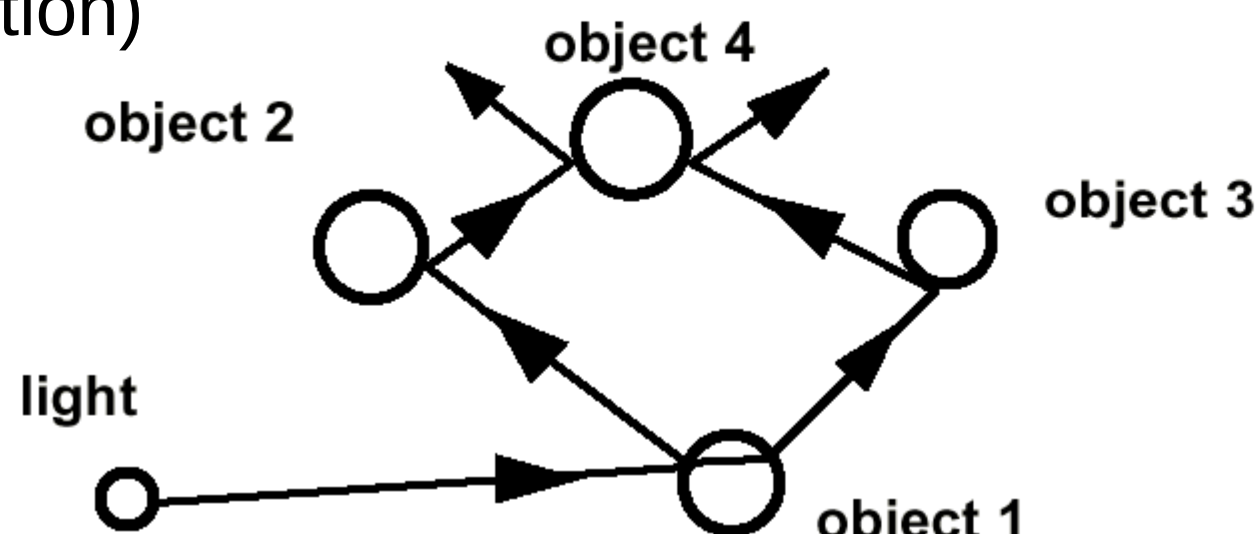
"All of the shading…seems like enormous hacks. Is this true? Yes. However, they are carefully designed hacks that have proven useful in practice."

Illumination (Lighting) Model

- To model the interaction of light with surfaces to determine the final color & brightness of the surface
  - Global illumination
  - Local illumination

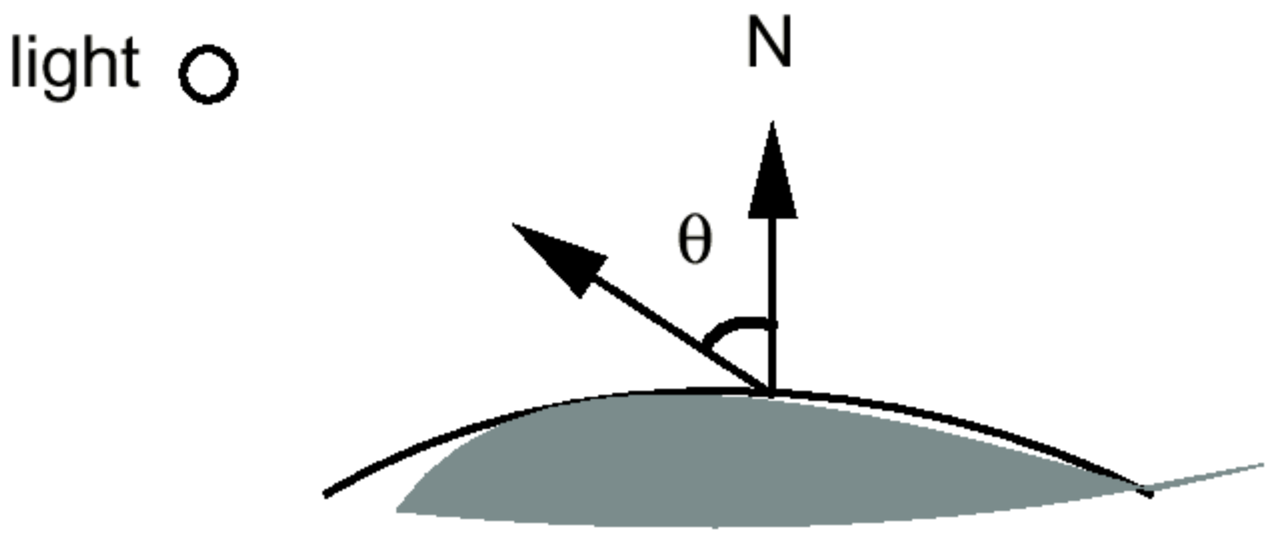
Global Illumination

- Global Illumination models: take into account the interaction of light from all the surfaces in the scene. (will cover under the Radiosity section)



Local Illumination

- Only consider the light, the observer position, and the object material properties



Basic Illumination Model

- Simple and fast method for calculating surface intensity at a given point
- Lighting calculations are based on:
  - The background lighting conditions
  - The light source specification: color, position
  - Optical properties of surfaces:
    - Glossy OR matte
    - Opaque OR transparent (control refection and absorption)

Ambient light (background light)

- The light that is the result from the light reflecting off other surfaces in the environment
- A general level of brightness for a scene that is independent of the light positions or surface directions
- Has no direction
- Each light source has an ambient light contribution, I_a
- For a given surface, we can specify how much ambient light the surface can reflect using an ambient reflection coefficient : K_a $(0 < K_a < 1)$

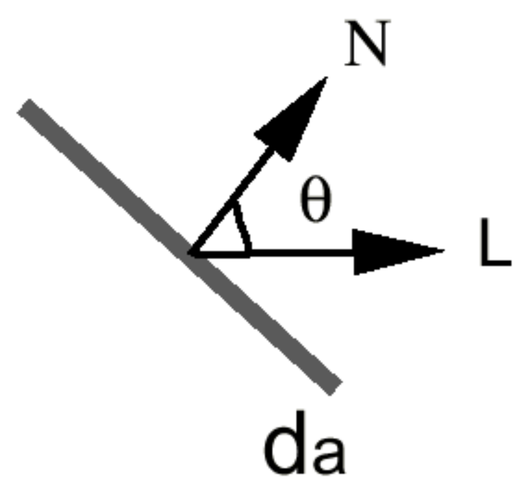
Ambient Light

- So the amount of light that the surface reflects is therefore:

$$I_{amb} = K_a I_a$$

Diffuse Light

- The illumination that a surface receives from a light source and reflects equally in all directions
- Diffuse reflection (or *Lambertian Reflection*) is exhibited by dull, matte surfaces (e.g. chalk, unfinished wood, carpet).
- The brightness depends only on the angle θ between the light direction and surface normal.



Does viewing direction matter?

- Lambertian surfaces have the property (Lambert's Law) that the amount of light reflected toward viewer is *inversely proportional* to the angle between line of sight and surface normal.

But…

- Amount of area seen is *directly proportional* to same angle.

