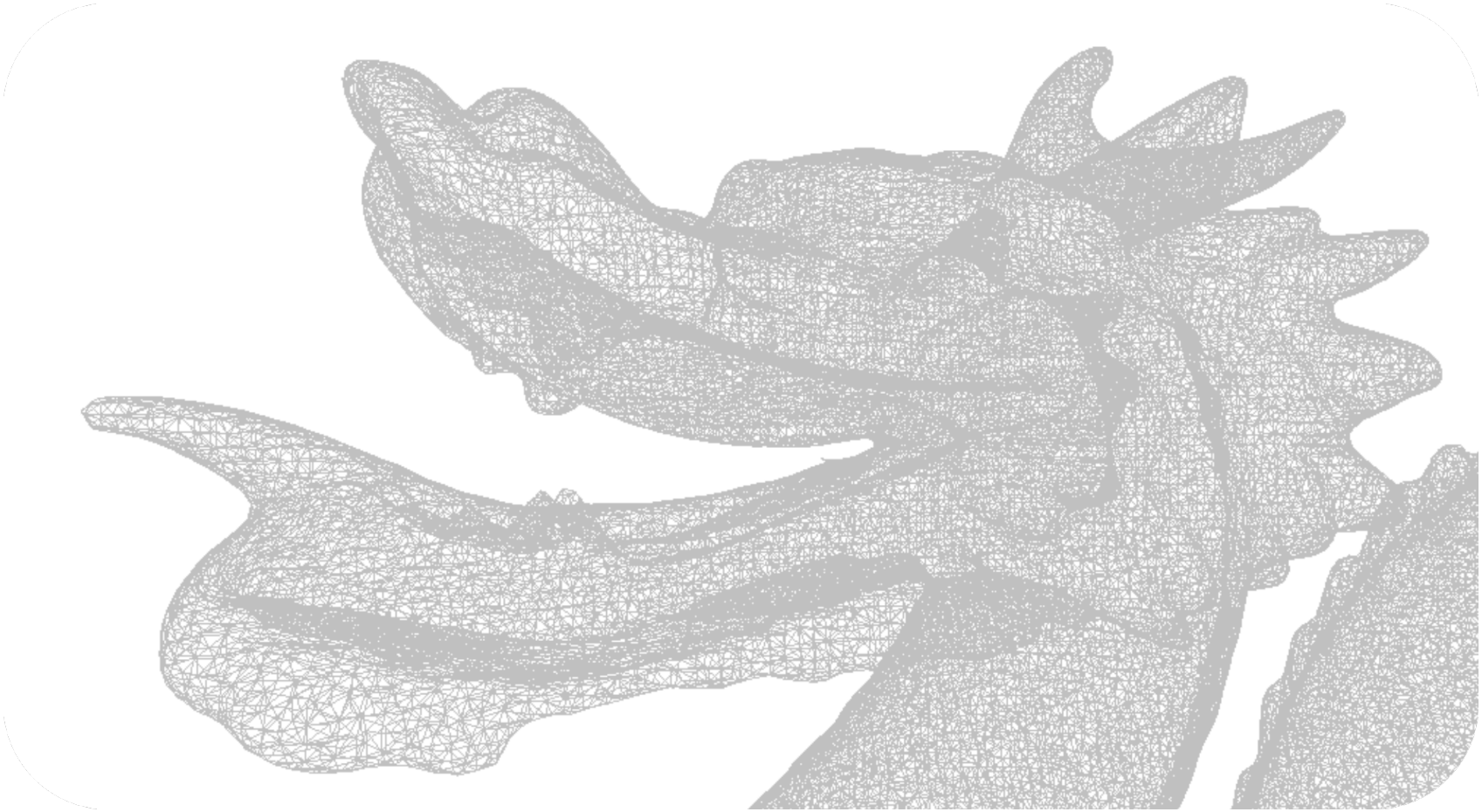
CS 461 - Computer Graphics

Shading

Recap (3D)

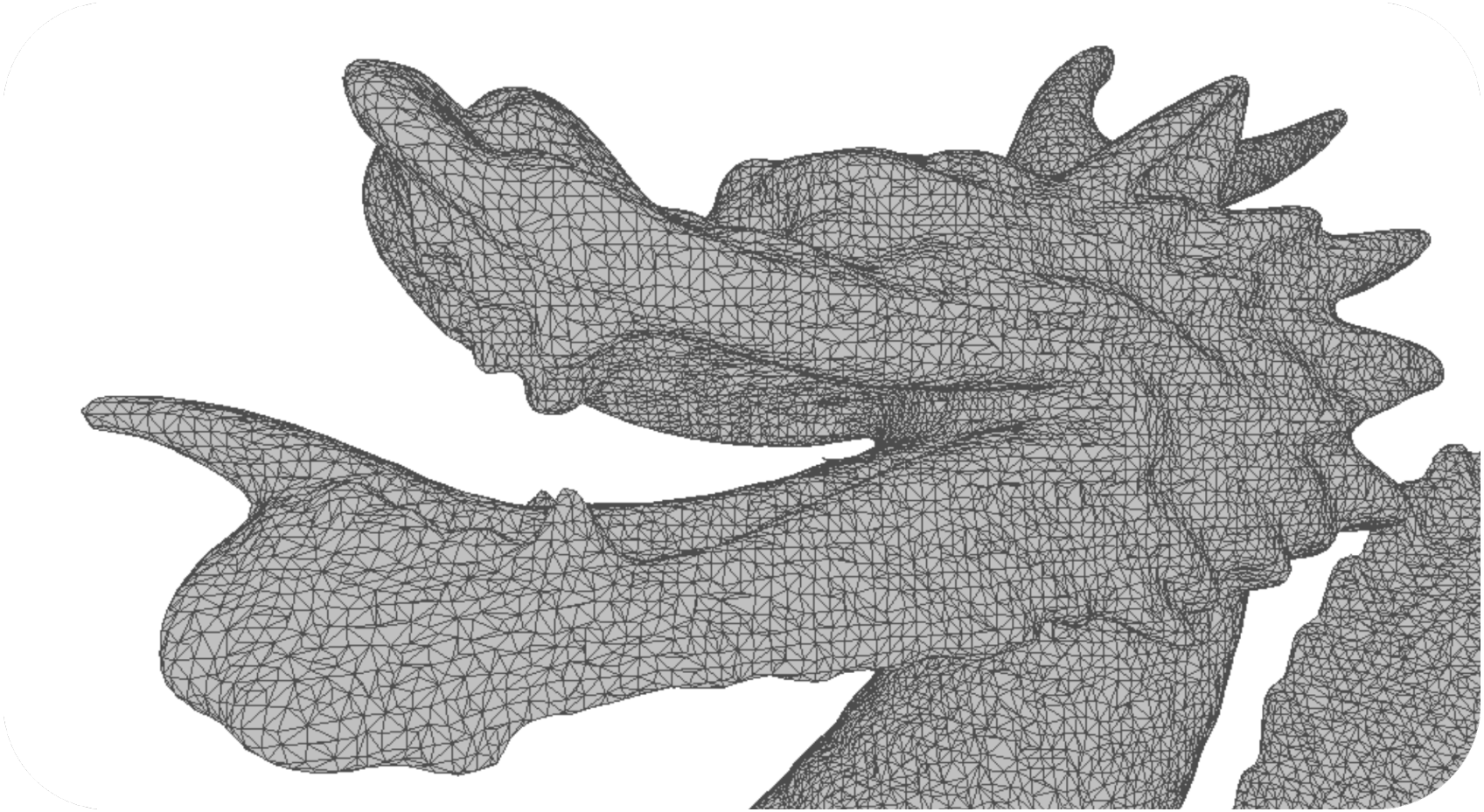
- Acquiring 3D models $\longrightarrow$ 3D scan+reconstruction, Photometric Reconstruction, 3D drawing (VR-based, sketch-based)
- 3D processing $\longrightarrow$ Simplification, noise removal, upsampling, remeshing
- Representing 3D models $\longrightarrow$ Polygon meshes, implicit representations, parametric representations, subdivision surfaces
- Removing unnecessary parts $\longrightarrow$ Culling, HSE algorithms
- Colors

