Lighting and Shading

Hendrik Speleers

Lighting and Shading

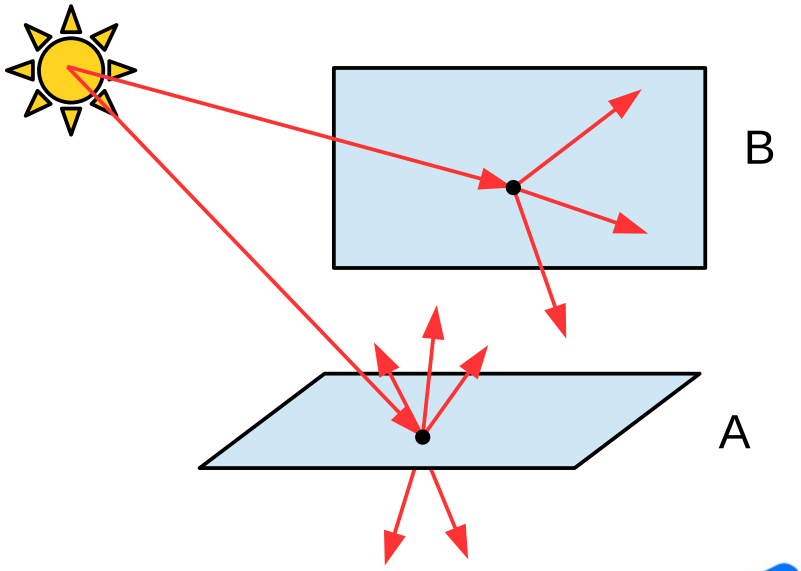
- Overview
  - Illumination: direct and indirect
  - Light sources in CG
  - Materials in CG
    - Diffuse reflection: Lambertian model
    - Specular reflection: Phong model
  - Shading models
    - Flat, Gouraud, Phong shading
    - Shadow buffering

Lighting and Shading

- Illumination
  - Realistic surface rendering: geometry + light sources
  - Lighting
    - The interaction between materials and light sources
    - Surface interaction is very complex
      - Microstructure of material
  - Shading
    - The process of determining the color of a pixel
    - How to simulate or model lighting interactions at CG level?
    - Could also use other methods: texture mapping, etc.

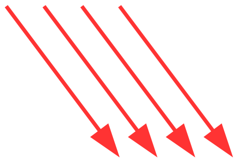
Lighting and Shading

- Illumination
  - Direct: light sources emit light
    - Position?
    - Direction?
  - Indirect: surfaces reflect light
    - Direction?
    - Absorption?
    - Reflection?
    - Transmittance?

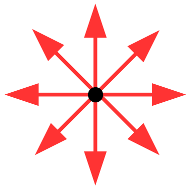


Lighting and Shading

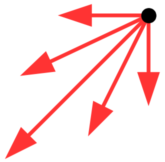
- Light sources in CG
  - Ambient light
    - Light is equal in all directions, all positions
    - A hack to simulate inter-reflections
  - Directional light
    - Light rays oriented in same direction
    - Good for distant sources (e.g., sunlight)



  - Point light
    - Light rays start at single point
    - Simulates a local source