Net effect: Amount of light seen is *independent* of viewing direction.

The Diffuse Component

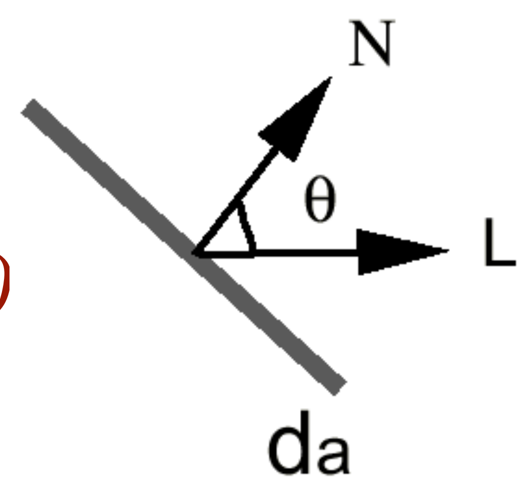
$$I_{diff} = K_d\, I_L \cos(\theta)$$

- K_d $(0<K_d<1)$ is the diffuse reflection coefficient or amount of diffuse light (material property)

- If N and L are normalized, $\cos(\theta) = N \bullet L$

$$I_{diff} = K_d\, I_L\, (N \bullet L)$$

(What if $\theta > 90^o$?)



- Adding ambient and diffuse:

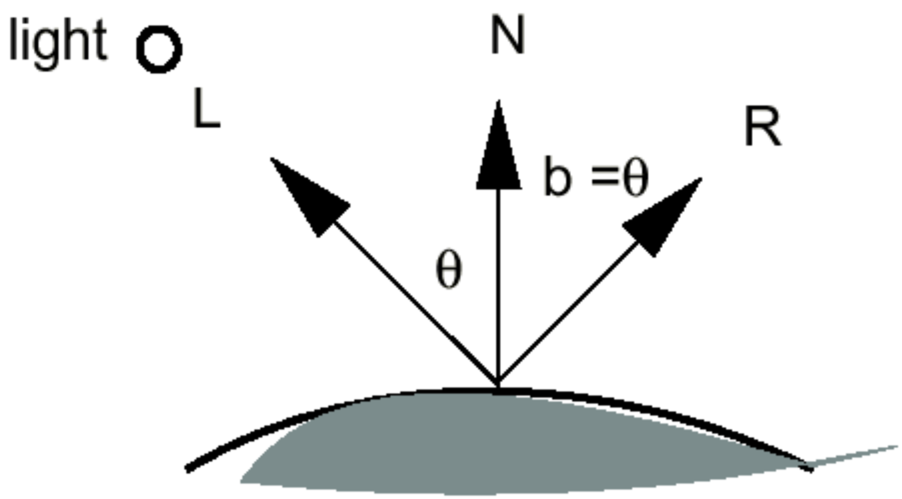
$$I = K_a\, I_a \;+\; K_d\, I_L\, (N \bullet L)$$

Examples

Sphere diffusely lighted from various angles !

Specular Light

- These are the bright spots (specular highlights) on objects (such as polished metal, apple ...)
- Light reflected from the surface unequally to all directions. Has to do with microscopic properties of surface.
- The result of near total reflection of the incident light in a concentrated region around the specular reflection angle