Why Shading

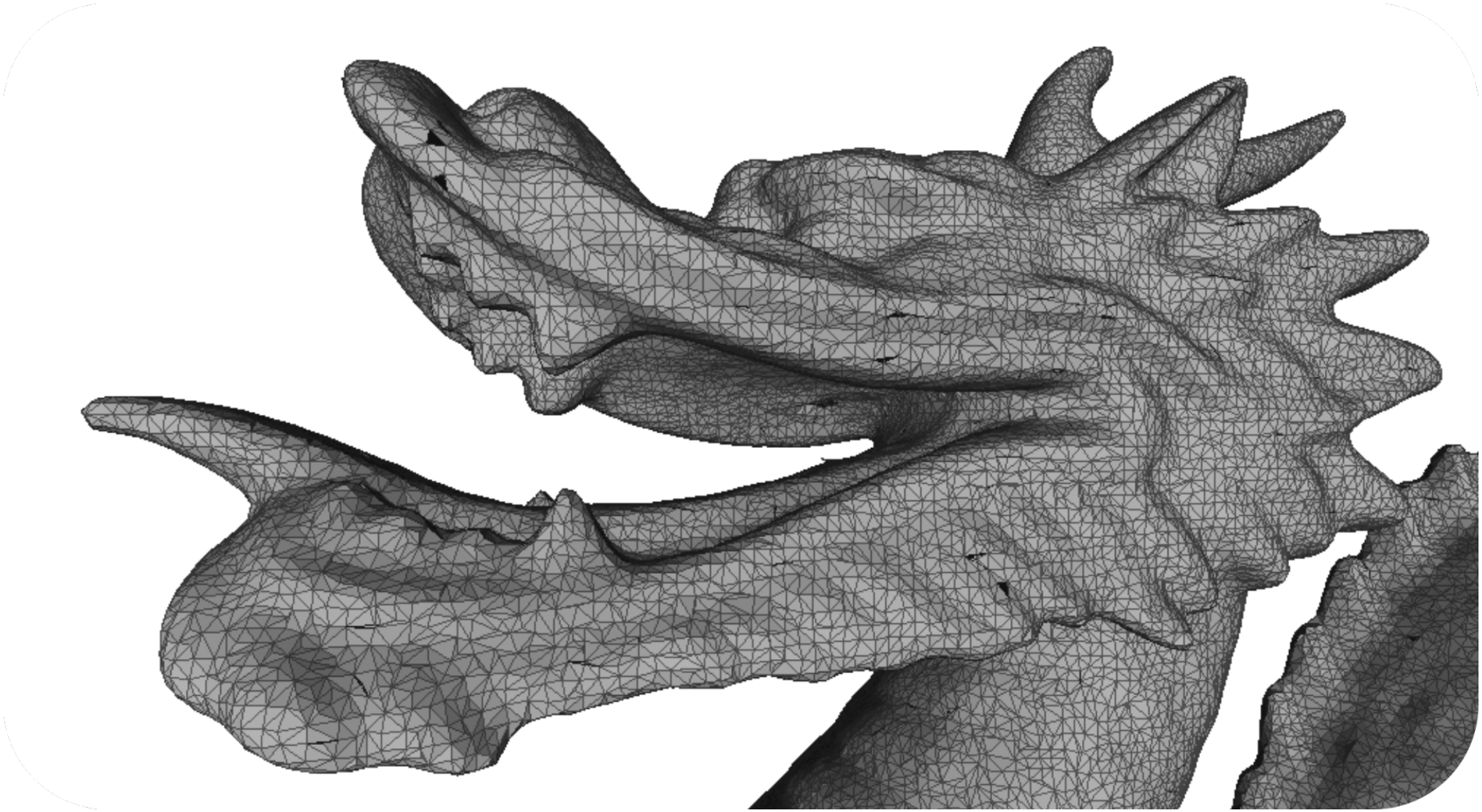


Why Shading

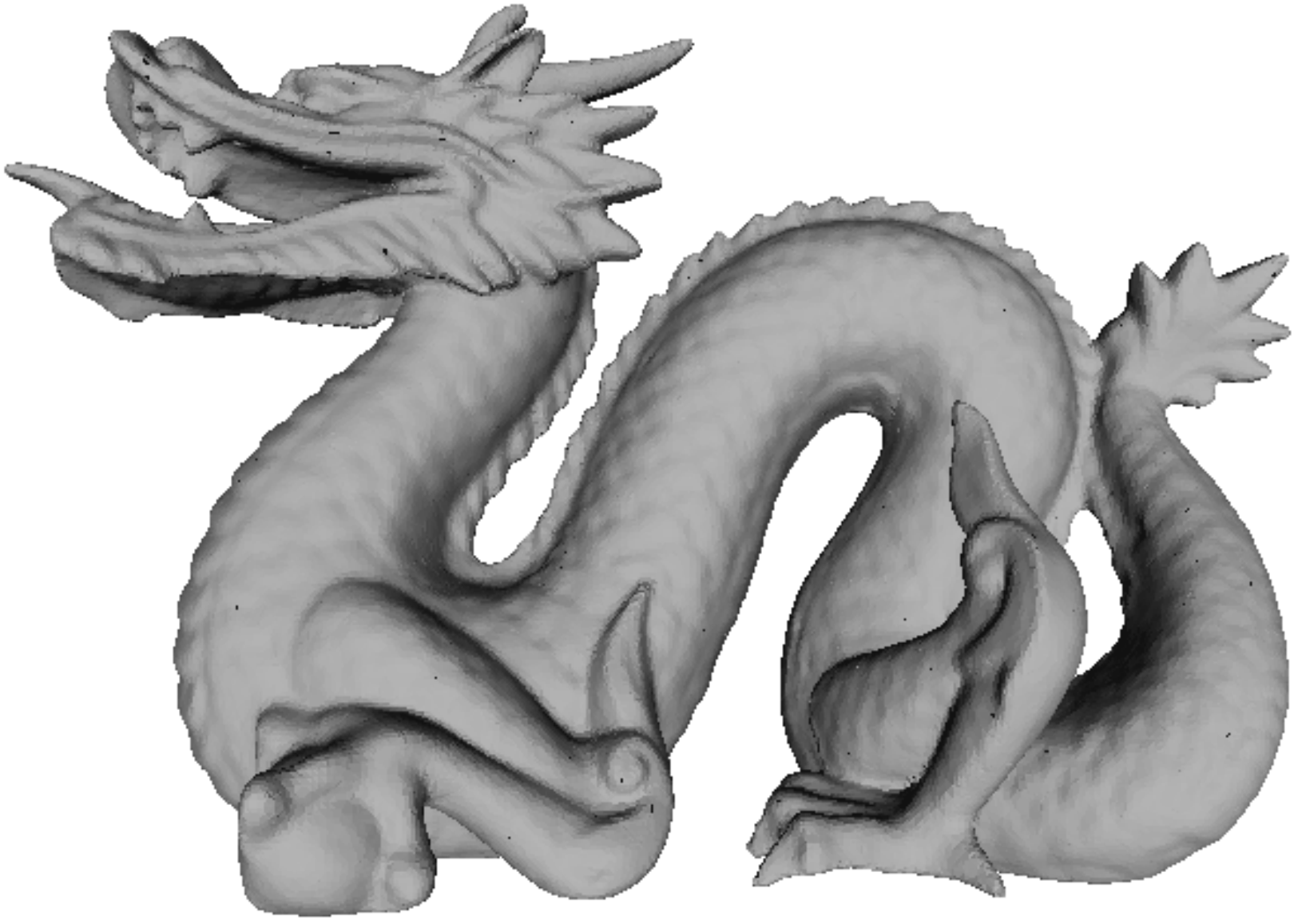
Why Shading



Why Shading



Why Shading

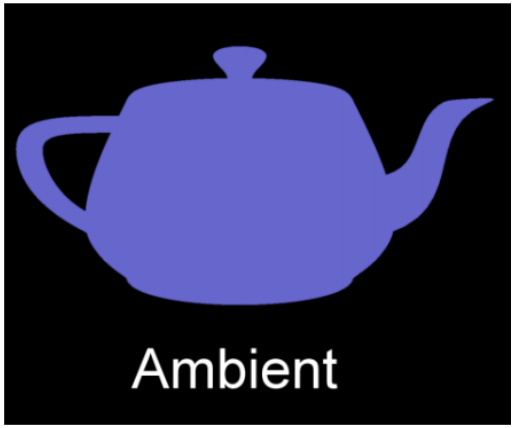


Light sources

- Sun, light bulbs, and any other light - emitting sources
- Point light sources - emits light equally in all directions
- Spotlights - a narrow range of angles through which light is emitted
- Distant light sources - parallel light sun

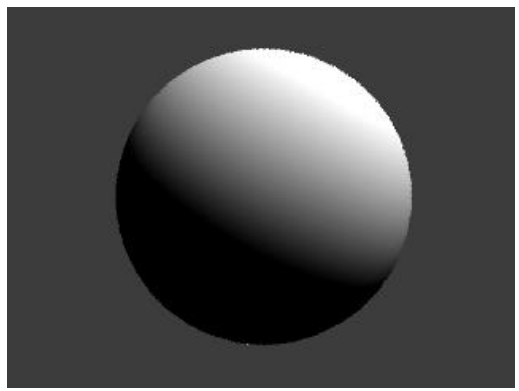
Ambient light

- Ambient light means the light that is already present in a scene, before any additional lighting is added
- It usually refers to natural light, either outdoors or coming through windows etc