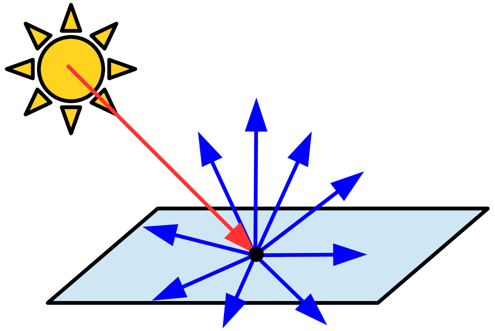
  - Spotlights: fall-off

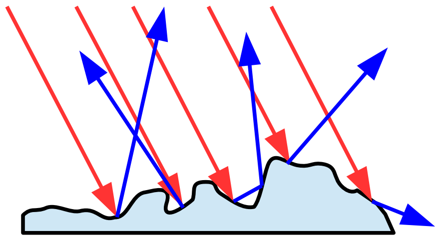


Lighting and Shading

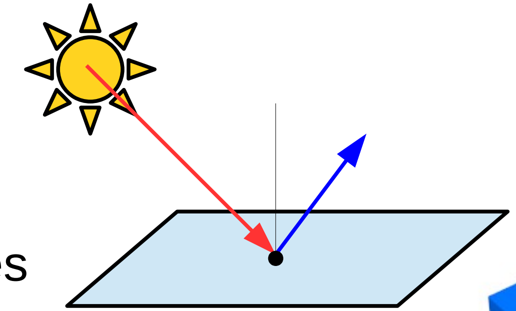
Università di Roma Tor Vergata

- Materials in CG
  - Diffuse reflection
    - Also called Lambertian reflection
    - A physical model for matte surfaces
      - rough surfaces at microscopic level



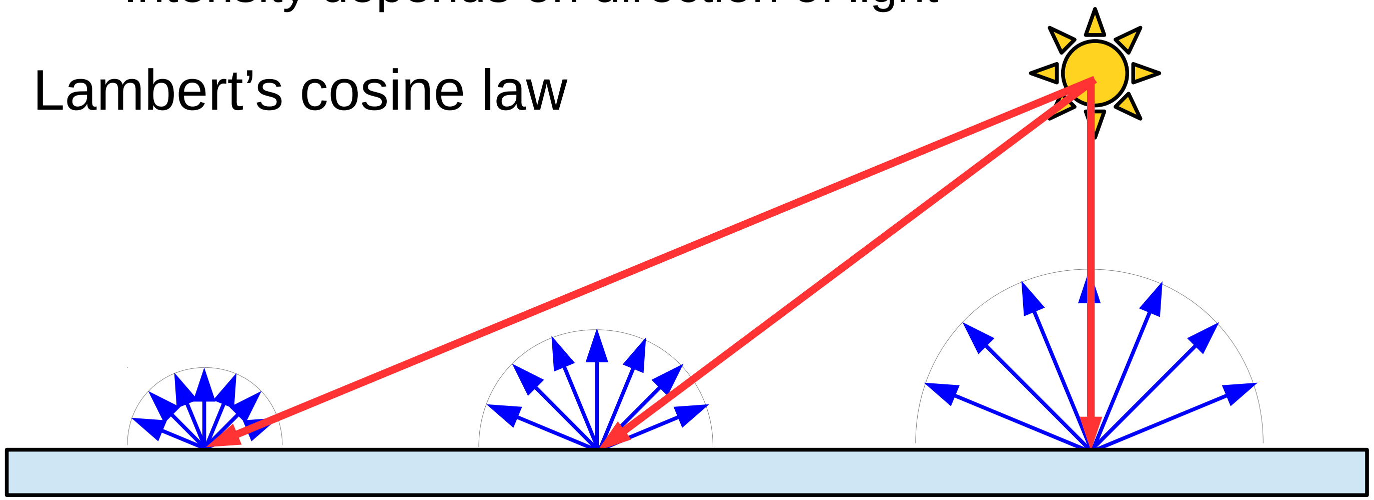


  - Specular reflection
    - Accounts for the highlight on some objects
    - Particularly important for smooth, shiny surfaces
      - e.g., metals, plastics, apples, ...



Lighting and Shading

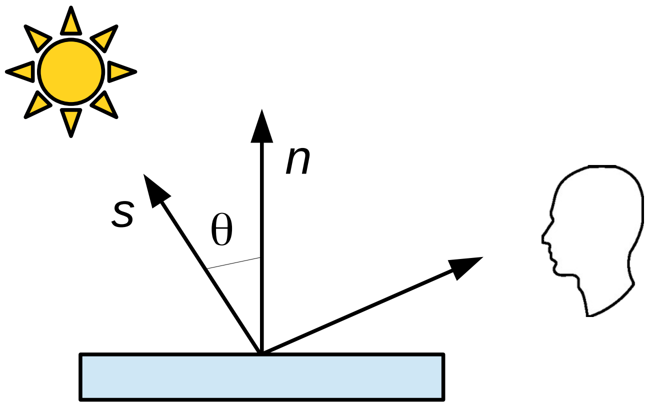
- Diffuse reflection
  - Ideal diffuse
    - Incoming light is scattered equally in all directions
    - Viewed intensity does not depend on viewing direction
    - Intensity depends on direction of light
  - Lambert’s cosine law