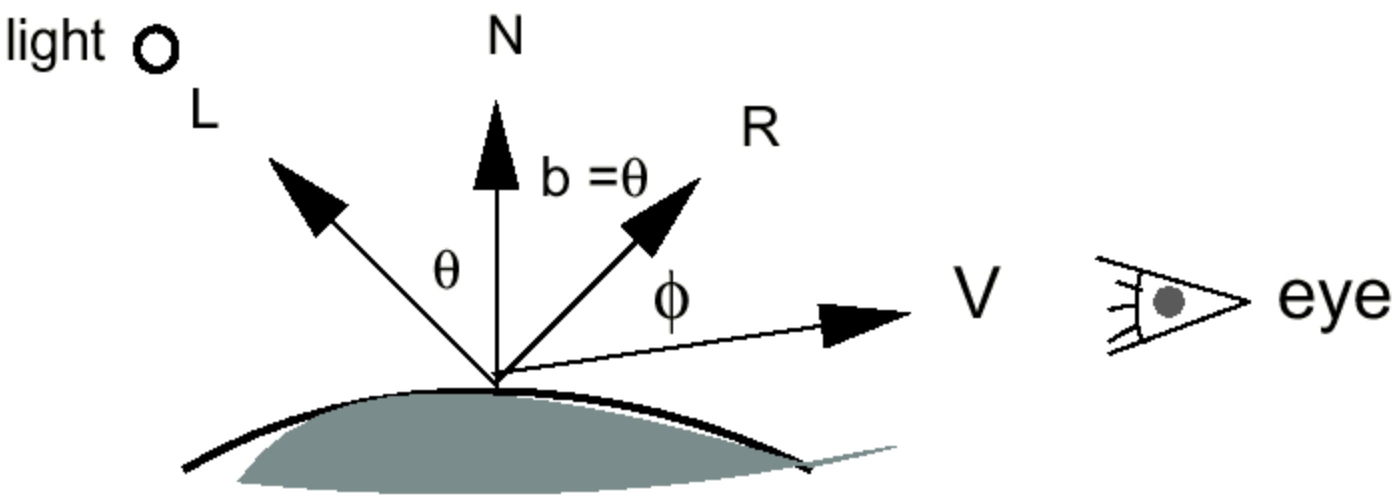
Phong's Model for Specular

- How much reflection light you can see depends on where you are

$$I_{spec} = K_s I_s \cos^n(\phi)$$

K_s is specular reflection coefficient

I_s is specular component of light source

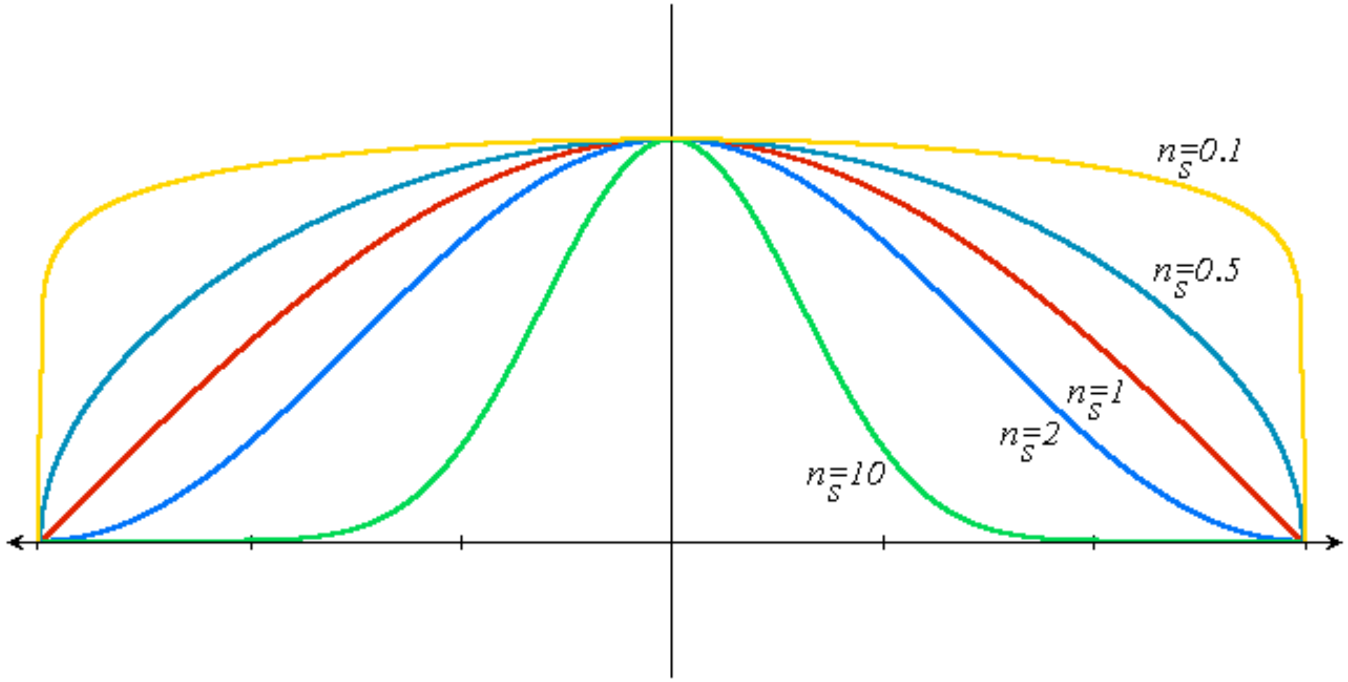


Phong Illumination Curves

Specular exponents are much larger than 1;

Values of 100 are not uncommon.

n : glossiness, rate of falloff

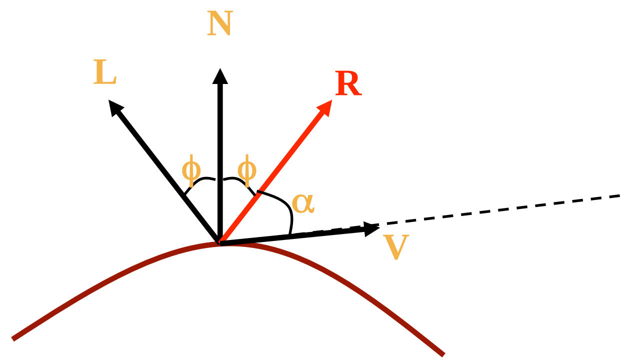


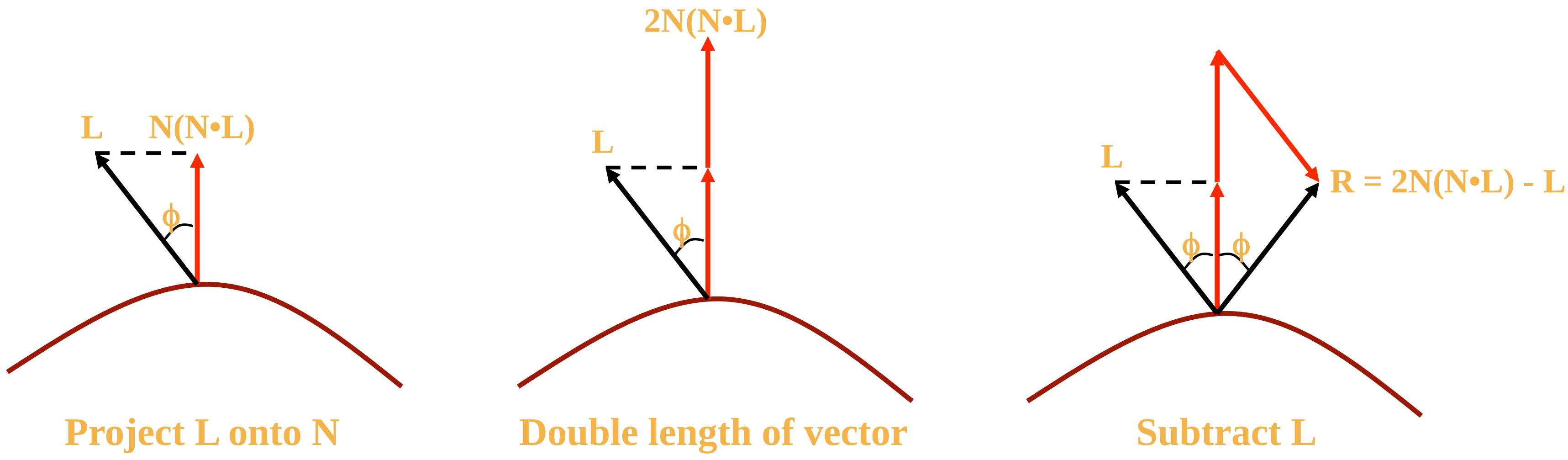
Reflected Ray

How to calculate R?

R + L = 2(N*L) N

R = 2(N*L) N - L



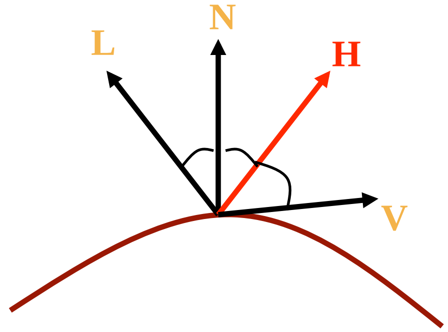


Halfway Vector

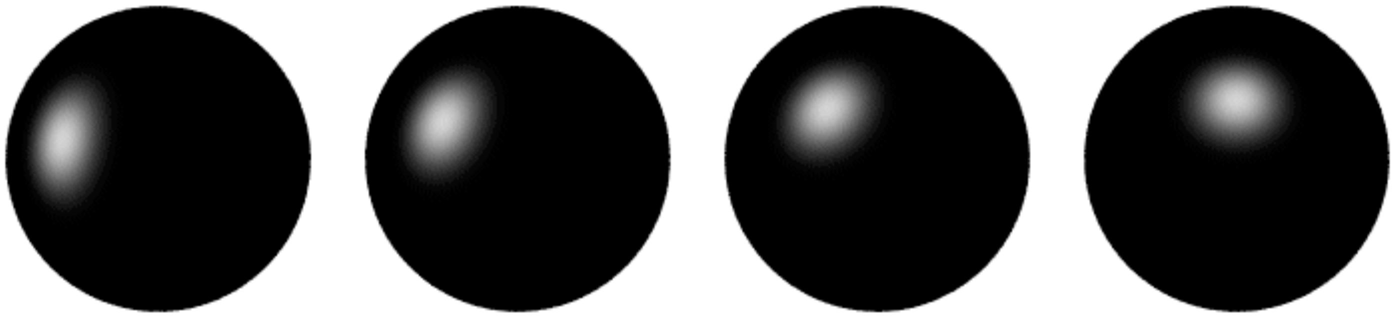
- An alternative way of computing phong lighting is: $I_s = K_s I_s (N \bullet H)^n$