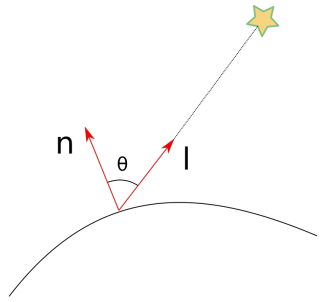


Diffusion light

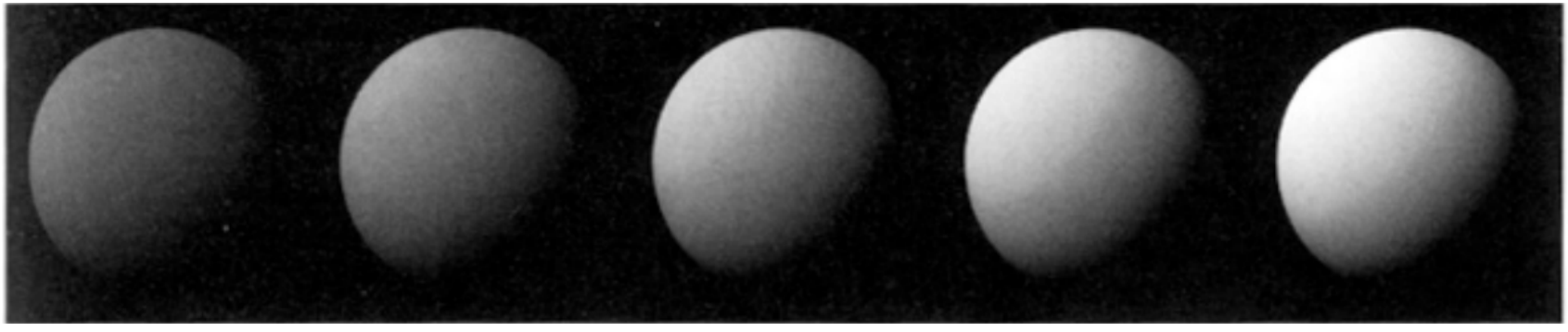
- Diffused light is a soft light with neither the intensity nor the glare of direct light
- It is scattered and comes from all directions
- It seems to wrap around objects



Lambertian shading model

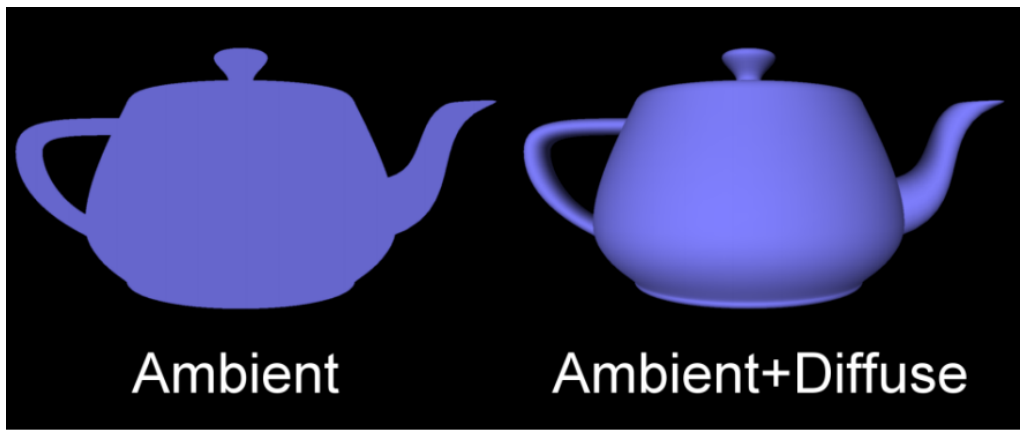


- New pixel color = Surface color * Diffusion intensity * max(0, n.l)
- Viewing direction does not matter
- Diffuse case



Ambient+Diffusion lighting

- $I_R = C_R(amb_R + diff_R * \max(0, n.l))$
- $I_G = C_G(amb_G + diff_G * \max(0, n.l))$
- $I_B = C_B(amb_B + diff_B * \max(0, n.l))$

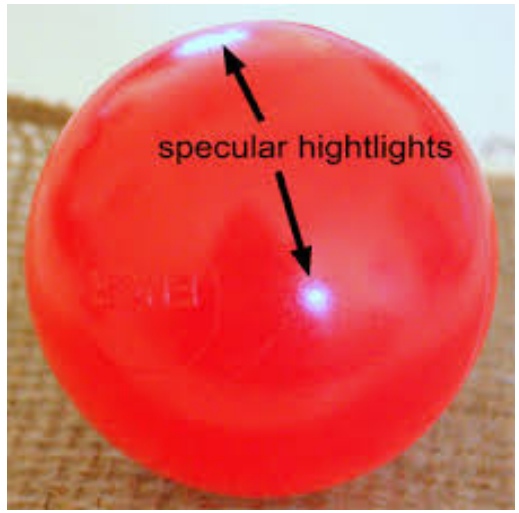


Eye position

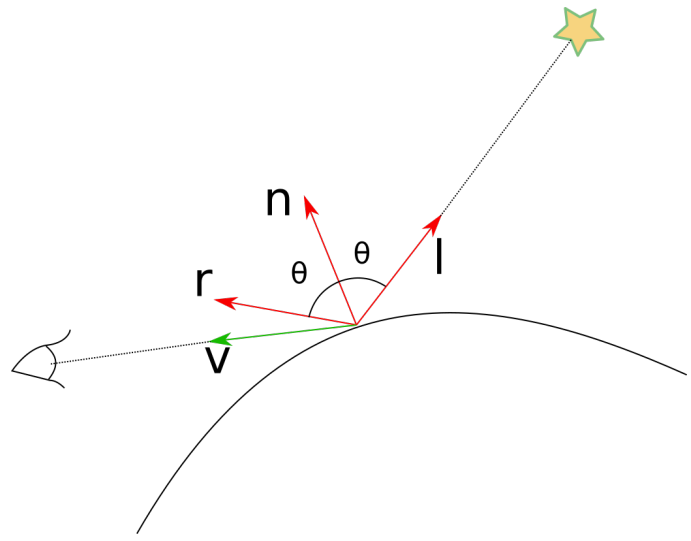
- Eye position and shading effect???
- Lambertian shading model and eye positon???