Lighting and Shading

- Diffuse reflection
  - Lambert's cosine law

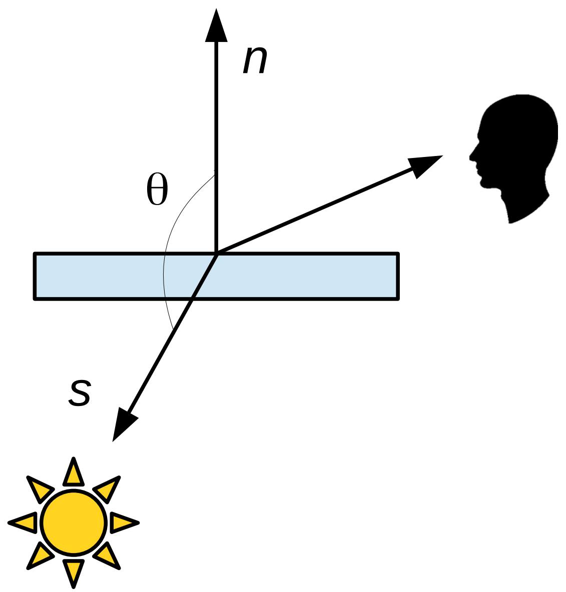
$$I_{diff} = I_{light} k_d \cos\theta = I_{light} k_d (n \cdot s)$$



- I_{light} : Light source intensity
- k_d : Diffuse reflectance coefficient of material, in [0, 1]
- θ : Angle between light ray and normal

Lighting and Shading

- Diffuse reflection
  - Lambert's cosine law



  - Light source not visible for θ > π/2

$$I_{diff} = I_{light} k_d \max(\cos\theta, 0)$$

  - Reflectance coefficient depends on wavelength
    - Usually specified as a color (RGB triple)

Lighting and Shading

- Ambient + diffuse reflection
  - Same sphere lit diffusely from different angles

  - Surfaces facing away are black: not so realistic
  - Ambient light
    - A hack to simulate (indirect) background light in the scene

$$I_{diff} = I_a k_a + I_{light} k_d \max(\cos\theta, 0)$$

Lighting and Shading

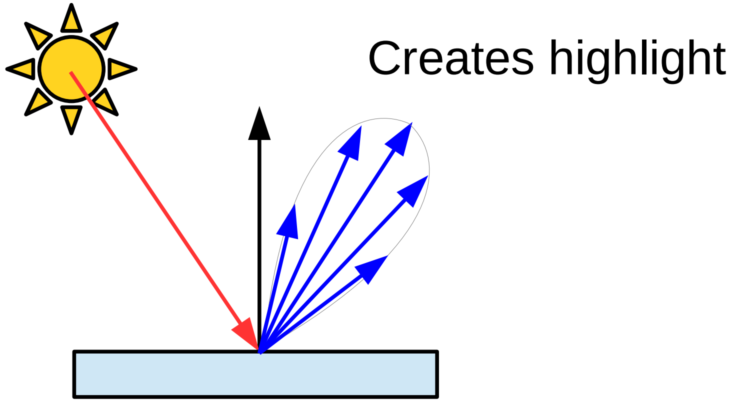
- Attenuation factor
  - Light attenuation
    - Light intensity decreases with distance
    - *d* = distance between light source and surface

$$I_{diff} = I_a k_a + f_{att} I_{light} k_d \max(\cos\theta, 0) \qquad f_{att} \sim 1/d^2$$

  - Atmospheric attenuation
    - Use viewer-to-surface distance for extra effects
    - Distance is used to blend the object's color with a "fog" color
      - Linear interpolation: d_{min} (100% object color) and d_{max} (100% fog color)

