CSC2503 Tutorial 1

Peter O'Donovan

September 19, 2011

Matlab Introduction

- CSLab/CDF/Research Group
- Matlab Primer
- IDE
- Paths (addpath, IDE)
- startup.m
- **Tutorials**

Matlab Basics

- Matrices
- Slicing/Colon notation
- plot
- repmat
- Backslash for solving linear systems
- reshape

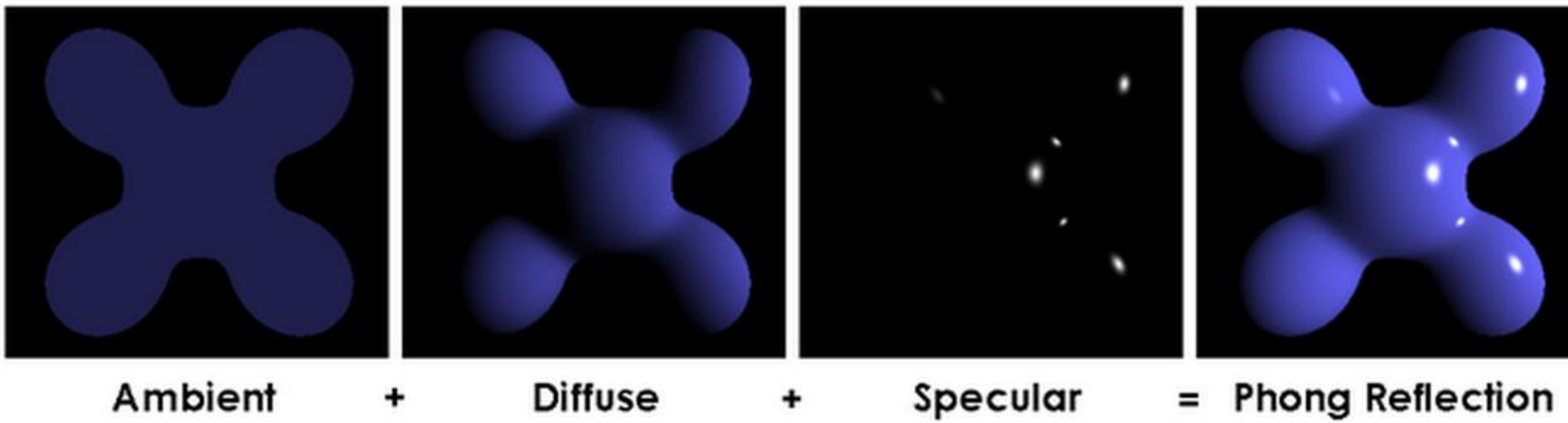
More Matlab

- Use the mathworks documentation site
  - Matlab has tons of builtin functions
- Avoid loops!
  - Beginner caveat
- Functions

Phong Model

Phong Lighting

Wikipedia

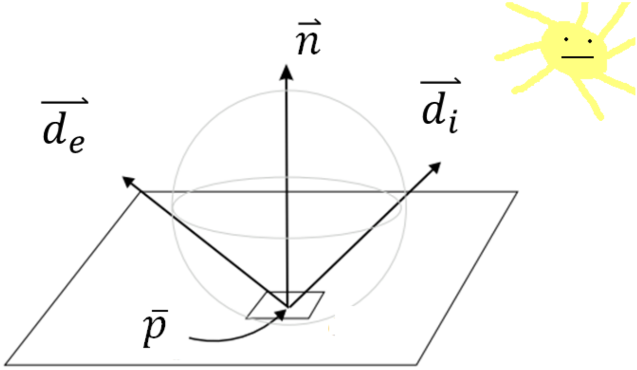


Local Illumination Model:
- Illumination only depends on local surface properties
- Light doesn’t bounce
- No inter-reflections
- No shadows

Diffuse/Lambertian Term

- Radiance from a surface patch to a camera
  – For a single light source

$$L(\bar{p}, \vec{d}_e) = \rho(\vec{d}_e,\ \vec{d}_i)\ I\ \frac{\vec{n} \cdot \vec{d}_i}{r^2}$$



