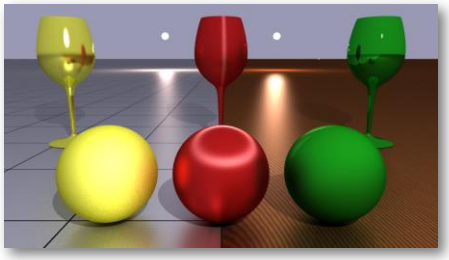
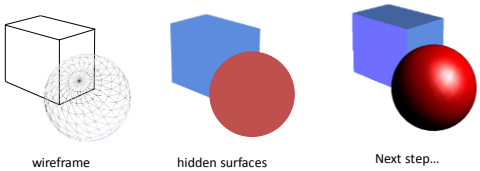
Illumination Models and Shading



1

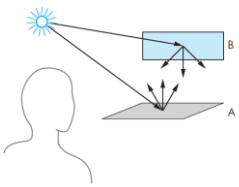
So Far …



2

What's Missing ?

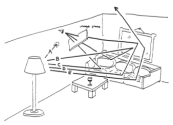
- Light !
- Need to understand:
  - How lighting works
    - Types of lights
    - Types of surfaces
  - How shading works
    - Shading algorithms



3

Lighting vs. Shading

- Lighting
  - Interaction between materials and light sources
  - Physics
- Shading
  - Determining the color of a pixel
  - Computer graphics
- Shading usually determined by lighting



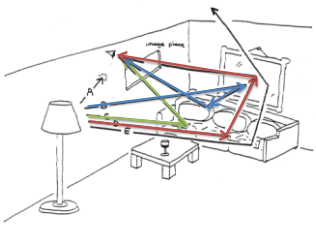
```
Zbuffer(Scene)
…
  putColor(x,y,col(P))
…
end
```

4

The Physics

Basic Illumination Model

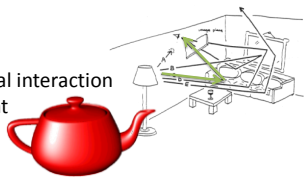
Light rays are emitted from light sources and bounce (reflect) in the scene until they reach the eye



5

Local vs. Global Illumination Model

- Local
  - only direct and local interaction of objects with light
- Global
  - ***all*** interactions and exchange of light
  - Can model more effects
    - Reflection, refraction, soft shadows



6

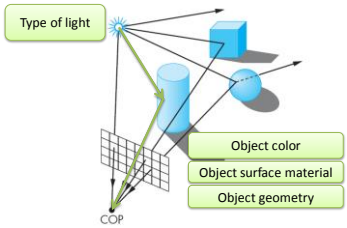
JACOBS TECHNION-CORNELL INSTITUTE AT CORNELL TECH

Local vs. Global

7

A Single Interaction

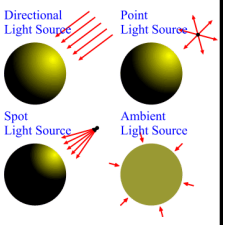
What determines pixel color?



8

Light Sources

- **Point**: All light originates at a point
  - Rays hit planar surface at different incidence angles
- **Directional**: All light rays are parallel
  - Rays hit a planar surface at identical incidence angles
  - May be modeled as point source at infinity
  - Also called *parallel source*
- **Area**: Light originates at finite area in space
  - In between the point and parallel sources
  - Also called *distributed source*
- **Ambient**: Background light
  - Comes equally from all directions

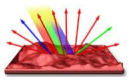


9

Material Properties

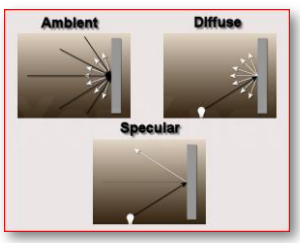
- Specular
  - Smooth surface
  - Reflects light at well-defined angle

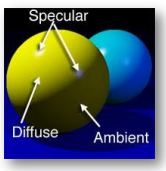
- Diffuse
  - Rough surface
  - Reflects light equally in all directions



10

Light/Reflection Types





11

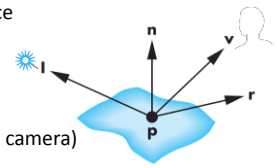
Phong Reflection Model

- Local illumination model
- Looking for a value $I(p)$ per point on the surface
- Represents the total contribution of all lights in the scene
- Takes into consideration
  - Position of the viewer
  - Position of the lights
  - Color of the lights
  - …

12

Some Notation

- l – direction to light source
- n – normal direction
- v – direction to COP (eye, camera)
- r – direction of reflected ray



13

Ambient Reflection

- Assume non-directional light in the environment
- Object illuminated with same light everywhere
  - Looks like a silhouette

k_a : fraction of ambient light reflected from surface
L_a: ambient light intensity

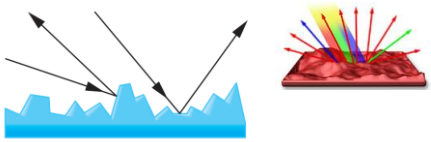
$$I_a = k_a L_a$$

- Defines object's color

14

Diffuse Reflection

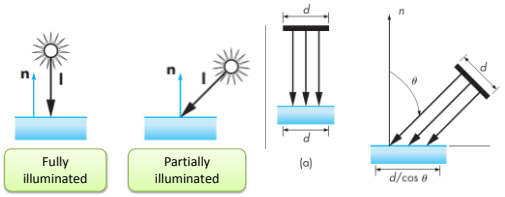
- Dull surfaces such as solid matte plastic reflects incoming light **uniformly in all directions**
- Called *diffuse* or *Lambertian* reflection



15

Diffuse Reflection

- How does the light direction affect the illumination?
- Larger angle θ with normal $\rightarrow$ less illumination density



16

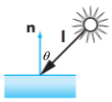
Diffuse Reflection

k_d : fraction of diffuse light reflected from surface

$$I_d = k_d(l \cdot n)L_d$$

L_d: diffuse light intensity

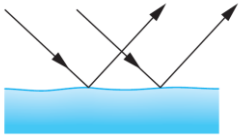
$l \cdot n = \cos\theta$



17

Specular Reflection

- Shiny objects (e.g. metallic) reflect light in preferred direction r determined by surface normal n.