- H (halfway vector): halfway between V and L: (V+L)/2

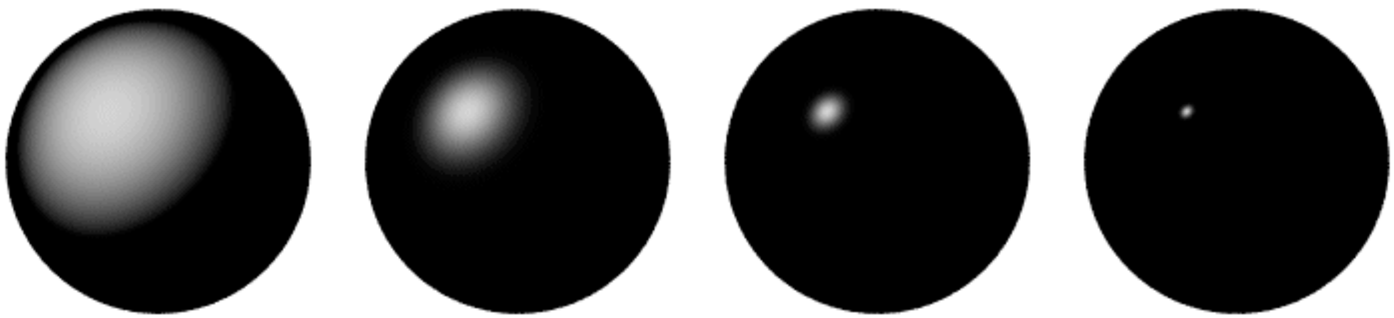
- Fuzzier highlight



Phong Illumination



Moving Light



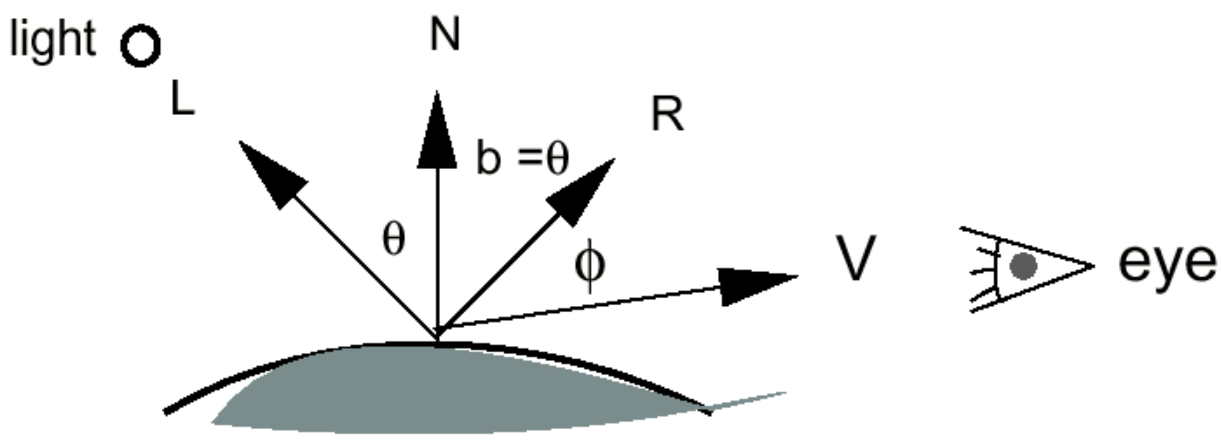
Changing n

Putting It All Together

- Single Light (white light source)

$$I = \quad I_{amb} \quad + \quad I_{diff} \quad + \quad I_{spec}$$

$$I = K_a I_a + K_d I_L (N \bullet L) + K_s I_L (R \bullet V)^n$$



Multiple Light Sources

- For m light sources:

$$I = K_a I_a + \sum_{1 \le i \le m} (K_d I_{Li} (N \bullet L_i) + K_s I_{Li} (R_i \bullet V)^n)$$

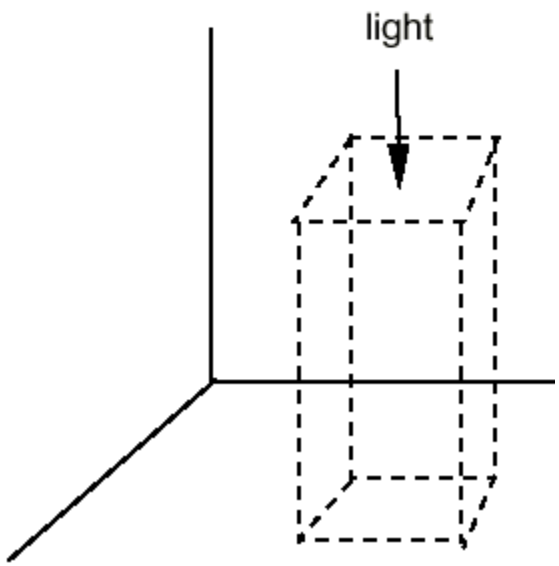
- For multiple light sources
  - Repeat the diffuse and specular calculations for each light source
  - Add the components from all light sources
  - The ambient term contributes only once
- The different reflectance coefficients can differ.
  - Simple "metal": K_s and K_d share material color,
  - Simple plastic: K_s is white
- Remember, when cosine is negative, lighting term is zero!

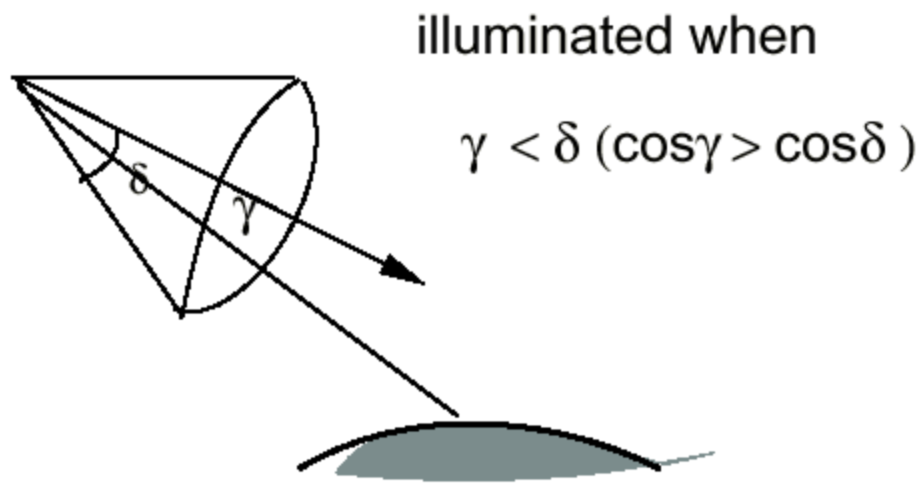
Light Sources

- Directional light source: e.g. sun light
- Positional (point) light source: e.g. lamp
- Spot light

Spot Light

- To restrict a light's effects to a limited area of the scene
- Flap: confine the effects of the light to a designed range in x, y, and z world coordinate
- Cone: restrict the effects of the light using a cone with a generating angle δ





