

Computer Graphics and Image Processing Illumination II

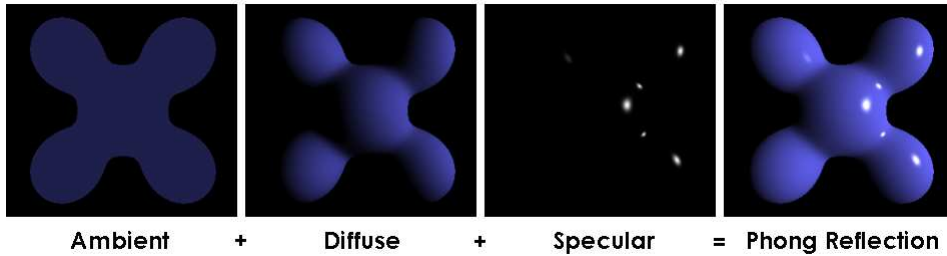
Part 1 – Lecture 8

1

Today's Outline

- Recap: Phong Illumination Model
- Shading Algorithms
 - Flat Shading
 - Gouraud Shading
 - Phong Shading
- Shadows
 - Ground-Plane Projection
 - Shadow Buffer

2



Ambient + Diffuse + Specular = Phong Reflection

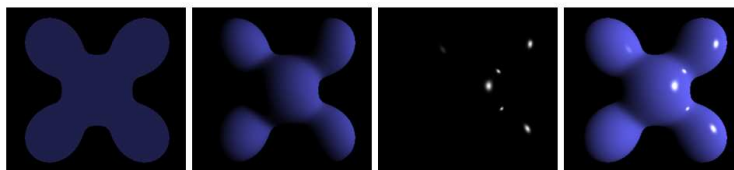
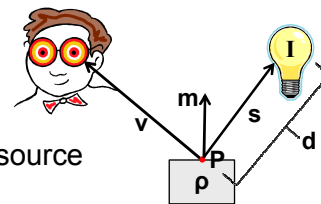
RECAP: PHONG ILLUMINATION MODEL

Image thanks to Brad Smith

3

Phong Illumination Model

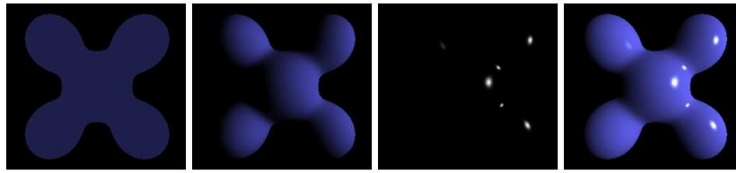
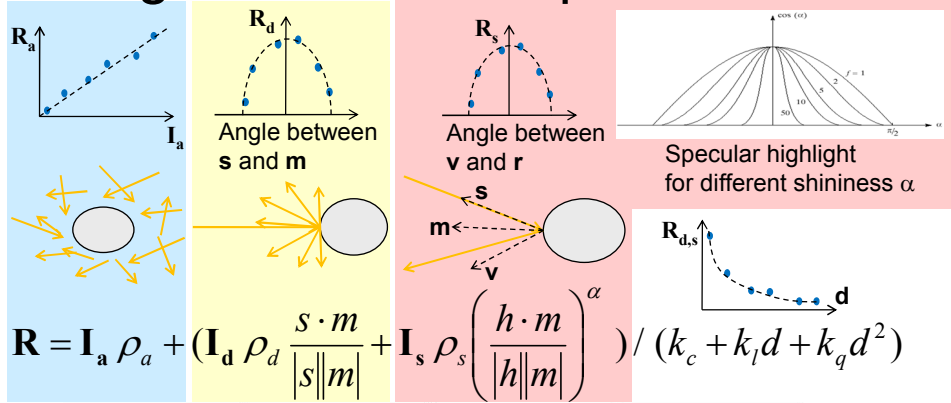
- **Idea:** calculate intensity R (and color) of visible light at a point as the sum of ambient, diffuse and specular reflection
- Variables taken into account:
 - Intensities I_a , I_d , I_s for incident light
 - Surface normal vector \mathbf{m}
 - Vector \mathbf{s} describing the direction to the light source
 - Distance d to light source
 - Vector \mathbf{v} describing the direction to the viewer
 - Reflection coefficients of the surface material ρ_a , ρ_d , ρ_s