Specular light

- Specular lighting identifies the bright specular highlights that occur when light hits an object surface and reflects back toward the camera
- Specular lighting is more intense than diffuse light and falls off more rapidly across the object surface



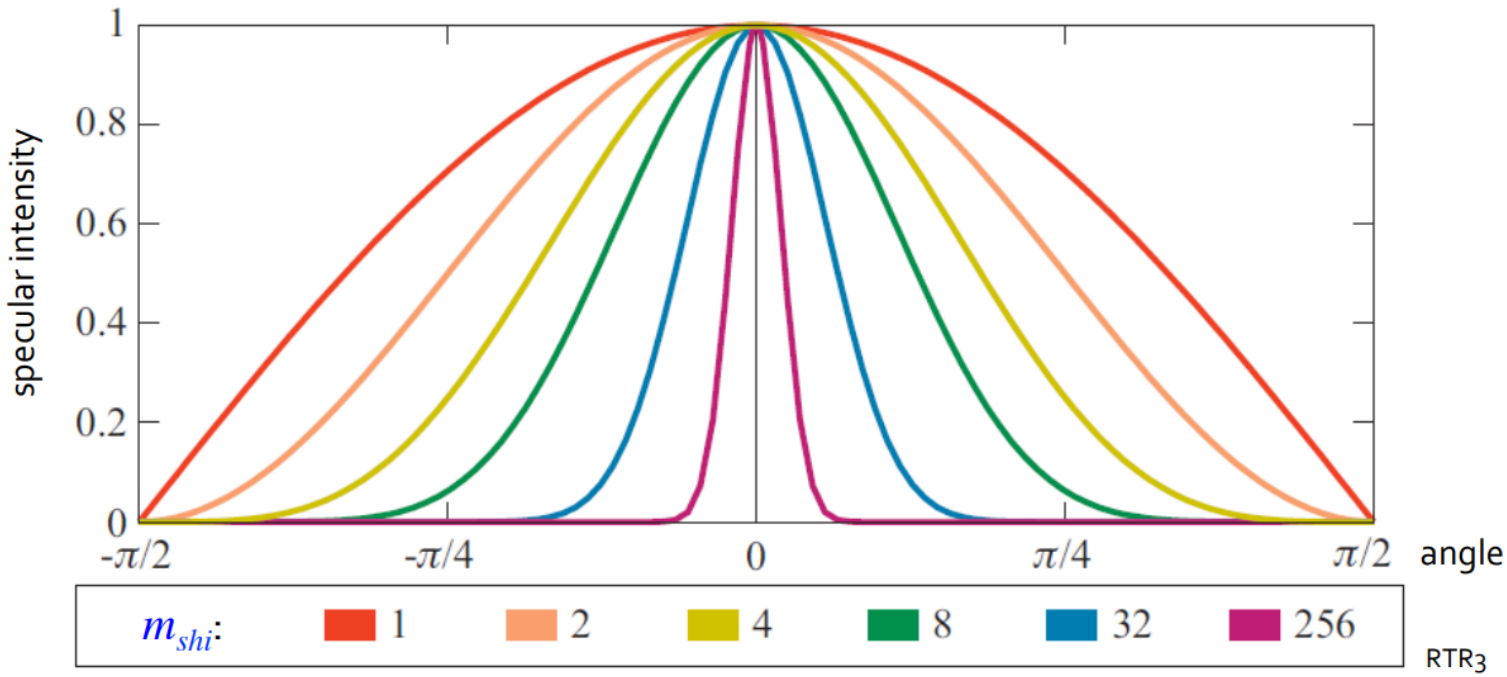
Phong reflection model



- Simple idea: v.r
- But we won't use $\longrightarrow$ Slow transition
- What can we do?