Light Source Attenuation

- Takes into account the distance of the light from the surface

$$I'_L = I_L \; f_{att}(d)$$

I'_L : the received light after attenuation

I_L : the original light strength

f_{att} : the attenuation factor

d : the distance between the light source and the surface point

- $f_{att} = max\,(1 / (c1 + c2\, d + c3\, d^2) , 1)$
- $c1$, $c2$, and $c3$ are user defined constants associated with each light source

OpenGL – Material Properties

```
GLfloat white8[] = {.8, .8, .8, 1.};
GLfloat white2[] = {.2, .2, .2, 1.};
GLfloat mat_shininess[] = {50.}; // Phong exponent

glMaterialfv(GL_FRONT_AND_BACK,
   GL_AMBIENT_AND_DIFFUSE, white8);

glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, white2);

glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,
     mat_shininess);
```

OpenGL Lighting

```
GLfloat white[] = {1., 1., 1., 1.};
Glfloat amb[] = {.3, .3, .3, 1};
/* directional light (w=0) */
GLfloat light0_position[] = {1., 1., 5., 0.};

glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_DIFFUSE, white);
glLightfv(GL_LIGHT0, GL_SPECULAR, white);
glLightfv(GL_LIGHT0, GL_POSITION, light0_position);
glEnable(GL_LIGHT0);

/* normalize normal vectors */
glEnable(GL_NORMALIZE);

glEnable(GL_LIGHTING);
```

Shading Models for Polygons

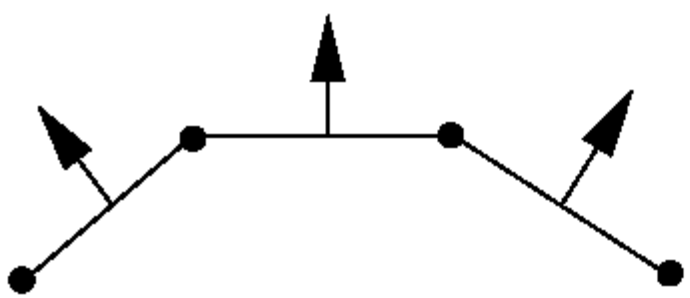
- Constant Shading (flat shading)
  - Compute illumination at any one point on the surface. Use face or one normal from a pair of edges. Good for far away light and viewer or if facets approximate surface well.
- Interpolated Shading
  - Compute illumination at vertices and interpolate color.
- Per-Pixel Shading
  - Compute illumination at *every* point on the surface.

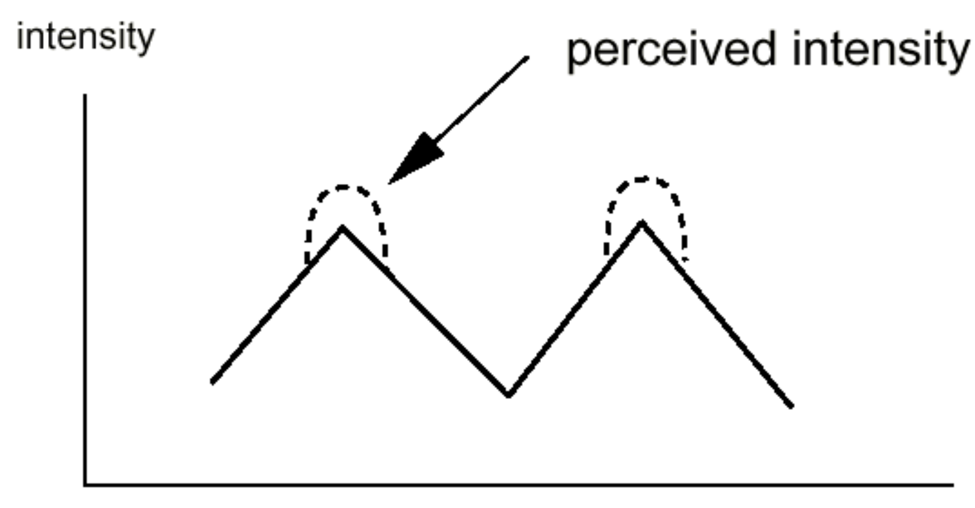
Constant Shading

- Compute illumination only at one point on the surface
- Okay to use if all of the following are true
  - The object is not a curved (smooth) surface (e.g. a polyhedron object)
  - The light source is very far away (so N•L does not change much across a polygon)
  - The eye is very far away (so V•R does not change much across a polygon)
  - The surface is quite small (close to pixel size)

Polygon Mesh Shading

- Shading each polygonal facet individually will not generate an illusion of smooth curved surface
- Reason: polygons will have different colors along the boundary, unfortunately, human perception helps to even accentuate the discontinuity: mach band effect





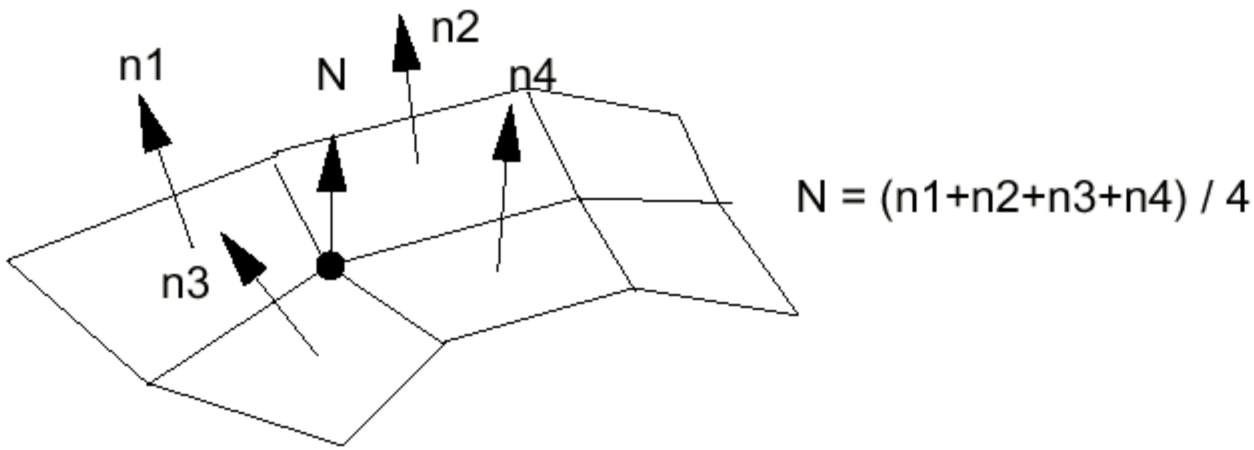
Mach Band Effect

Smooth Shading

- Need to have per-vertex normals
- Gouraud Shading
  - Interpolate color across triangles
  - Fast, supported by most of the graphics accelerator cards
- Phong Shading
  - Interpolate normals across triangles
  - More accurate, but slow. Not widely supported by hardware