Ambient + Diffuse + Specular = Phong Reflection

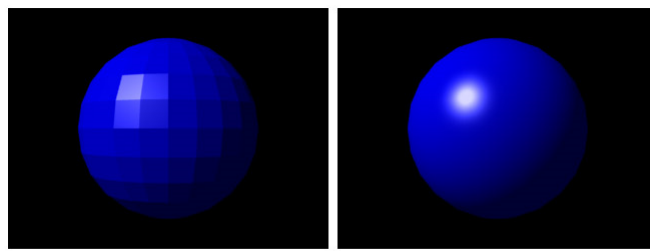
4

Phong Illumination Equation



Ambient + Diffuse + Specular = Phong Reflection

5



FLAT SHADING

PHONG SHADING

SHADING ALGORITHMS

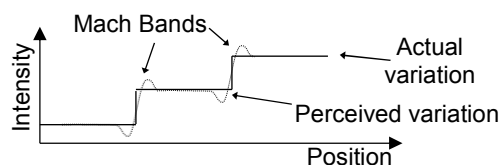
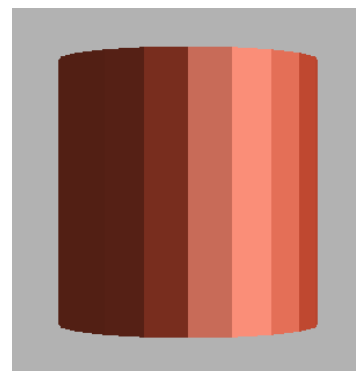
6

Shading Algorithms

- Phong illumination equation:
how to calculate color on every point of surface
(given lights, materials, etc.)
- **Problem:** calculating Phong equations at every single point
(pixel) would be slow!
- **Solution:** use a shading algorithm
 - Uses Phong equation only at some points (usually vertices)
 - Then uses interpolation to get colors for in-between points (in-between pixels)
- Three popular shading algorithms:
 - Flat shading (fastest but worst quality)
 - Gouraud shading (balance of speed and quality)
 - Phong shading (slowest but best quality)

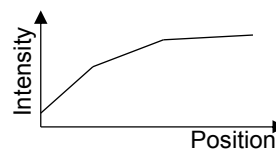
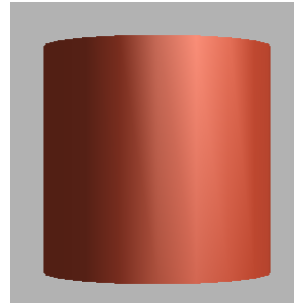
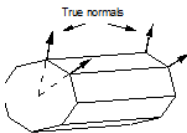
Flat Shading (Constant Shading)

- Apply Phong equation once per face
(using face normal)
- Shade whole face that color
- **Advantage:** simple and fast
- **Disadvantage:** very poor display of
polygon-mesh approximations to
curved surfaces
 - Human eye very sensitive to
discontinuities
 - Exaggerates them
into *Mach Bands*



Gouraud Shading

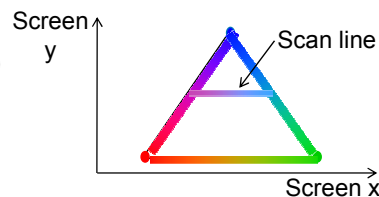
- Apply Phong equation at each vertex (using “true” surface normal)
- Linearly interpolate colors between vertices
- Advantages:
 - Still fast
 - Avoids 0th-order color discontinuities over polygon mesh (color continuous between faces)
- Disadvantages:
 - Still 1st order color discontinuity (→ slight Mach bands)
 - Invariance problem with quadrilaterals
 - Problems with highlights



Gouraud Shading Contd.

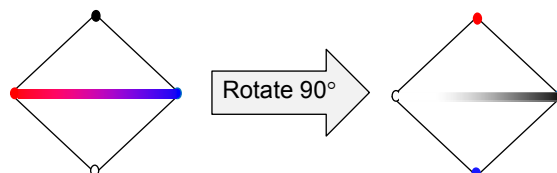
Triangles

1. Get color for each vertex (Phong equation)
2. Interpolate pixel colors between vertices
3. Interpolate pixel colors along all horizontal scan lines



Quadrilaterals

Problem: not rotationally invariant



When rotating the quad, the color of the middle pixel changes (first purple, then gray)