  – **p**: patch location, **d_e**: camera direction, I: radiant intensity, **n**: surface normal, **d_i**: incident light direction, r: distance from patch to light
  – Looks time consuming
    - Have to integrate over all incident directions **d_i** ☹

Keep it simple

- Assume a constant BRDF
  - Albedo doesn't change based on view/light direction
  - True or false for some materials. More later…

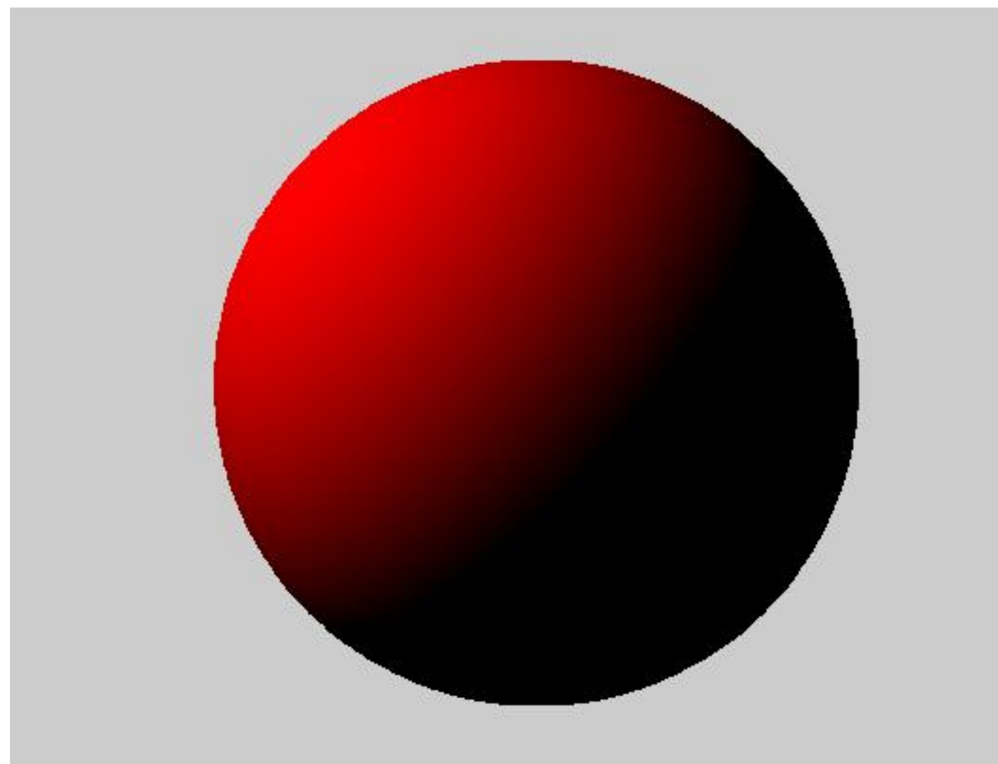
$$L_d(\bar{p}, \vec{d}_e) = \rho_0 \, I \, \frac{\vec{n} \cdot \vec{d}_i}{r^2}$$

- Assume distant point light & camera
  - 1/r^2 now effectively constant
  - Directions **d_e**, **d_i** are now constants
  - No dependence on surface location (only **n**)
  - No dependence on camera position

$$L_d(\bar{p}) = r_d \, I \, \vec{s} \cdot \vec{n}$$

Actually…

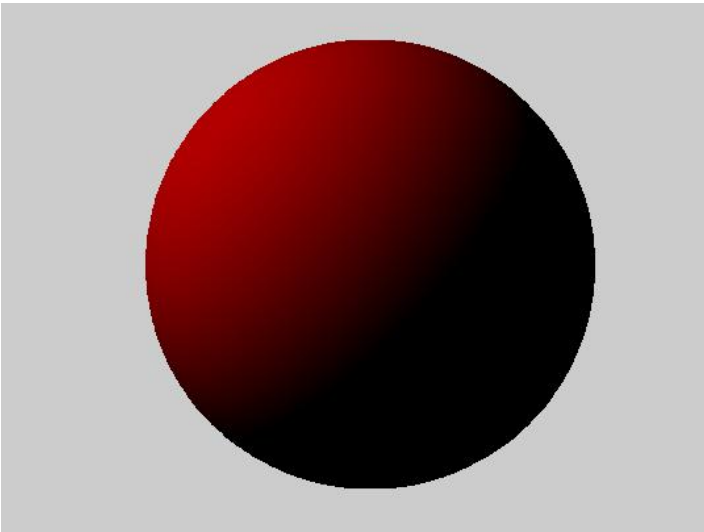
$$L_d(\bar{p}) = r_d \, I \, \max(0, \vec{s} \cdot \vec{n})$$