Gouraud Shading

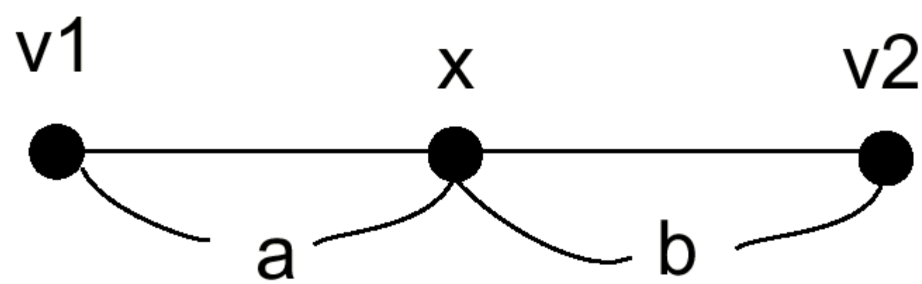
- Normals are computed at the polygon vertices
- If we only have per-face normals, the normal at each vertex is the average of the normals of its adjacent faces
- Intensity interpolation: linearly interpolate the pixel intensity (color) across a polygon surface



Linear Interpolation

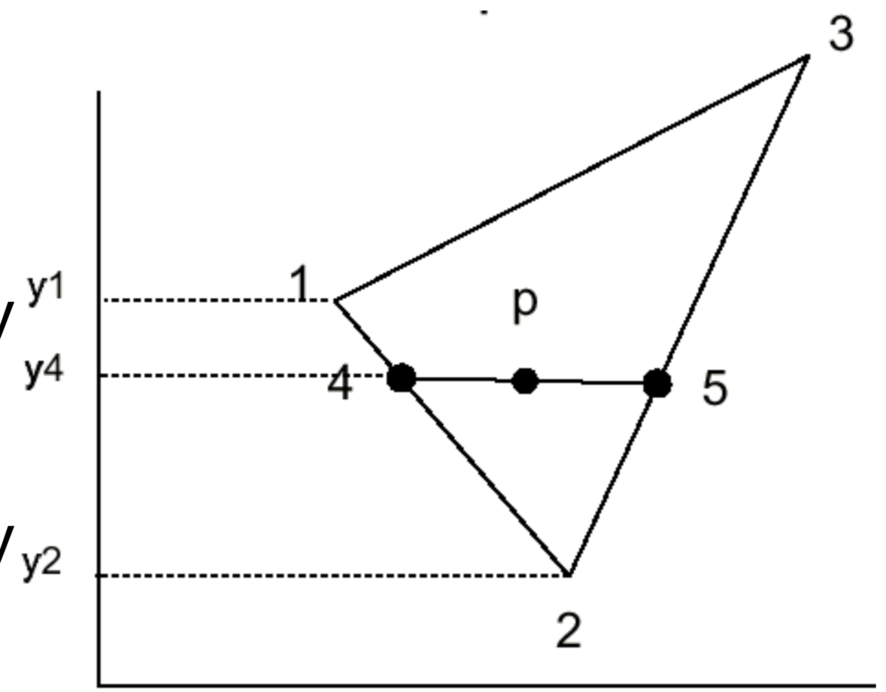
- Calculate the value of a point based on the distances to the point's two neighbor points
- If v1 and v2 are known, then

$$x = b/(a+b) * v1 + a/(a+b) * v2$$



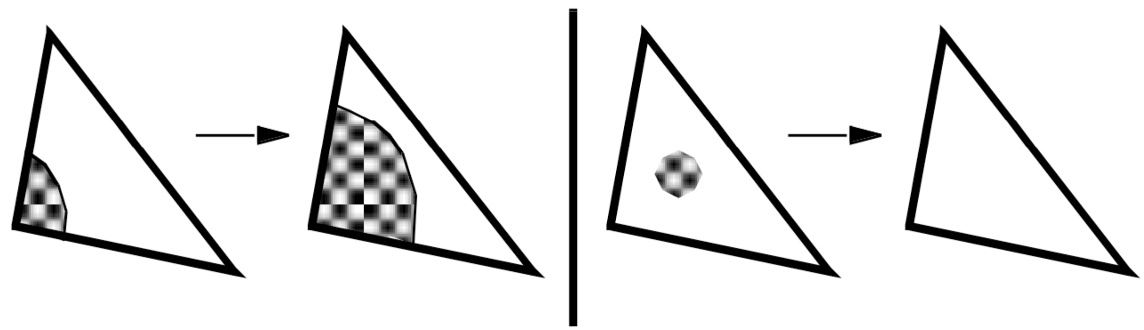
Linear Interpolation in a Triangle

- To determine the intensity (color) of point P in the triangle,
- we will do:
- determine the intensity of 4 by linearly interpolating between 1 and 2
- determine the intensity of 5 by linearly interpolating between 2 and 3
- determine the intensity of P by linear interpolating between 4 and 5