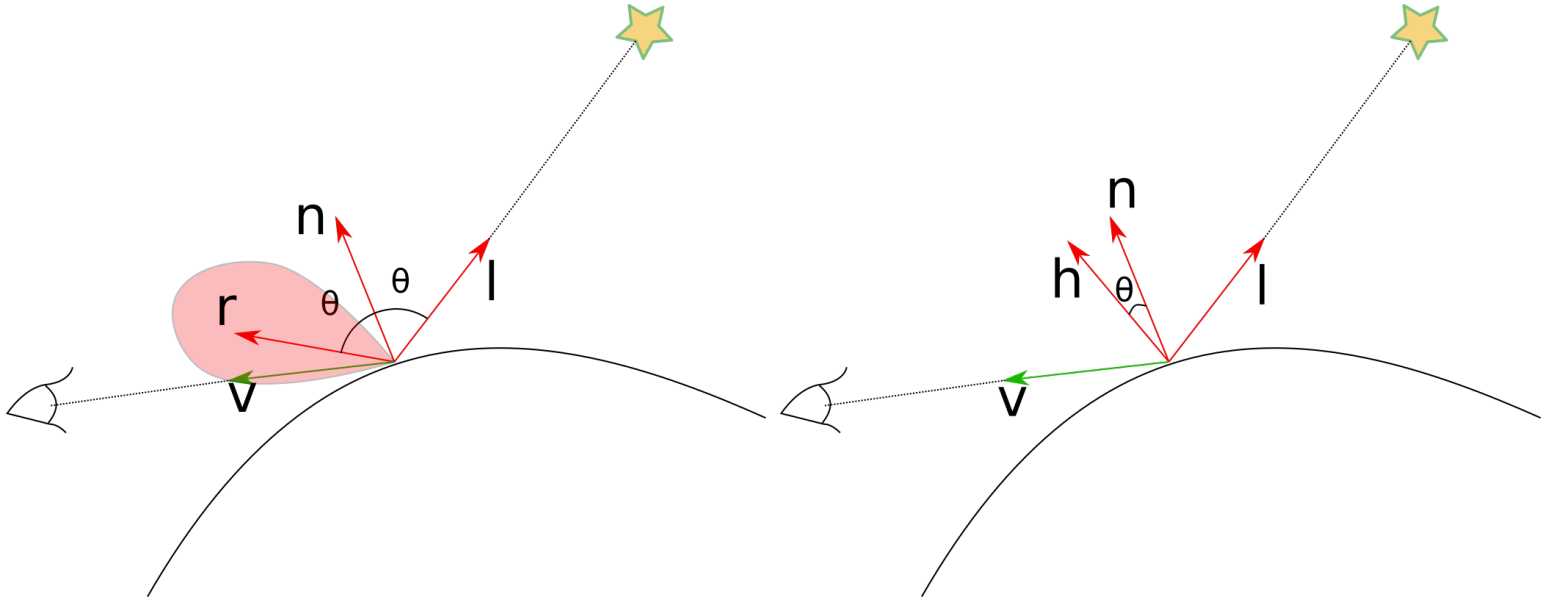
Phong reflection model

- Play with ***cos*** function



- $(v.r)^{glossiness}$
- Glossiness determines how quickly the light disappears

Phong reflection model

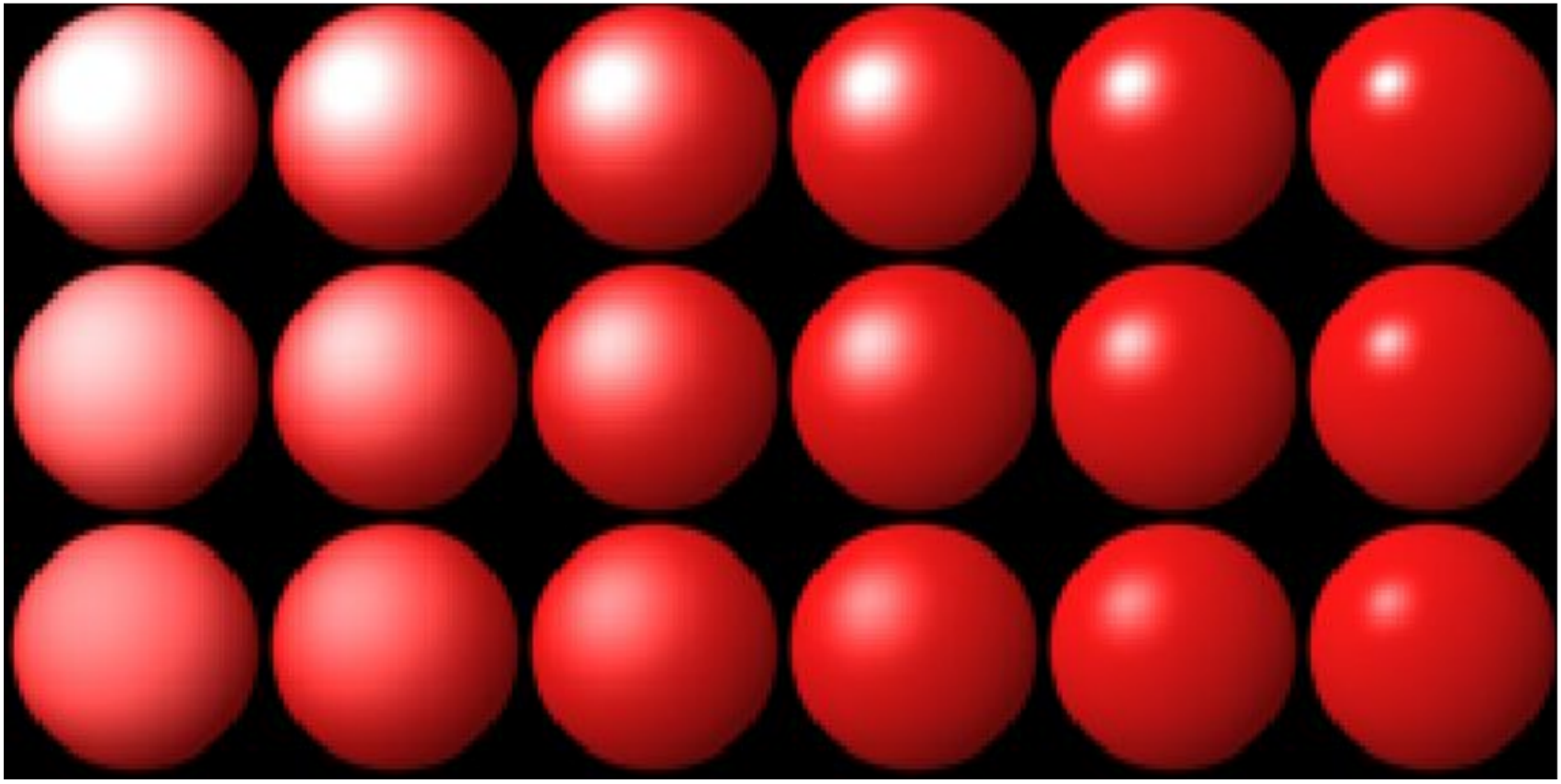


- Computing for large glossiness is difficult
- Need some hack
- Define a new vector in the middle of L and R
- What is the meaning?

Phong reflection model

- Can be written as $(n.h)^m$
- Not all incident light get reflected
- So, $specular\_constant * (n.h)^m$
- Depends on light color: $LC * spec * (n.h)^m$