Lighting and Shading

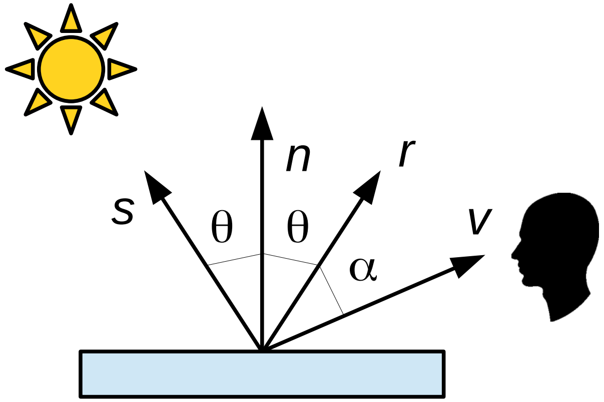
- Specular reflection
  - Shiny surfaces look different from different viewpoints
  - Light is reflected in a single direction or a “lobe”
  - Mirror is perfect specular
  - Phong reflection model
    - Approximates specular fall-off
    - No real physical basis



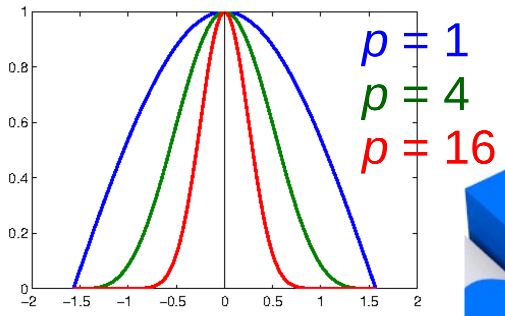
Università di Roma Tor Vergata

- Specular reflection
  - Phong reflection model

$$I_{spec} = I_{light} k_s (\cos\alpha)^p = I_{light} k_s (r \cdot v)^p$$

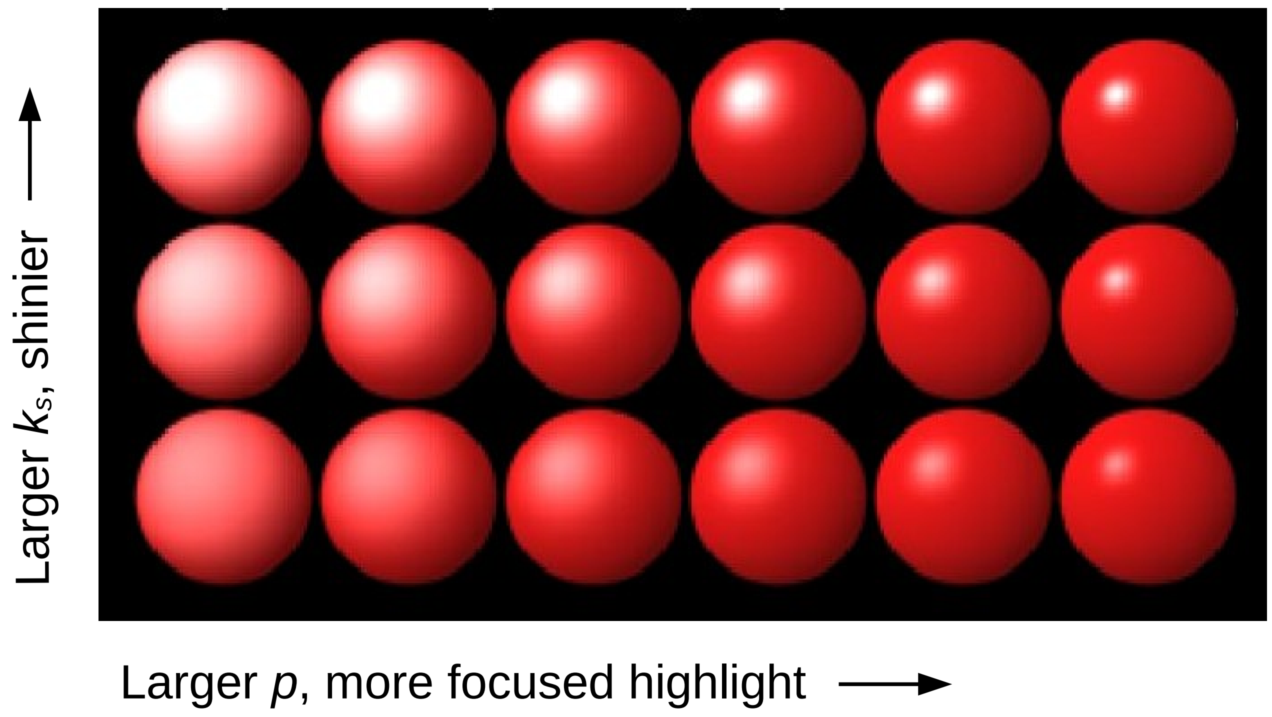


- k_s : Specular reflectance coefficient
- p : Rate of specular fall-off (Phong exponent)
  - Larger p, more focused highlight
  - Can vary from 1 … 100



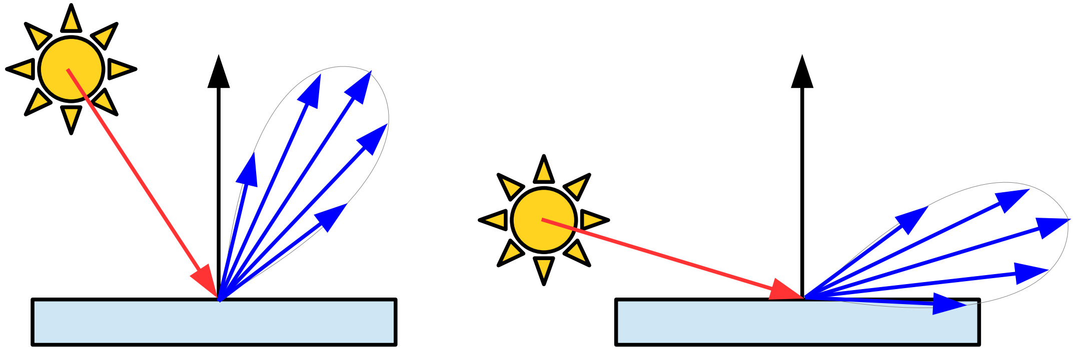
Lighting and Shading

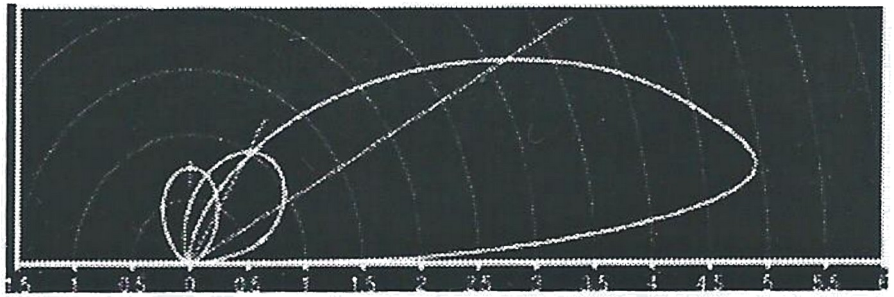
- Specular reflection
  - Phong reflection model



Lighting and Shading

- Specular reflection
  - Phong reflection model
    - Artefacts: it is just a model





Physically based model
(PCG – Cornell University)

    - Energy is not preserved