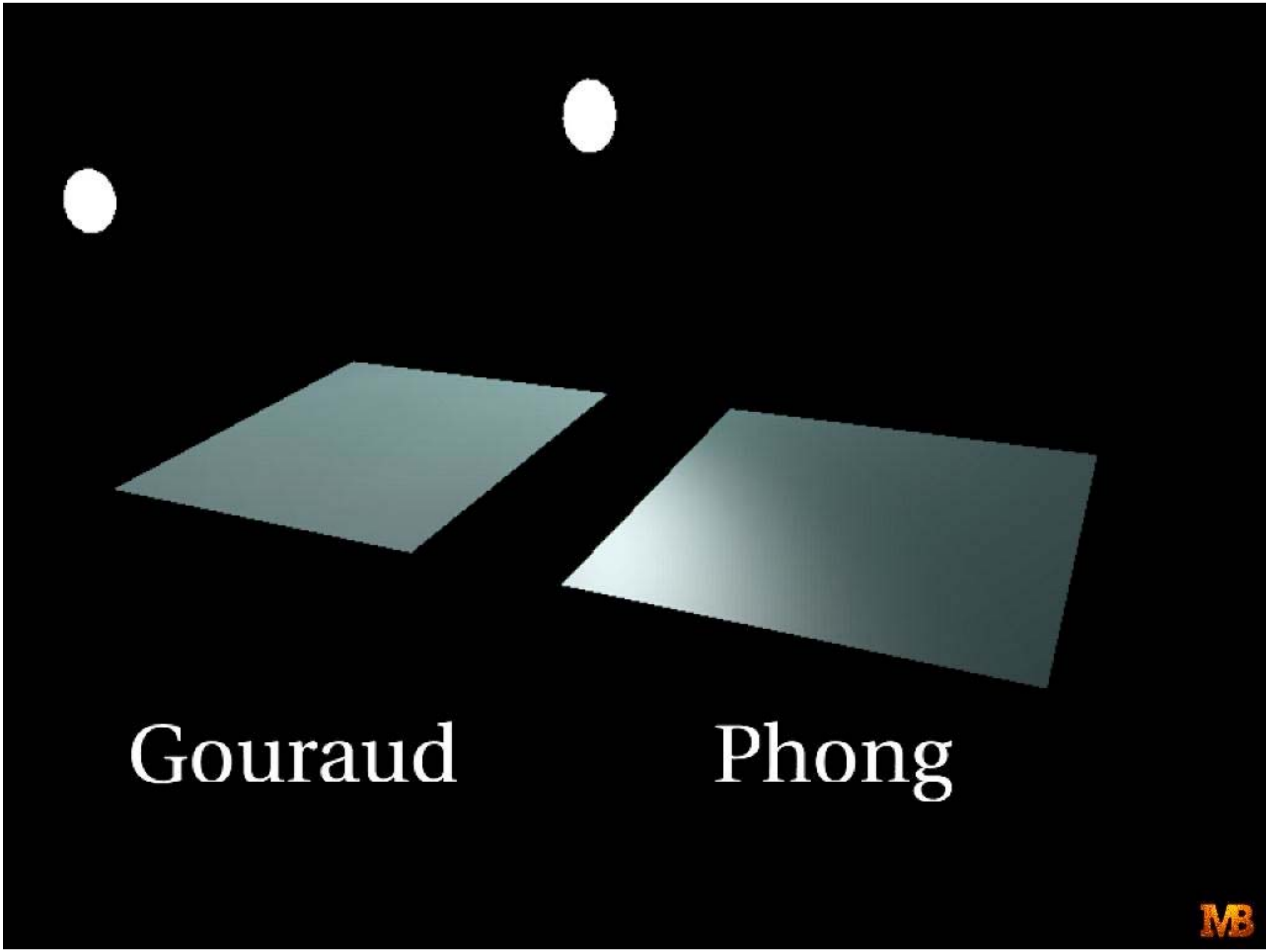


Phong Shading Model

- Gouraud shading does not properly handle specular highlights, especially when the *n* parameter is large (small highlight).



- **Reason:** colors are interpolated.
- **Solution:** (Phong Shading Model)
  - 1. Compute averaged normal at vertices.
  - 2. Interpolate *normals* along edges and scan-lines. (component by component)
  - 3. Compute *per-pixel* illumination.


Gouraud
Phong

Interpolated Shading - Problems

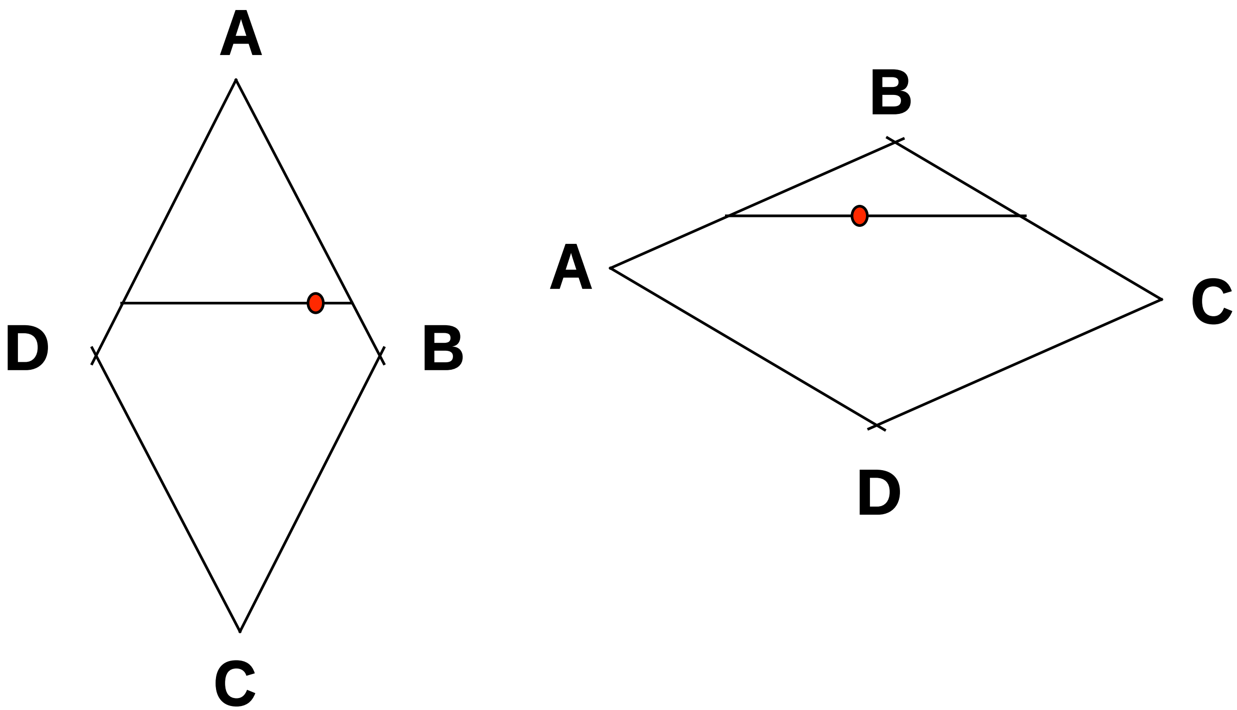
- Polygonal silhouette – edge is always polygonal. Solution ?

- Perspective distortion – interpolation is in screen space and hence for-shortening takes place. Solution ?

In both cases finer polygons can help !

Interpolated Shading - Problems

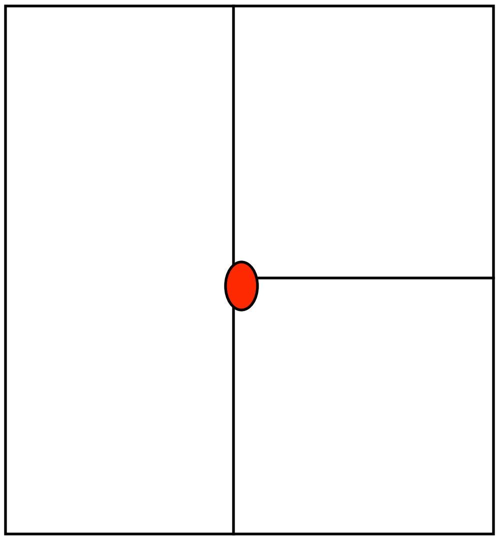
- Orientation dependence - small rotations cause problems