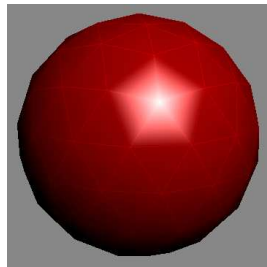
Solution: cut each quadrilateral into two triangles

Gouraud Shading: Highlights

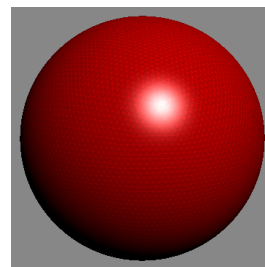
Problem: highlights can only be rendered on a vertex

- Highlight may not be sharp, i.e. gets smeared over adjacent faces
- Highlight may not be visible if not near a vertex

Solution: use more vertices in your mesh



Low number of vertices



High number of vertices

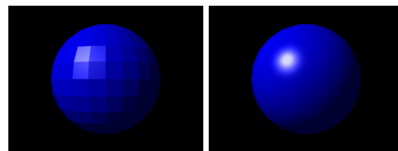


Images thanks to Zom-B

11

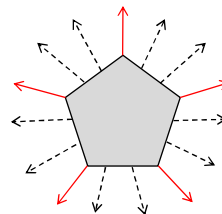
Phong Shading

- To get crisp specular highlights with Gouraud shading, we need many vertices
- Bui Tuong-Phong suggested Phong shading to solve this
 1. Linearly interpolate the **normal** over the polygon (instead of **color** as in Gouraud shading)
 2. Then evaluate Phong equation at each pixel



FLAT SHADING

PHONG SHADING



- **Advantage:** crisp highlights with few vertices
- **Disadvantage:** slower because Phong calculation for every Pixel

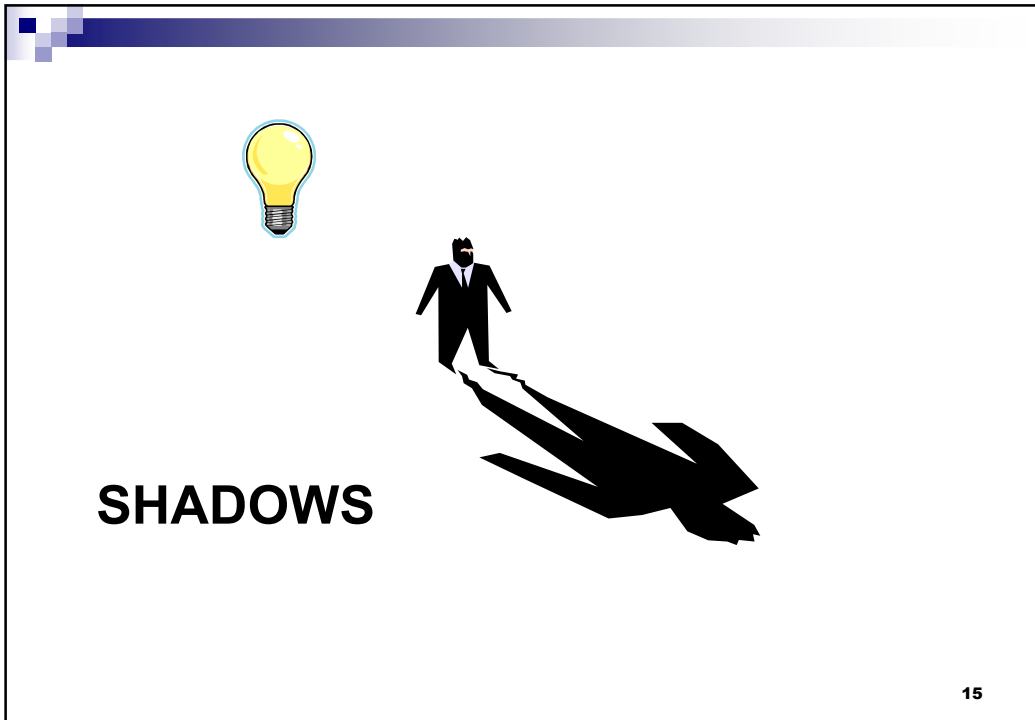
Cost of Shading



- **Flat shading:**
 - Pixel colors constant for entire triangle
 - 1 normal calculation per triangle
 - 1 color calculation per triangle (Phong equation)
- **Gouraud shading:**
 - 1 normal calculation per vertex
 - 1 color calculation per vertex (Phong equation)
 - 1 color interpolation calculation per pixel
- **Phong shading:**
 - 1 normal calculation per vertex
 - 1 normal interpolation between vertex normals per pixel
 - 1 color calculation per pixel (Phong equation)