    - Maximum always in specular direction

Lighting and Shading

Università di Roma Tor Vergata

- Specular reflection
  - Other common models
    - Blinn-Phong model
      - Using halfway vector *h* (between *s* and *v*)

$$I_{spec} = I_{light} k_s (n \cdot h)^p$$

      - Represents the cosine of an angle that is half of the angle used in Phong's model if *s*, *v*, *n* and *r* all lie in the same plane
    - Cook-Torrance model
      - Based on physical parameters

Lighting and Shading

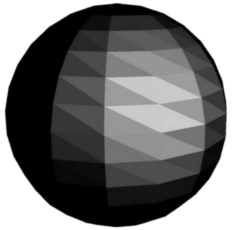
- Putting it all together
  - Combining ambient, diffuse and specular illumination

$$I = I_a k_a + f_{att} I_{light} [k_d \cos\theta + k_s (\cos\alpha)^p]$$

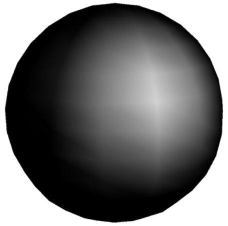
  - For multiple light sources
    - Repeat the diffuse and specular calculations for each light source
    - Add the components from all light sources
    - The ambient term contributes only once
  - Choice of different reflectance coefficients
    - Simple metal: k_a and k_d share material color, k_s is white
    - Simple plastic: k_s also includes material color