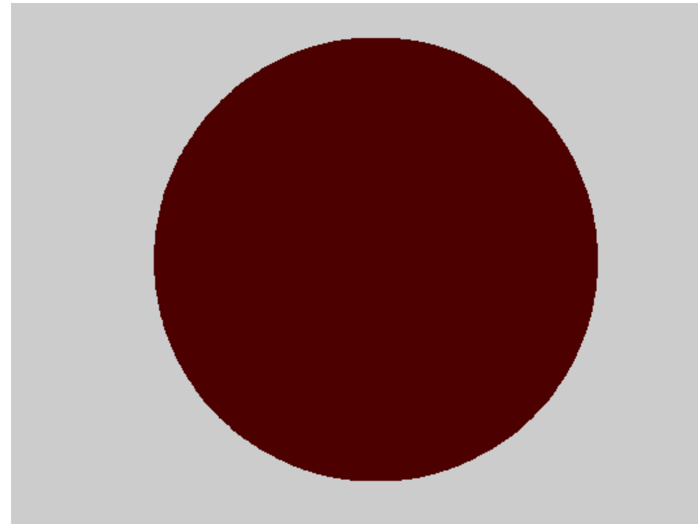
Ambient Term

- Lambertian local illumination is very unrealistic
  - Light bounces around a lot
  - Look under your desk…
- A simple hack
  - Add a constant term which approximates all this light
  - For every surface point, assume the same homogeneous irradiance from the entire hemisphere of incident directions.
  - Doesn’t depend on normals, lights, camera direction…