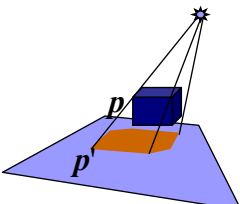
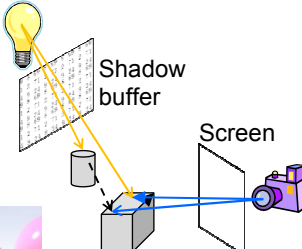
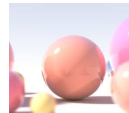
Shading Algorithms

Flat Shading	Gouraud Shading	Phong Shading
		
<p>Simple and fast Phong equation only once per face</p>	<p>Still fast Phong equation at each vertex No 0th-order color discontinuities</p>	<p>Crisp highlights with few vertices</p>
<p>Mach Bands</p>	<p>Slight mach bands, Color invariance with quadrilaterals, Problems with highlights</p>	<p>Slow Phong calculation for every Pixel</p>



How to Render Shadows?

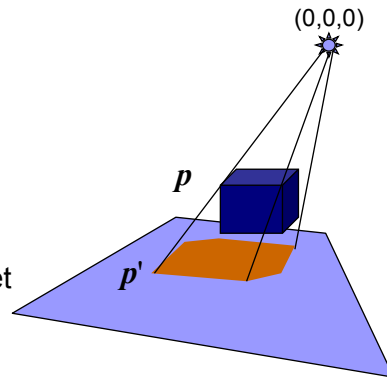
- **Where?**
Points that can be seen but are not illuminated by a particular light source
- **How?** Several possibilities...
 1. **Ground-plane projection:**
Draw shadows of objects as separate (flat and dark) objects onto a plane (fast but limited possibilities)
 2. **Shadow buffer:** Efficient way to determine if a visible point is illuminated by a particular light source
 3. **Ray tracing:** trace the path of light rays (slow but high quality)

16

Plane Projection Transformation

1. Assume light source is at origin
2. Line from light source through \mathbf{p} is
 $\mathbf{q}(t) = t \mathbf{p}$
3. Let plane be $ax+by+cz+d = 0$
4. Then at \mathbf{p}' , $\mathbf{q}(t)$ intersect the plane:
 $a t p_x + b t p_y + c t p_z + d = 0$
5. Solve for t , calculate $\mathbf{q}(t)=\mathbf{p}'$ and hence get
 $\mathbf{p}' = -d (p_x, p_y, p_z) / (a p_x + b p_y + c p_z)$
 $= -d \mathbf{p} / (a p_x + b p_y + c p_z)$



© 2004 Lewis Hitchner, Richard Lobb & Kevin Novins

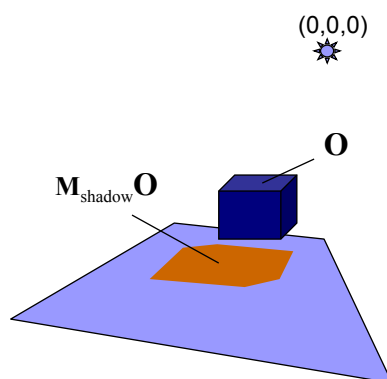
Plane Projection Transformation Contd.

$$\mathbf{p}' = -d \mathbf{p} / (a p_x + b p_y + c p_z)$$

can be written as:

$$\mathbf{p}' = \begin{pmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} -d p_x \\ -d p_y \\ -d p_z \\ a p_x + b p_y + c p_z \end{pmatrix}$$

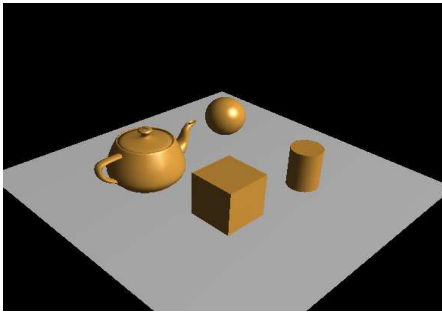
$$= \mathbf{M}_{\text{shadow}} \mathbf{p}$$