Lighting and Shading

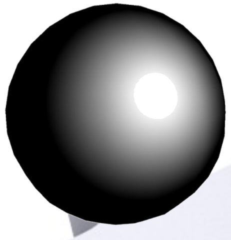
- Shading models
  - Polygonal meshes: easy to compute normals for polygons
  - Flat shading ~ per-polygon shading
    - Constant color for each polygon
    - Fast and simple, but non-smooth shading



  - Gouraud shading ~ per-vertex shading
    - Compute color at each vertex using average normals
    - Interpolate color for each interior pixel

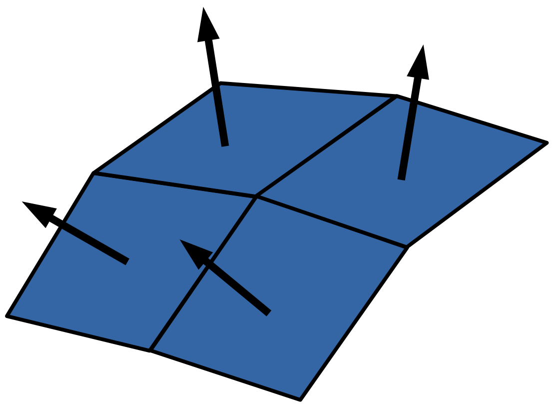


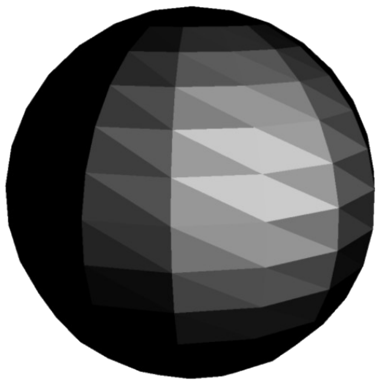
  - Phong shading ~ per-pixel shading
    - Interpolate normals instead of colors



Lighting and Shading

- Flat shading
  - Constant color for each polygon



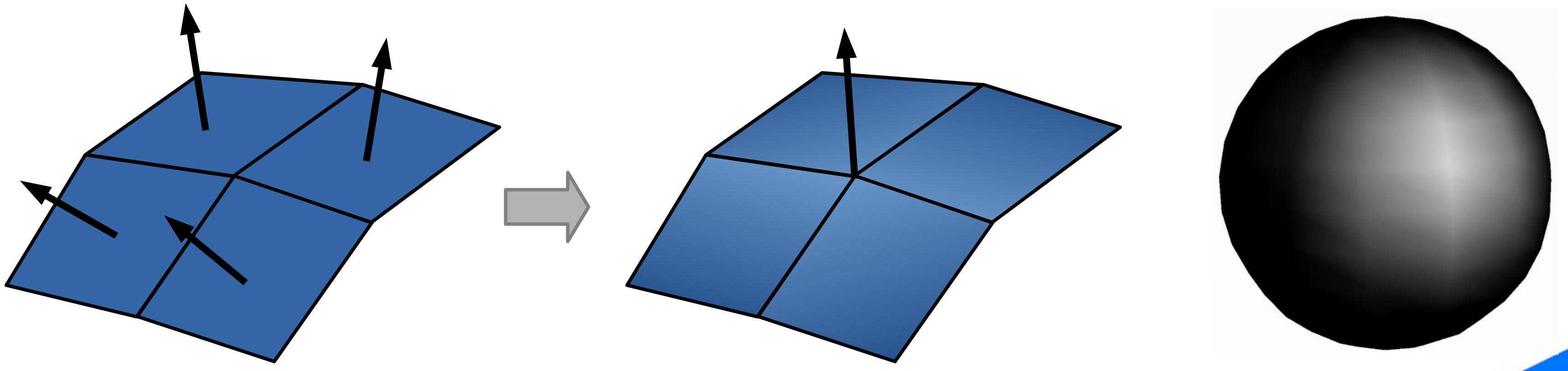


  - Fast and simple
  - Non-smooth shading is not so realistic

Lighting and Shading

Università di Roma Tor Vergata

- Gouraud shading
  - Basic idea (Henry Gouraud)
    - Compute normals at vertices as average of normals for adjacent faces
    - Compute colors at vertices, and then interpolate colors (linear) across faces