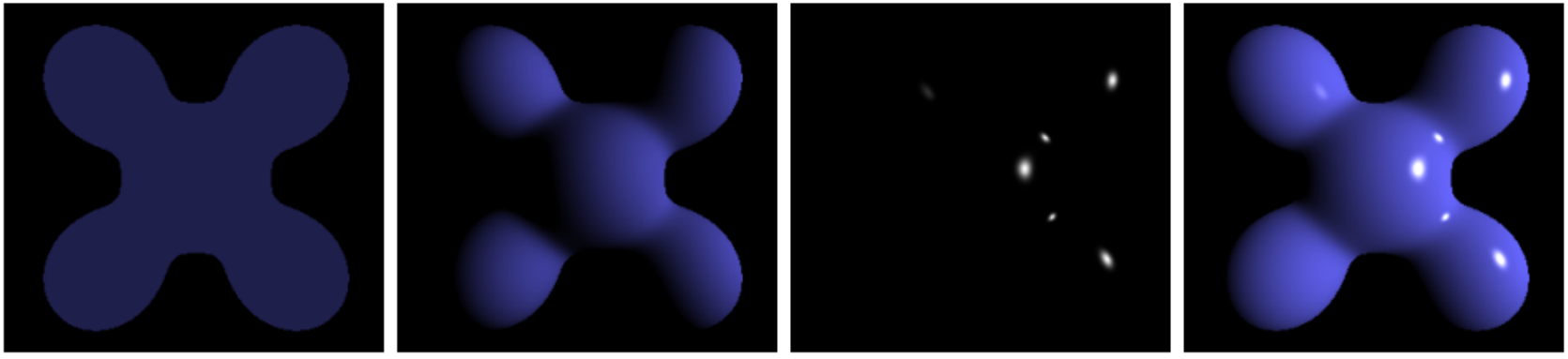
Effect of *spec* and *m* parameters



Putting it all together

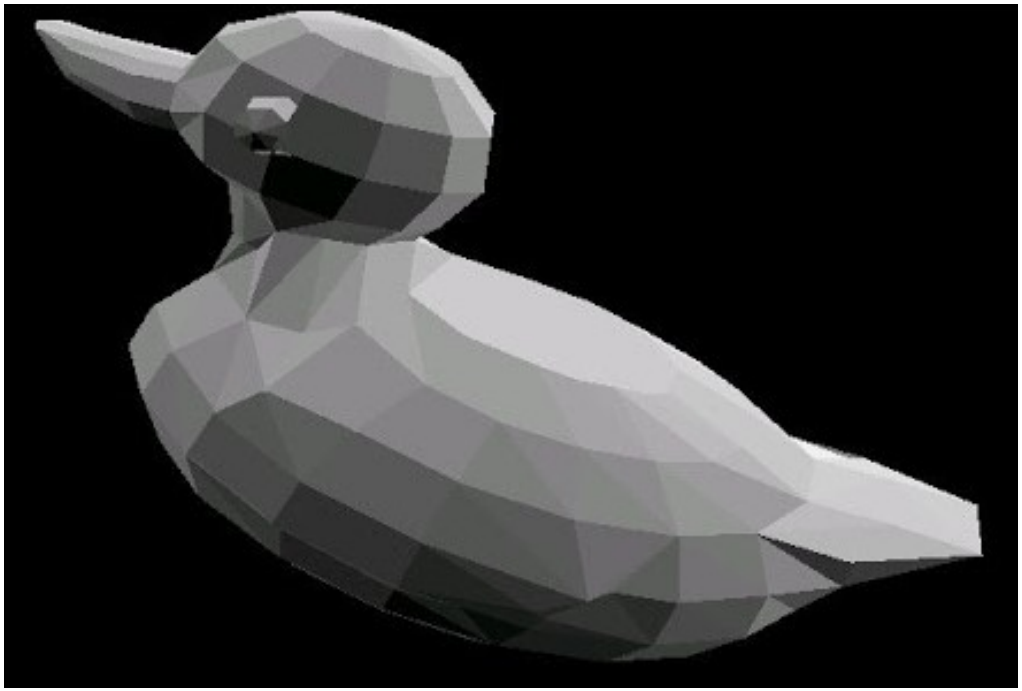
- Adding specular component to our initial equations:
  - $I_R = C_R*(amb_R + diff_R*\max(0,n.l))+LC_R * spec_R * (n.h)^m$
  - $I_G = C_G*(amb_G + diff_G*\max(0,n.l))+LC_G * spec_G * (n.h)^m$
  - $I_B = C_B*(amb_B + diff_B*\max(0,n.l))+LC_B * spec_B * (n.h)^m$

Example



Flat shading

- Comes back to triangles!!!
- Use the same normal
- Single color



Gouraud shading

- Compute vertex colors
- Linear interpolation of color across polygon surface

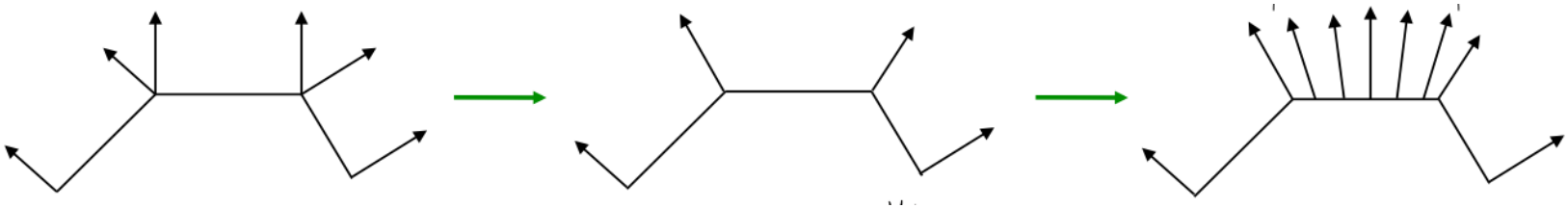
Gouraud shading algorithm:

- determine average normal at each vertex (averaged over the normals of all polygons that share the vertex)
- compute color at each vertex using the average normal
- linearly interpolate color across a single polygon surface

Phong shading

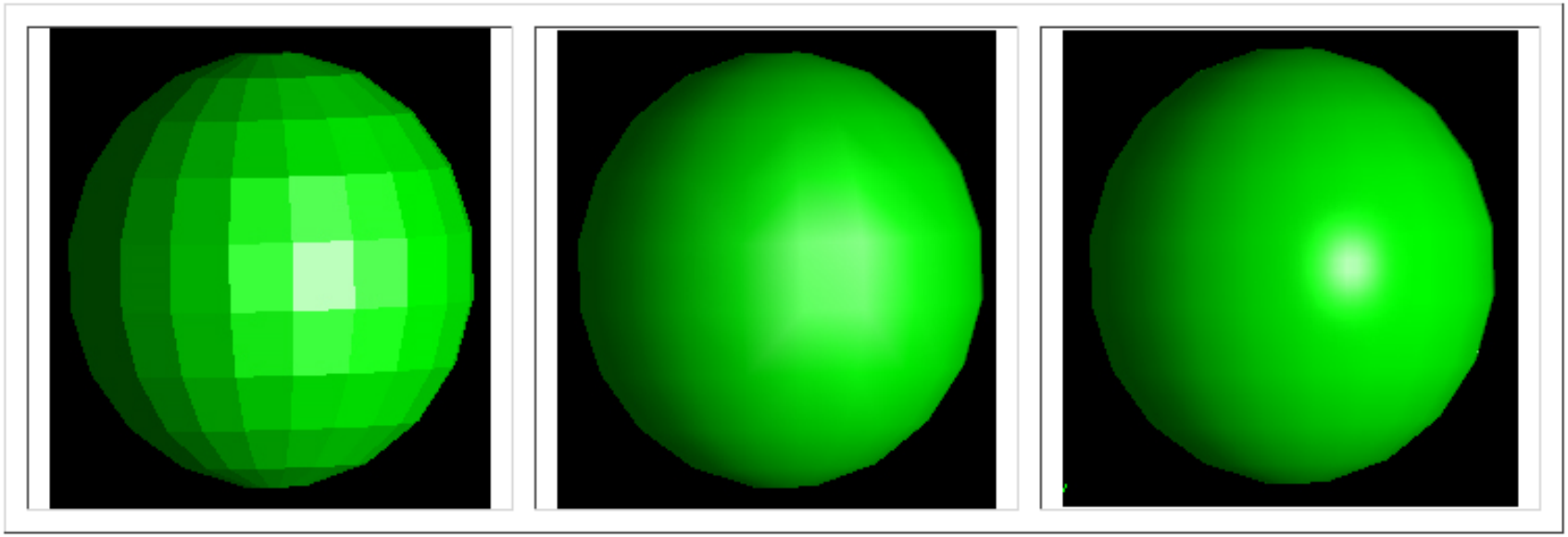
- Normals are computed for vertices
- Interpolated for each interior points