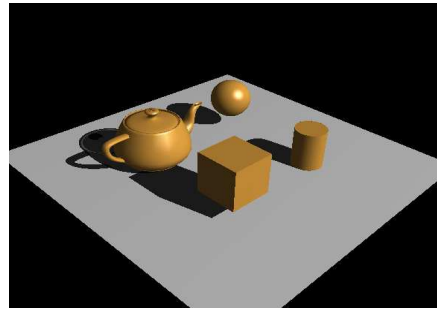
- Do division by converting to ordinary coordinates.
- Applying $\mathbf{M}_{\text{shadow}}$ to an object yields its planar projection onto the given plane (with center of projection at origin)

Ground-Plane Projection Example

Demo program, [LightAndShadows](http://www.cs.auckland.ac.nz/compsci372s2c/christofLectures/LightAndShadowsNET.zip), available in 372 Lecture Notes web page,
<http://www.cs.auckland.ac.nz/compsci372s2c/christofLectures/LightAndShadowsNET.zip>



No shadows
(can't see "floating" objects)



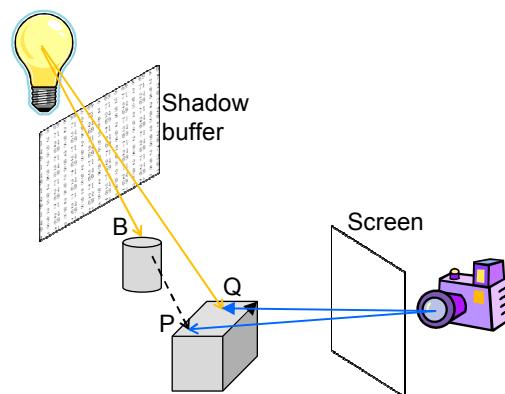
Shadows

© 2004 Lewis Hitchner, Richard Lobb & Kevin Novins

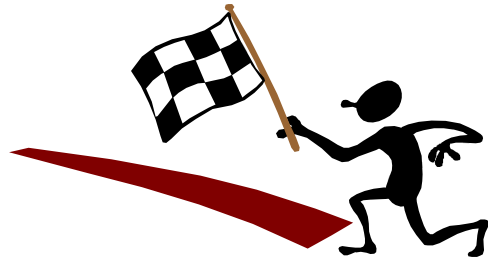
Shadow Buffer

Idea: points that are hidden from the light source are in shadow

- Calculate depth buffer from light source position (shadow buffer), i.e. values for distance between light and closest object
- For each screen pixel pointing to a point P:
 1. Get depth d_p from light source to P
 2. Find element $d[i,j]$ in shadow buffer that points towards P
 3. If $d[i,j] < d_p$ then draw only ambient light (shadow), otherwise full illumination



Advantage:
shadows can be cast from all objects onto all other objects



SUMMARY

21

Summary

- **Flat shading:** one color calculation per face
- **Gouraud shading:**
one color calculation per vertex, interpolate over faces
- **Phong shading:**
interpolate vertex normals and calculate color for every pixel
- Project objects from light sources onto planes to get simple shadow effect
- Use **shadow buffer** to detect covered points for better shadows

References:

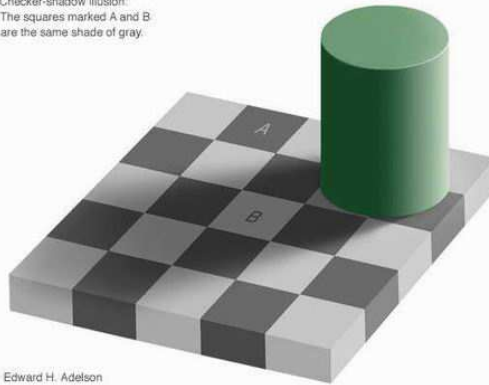
- Shading Algorithms: Hill, Chapter 8.3
- Shadows: Hill, Chapter 8.6

22

Quiz

1. Describe one disadvantage of Flat shading.
2. Why can Gouraud shading render a highlight only on a vertex?
3. What is a shadow buffer?
4. How can we use a shadow buffer to render shadows?

Checker-shadow illusion:
The squares marked A and B
are the same shade of gray.



The squares marked
A and B are the same
shade of gray.

Edward H. Adelson