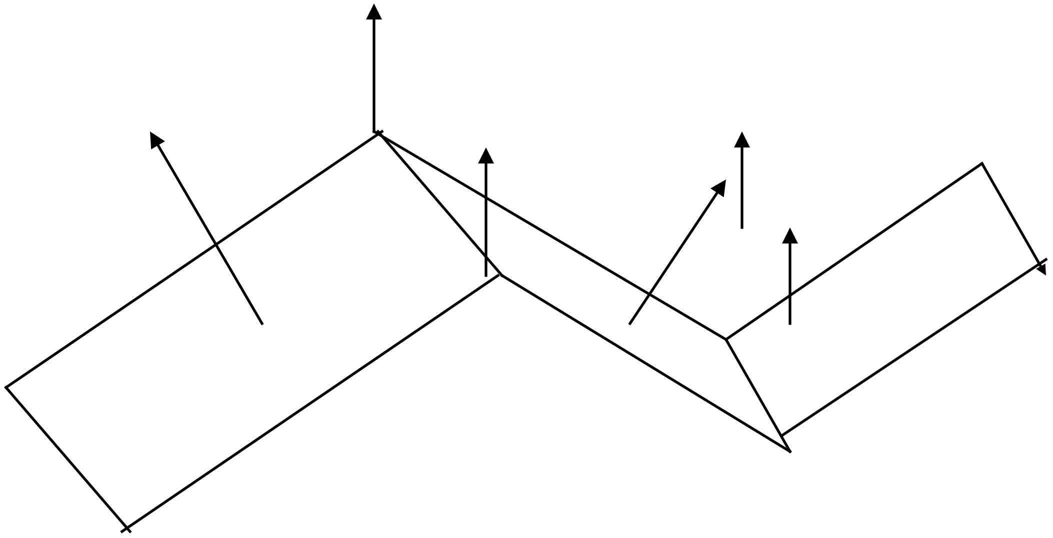


Interpolated Shading - Problems

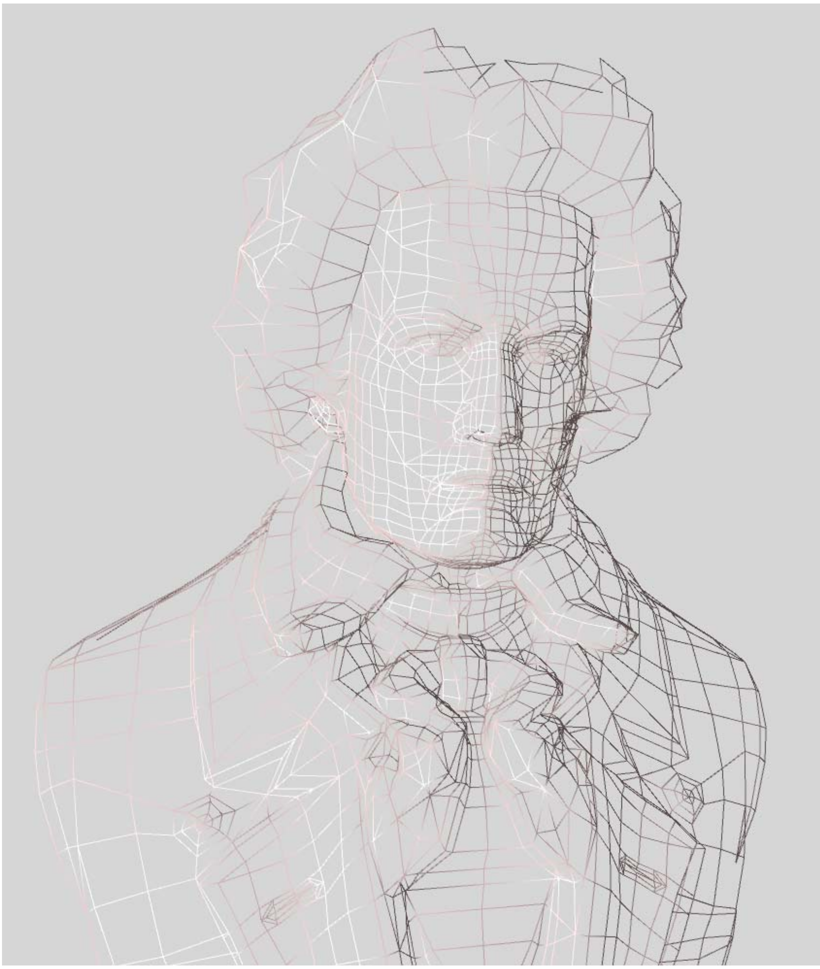
- Problems at shared vertices – shared by right polygons and not by one on left and hence discontinuity

- Incorrect Vertex normals – no variation in shade

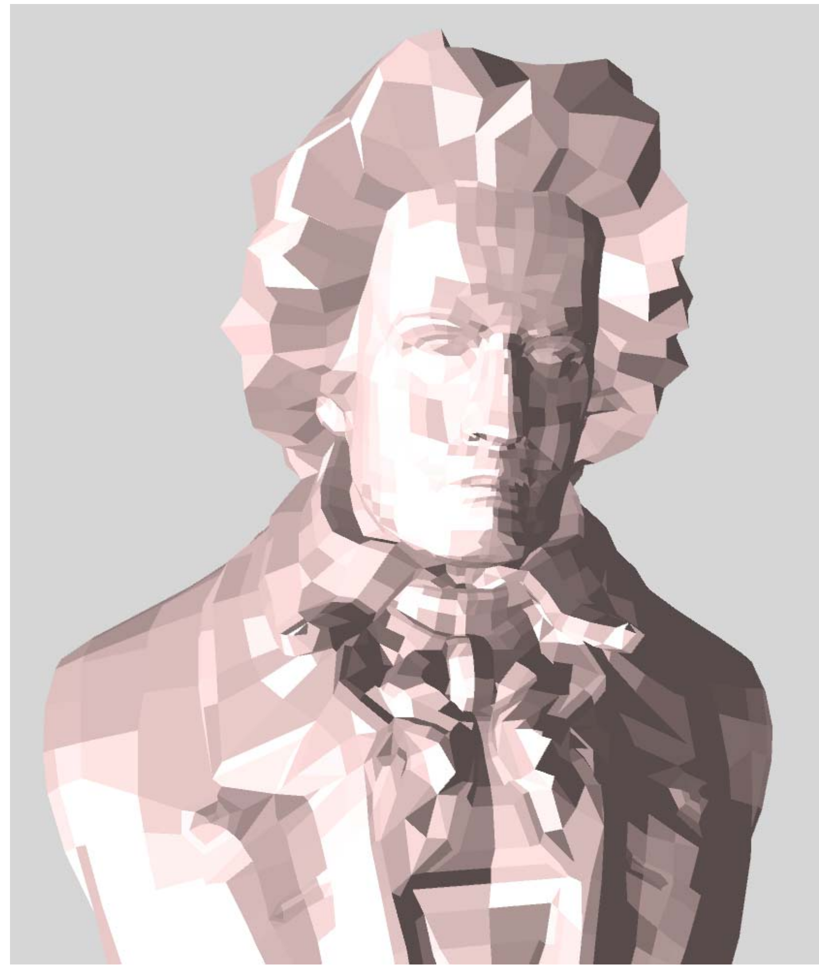




OpenGL Examples



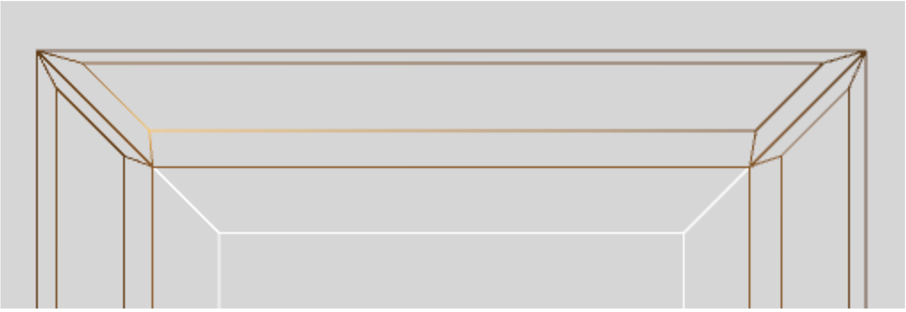
Mesh



Flat Shading

Gouraud Shading

Phong Shading


mesh


Flat Shading


Gouraud Shading


Phong Shading

Gouraud

Phong