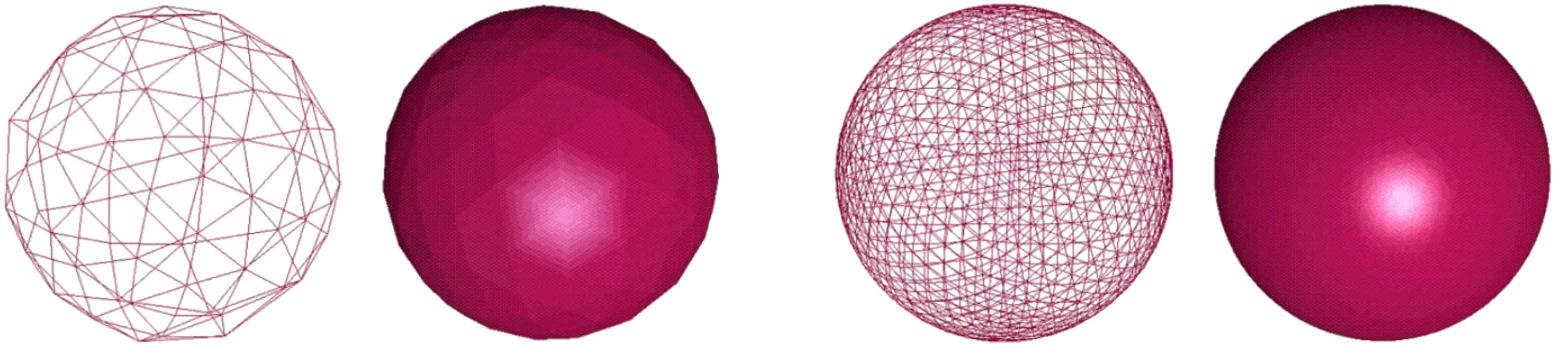


  - Still pretty fast and simple, and gives better sense of form

Lighting and Shading

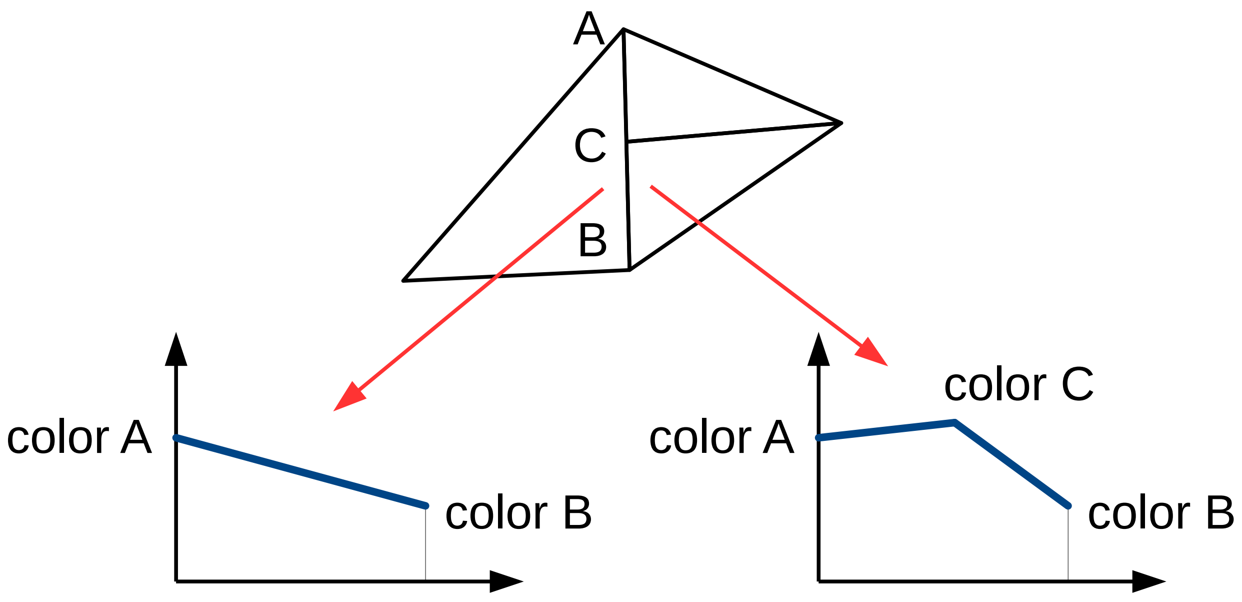
Università di Roma
Tor Vergata

- Gouraud shading
  - Problems with interpolated shading
    - Quality of highlights depends on the size of primitives
      - They tend to spread out at the vertices
      - They disappear in the middle area of polygons



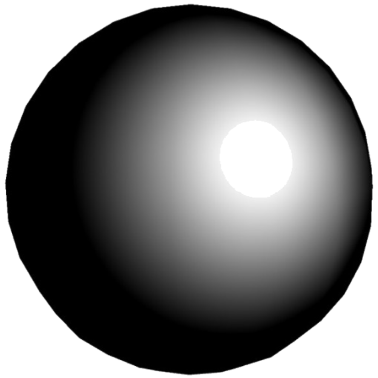
Lighting and Shading

- Gouraud shading
  - Problems with interpolated shading
    - T-vertices: visual discontinuity in colors



Lighting and Shading

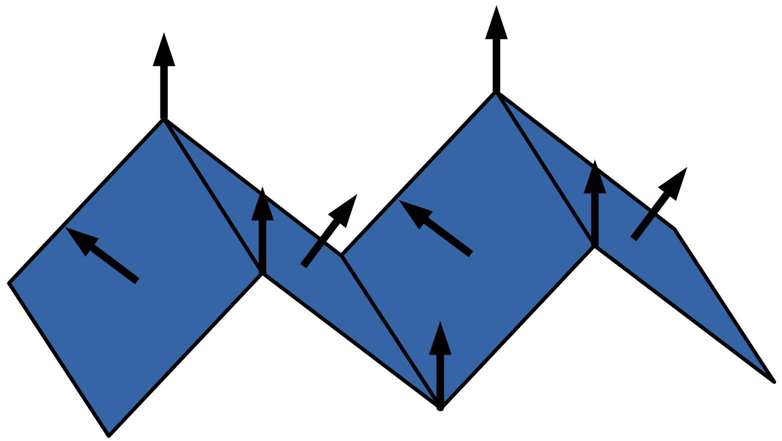
- Phong shading
  - Basic idea (Bui Tuong Phong)
    - Interpolate normals before computing colors
    - This is not Phong reflection!



  - Traditional pipeline cannot handle this
    - Interpolation needs to be done before perspective transform
    - But … recent hardware provides per-pixel capabilities

Lighting and Shading

- Phong shading
  - Results are much improved over Gouraud
    - Highlights are better visualized
    - Harder to tell low- from high-polygon models
  - Still problems with interpolated normals
    - Regular meshes: all vertex normals can be parallel



Lighting and Shading

- Some limitations of classical (real-time) models
  - No light that reflects off one object and hits another
  - No refraction of light through translucent materials
  - No shadows
- A lot of hacks available
  - Texture and bump mapping
    - The color of a point can be specified by a pre-defined image-map
    - The normal can be perturbed by a pre-defined bump-map
  - Shadow buffering
    - Store which objects are lighted in a scene, and use during rendering

Lighting and Shading

- Shadow buffering
  - Pre-process the shadow buffer
    - Render scene as seen from light source
    - Store depth of each pixel in shadow buffer (~ *Z*-buffer)
  - Compare depths when rendering
    - If depth is larger: point is in shadow
    - If depth is equal: point is not in shadow
  - Is available in OpenGL