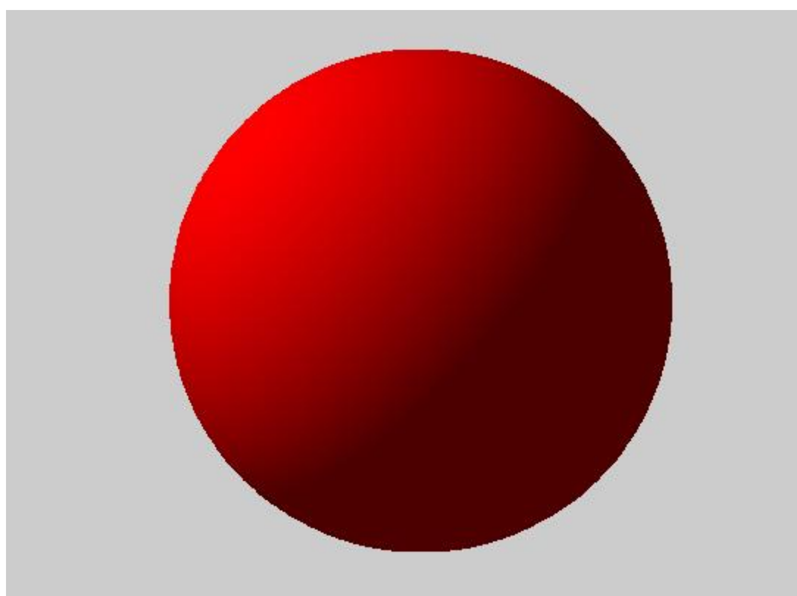
Diffuse



Ambient

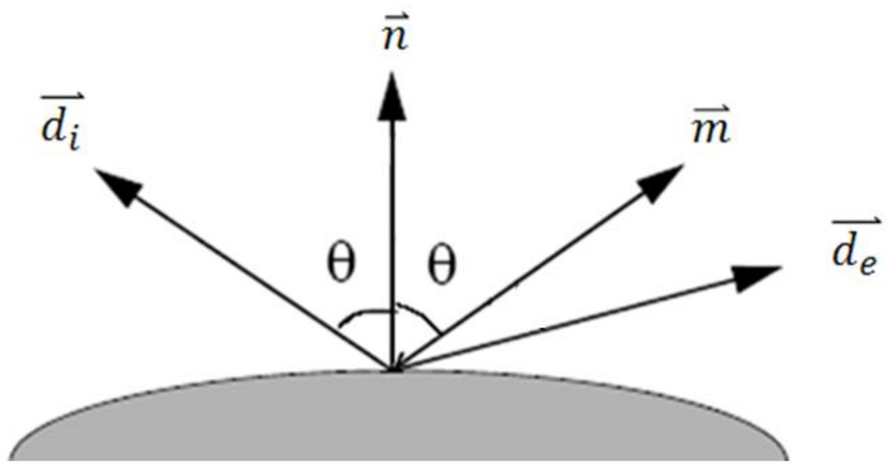


Diffuse+Ambient



Specular highlights are common on many materials

Specular Term

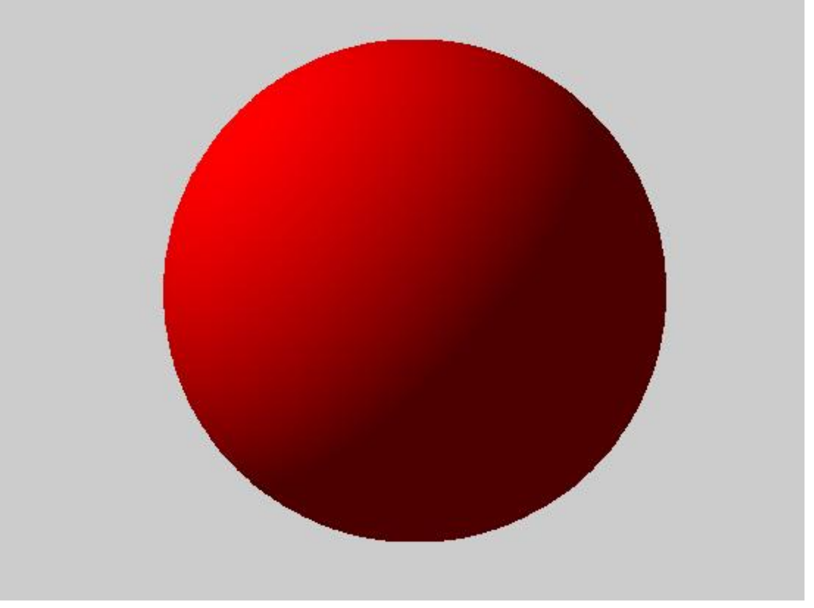


Incident light **d_i** from gets reflected to a perfect mirror direction **m**, **d_e** is the direction to the viewer

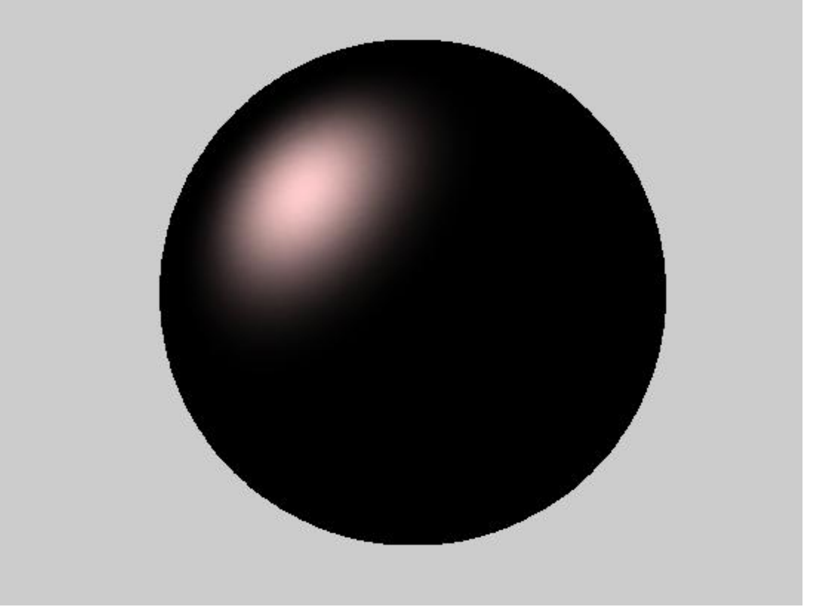
$$L_s(\vec{d_e}) = r_s I \max(0, \vec{m} \cdot \vec{d_e})^\alpha$$