Algorithm:
- determine average normal at each vertex
- linearly interpolate per-pixel normal across surface
- compute color for each pixel using the value of the approximated per-pixel normal

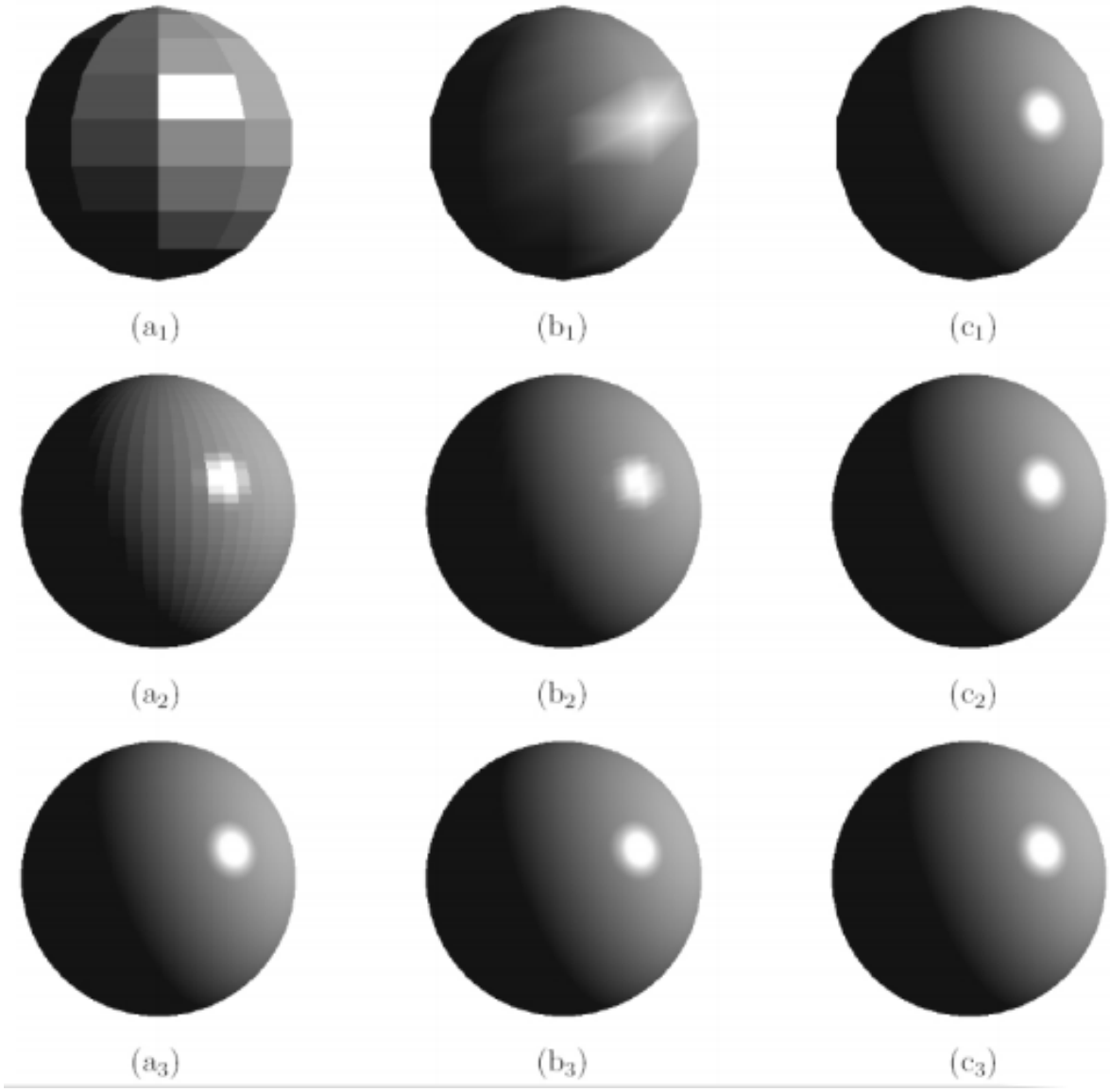


- Micro-polygon generator

Shading



Shading



Phong reflection model

- Equation is incorrect (not sufficient)
- Plastic shading
- Concept of global illumination
- Old animation movies

Next class

- 15^{th} October - 9 to 10
- Preponing seminars from 12^{th} and $16^{th} \longrightarrow 9^{th}$
- 31 projects - 64 members
- Find some slots between 2^{nd} to 8^{th} October - 11 am to 2 am + one report by 9^{th} morning
- Upcoming - Two more simple coding assignments (Rendering and animation) + one more written assignment ([1] Barycentric coordinates and [2] Quaternions)
- Theory part - 30 (seminar) + 30 (assignments - 3*10) + 40 (project)
- Lab component - 25*4 coding assignments