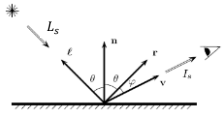


18

JACOBS TECHNION-CORNELL INSTITUTE AT CORNELL TECH

Specular Reflection

- Most objects are not ideal mirrors – also reflect in the immediate vicinity of r
- Approximate attenuation by $\cos^{\alpha} \varphi$ (no real physical basis)



19

Specular Reflection

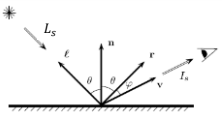
k_s : fraction of specular light reflected from surface

L_s: diffuse light intensity

$r \cdot v = \cos \varphi$

α : shininess coefficient

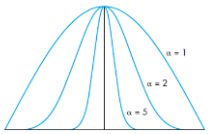
$$I_s = k_s(r \cdot v)^{\alpha} L_s$$



20

Specular Reflection

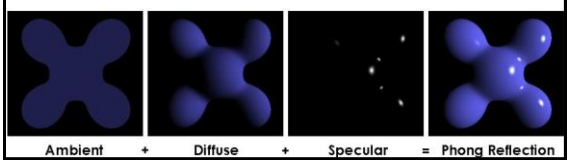
- Exponent α of cosine controls decay factor
- No physical basis, but looks good



21

Total Illumination

- $I = I_a + I_d + I_s$
- Sum over all light sources
- May use different coefficients for RGB components
- Beware of overflows



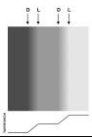
Triangle Shading Algorithms

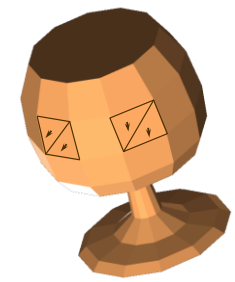
- Given the lights and materials in the scene, how do we compute the color at a given point on a triangle ?
- Three main types
  - Flat shading (per polygon)
  - Gouraud shading (per vertex)
  - Phong shading (per pixel)

23

Flat Shading

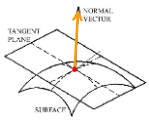
- Applied to piecewise linear polygonal models
- Simple surface lighting approximated over polygons
- Illumination value depends only on polygon normal ⇒ each polygon is colored with a uniform intensity
- Looks non-smooth (worsened by Mach band effect)

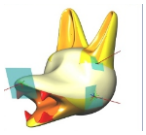




24

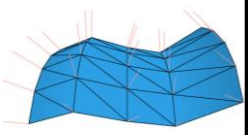
Normal per Vertex





If a polyhedron is an approximation of smooth surface:

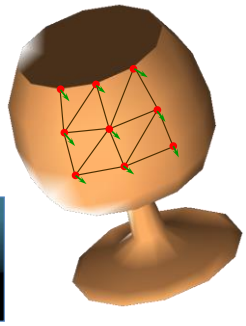
- Assign to each *vertex* the normal of original surface at that point
- If surface is not available use estimated normal (e.g. average of neighboring faces).

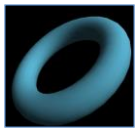


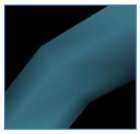
25

Gouraud Shading

- Compute illumination intensity at *vertices* using normals
- Linearly interpolate intensity over polygon interior



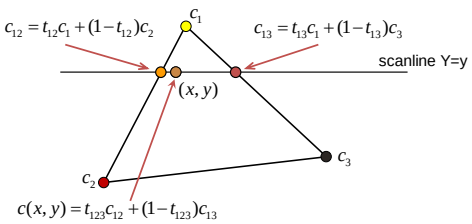




26

Gouraud Shading

Linearly interpolate lighting intensities at the vertices over interior pixels of the polygon, in the image plane

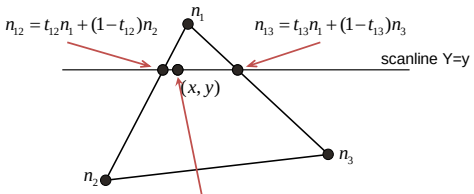


Question: Can Gouraud shading support specular lighting?

27

Phong Shading

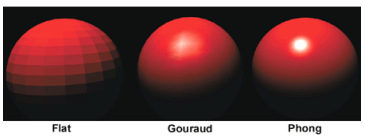
- Interpolate (at the vertices in image space) normal vectors instead of illumination intensities
- Apply the illumination equation for each interior pixel with its own (interpolated) normal

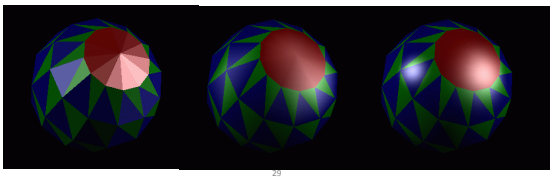


$n(x, y) = t_{123}n_{12} + (1-t_{123})n_{13}$; $c(x, y) = Illum(n(x, y))$

28

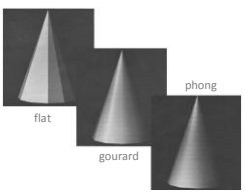
Comparison





29

Comparison



30