The alpha parameter indicates the falloff from the perfect mirror direction

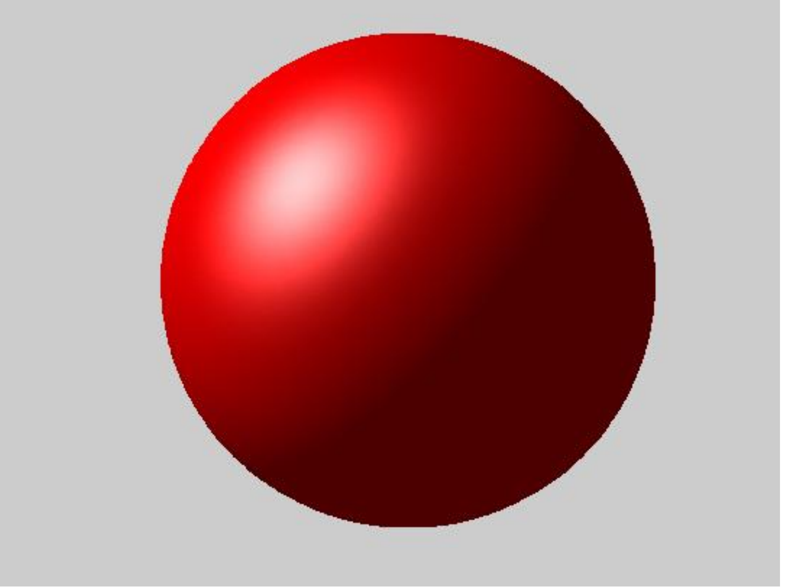
Diffuse+Ambient



Specular



Diffuse+Ambient+Specular



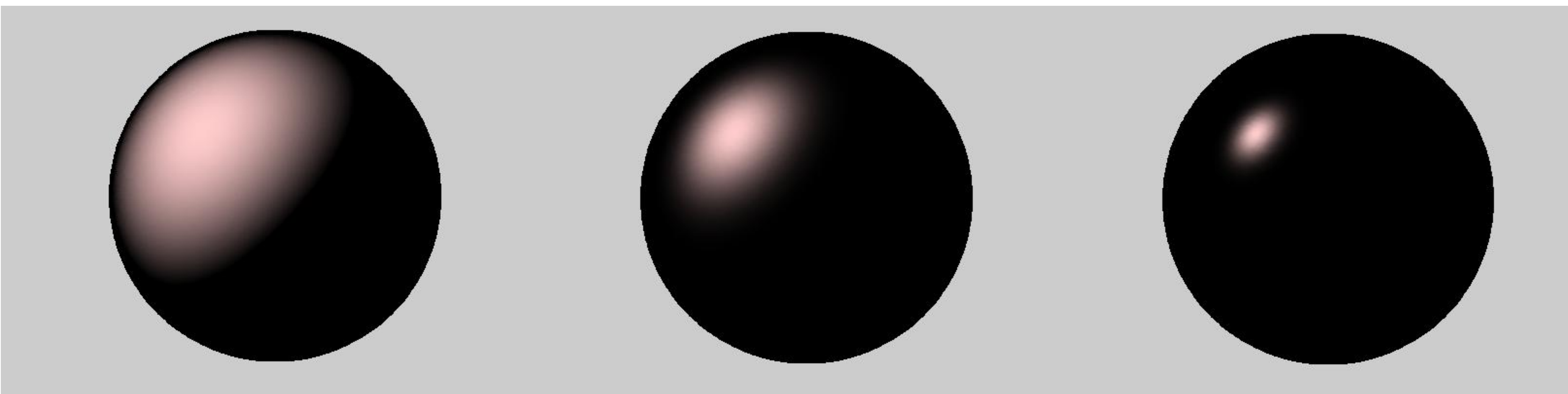
phongdemo.m

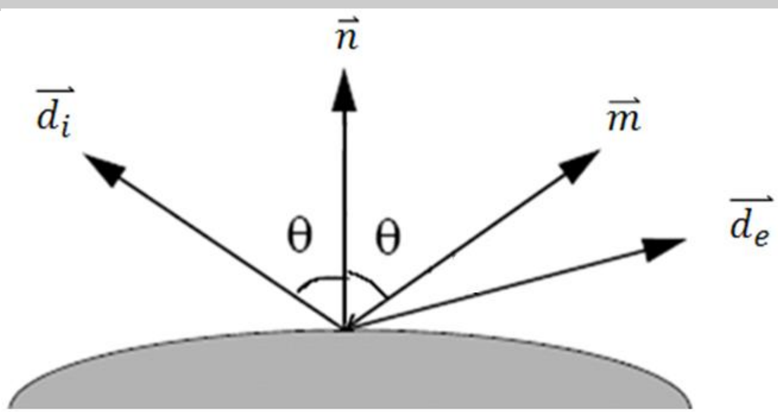
- Show phongdemo.m

- Show debugging

Modifying ke parameter in phongDemo.m

α= 1 α= 5 α= 30





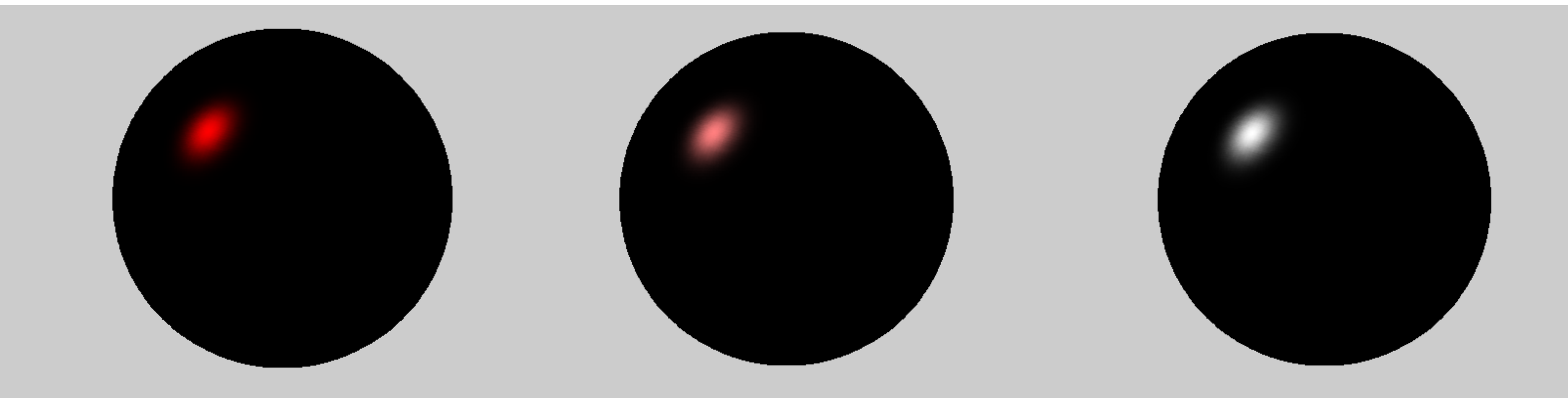
$$L_s(\vec{d_e}) = r_s I \max(0, \vec{m} \cdot \vec{d_e})^{\alpha}$$

Modifying scr parameter in phongDemo.m

scr= 0

scr= 0.5

scr= 1



$$L_s(\vec{d}_e) = r_s I \max(0, \vec{m} \cdot \vec{d}_e)^{\alpha}$$

Reflected color (r_s) is a combination of reflected light color and surface color

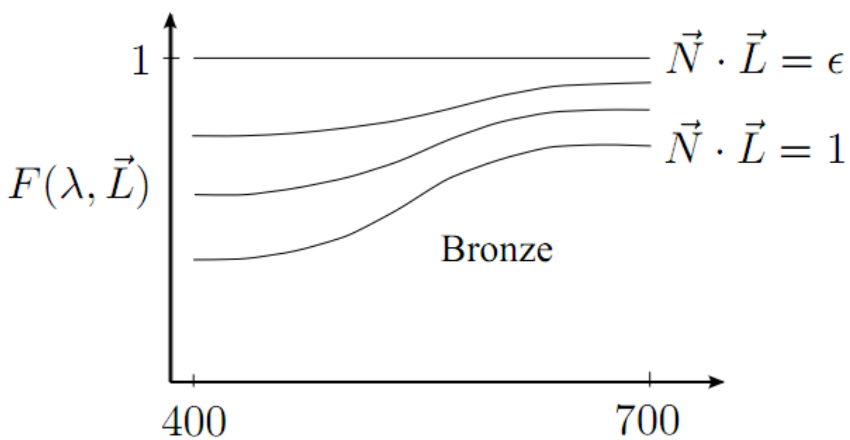
Fresnel Effect/Highlight Colors

- Non-constant BRDF $\rho(\vec{d}_e, \vec{d}_i)$
- How much incident light gets emitted is based on both angles

Notice how the color changes from greenish to orange/yellow depending on view angle

Fresnel Effect/Highlight Colors

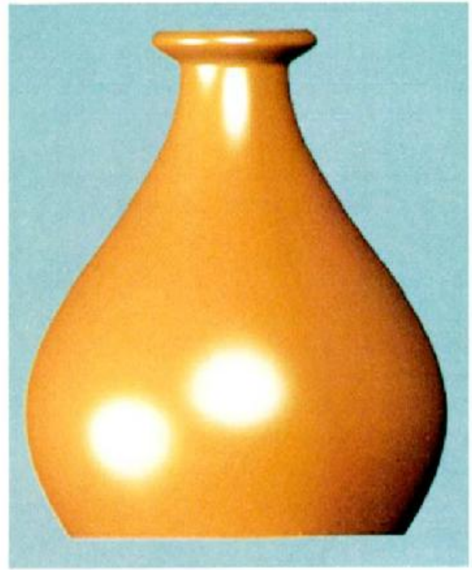
- How do we get this function?



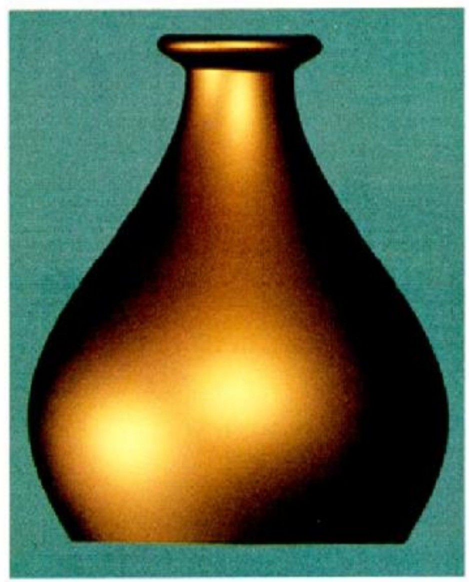
Parametric approximation to a real BDRF

Real measurements of a material/object

Plastic



Bronze



Cook and Torrance

Photometric Stereo

- Assume Lambertian lighting w/ distant source
  - Can we use light intensity to determine surface properties (like the albedo or normal)?

$$L_d(\bar{p}) = r_d \, I \, \vec{s} \cdot \vec{n}$$

Photometric Stereo

- Assume Lambertian lighting w/ distant source
  - Can we use light intensity to determine surface properties (like the albedo or normal)?

$$L_d(\bar{p}) = r_d \, I \, \vec{s} \cdot \vec{n}$$

  - Even if we know the light source direction **s**, then we have one equation in 3 unknowns (albedo and normal direction)

Photometric Stereo

- What if we had three images, with different light source locations?

Photometric Stereo

- What if we had three images, with different light source locations?
  - For each pixel we have three constraints and we can solve a linear system for **n** and a

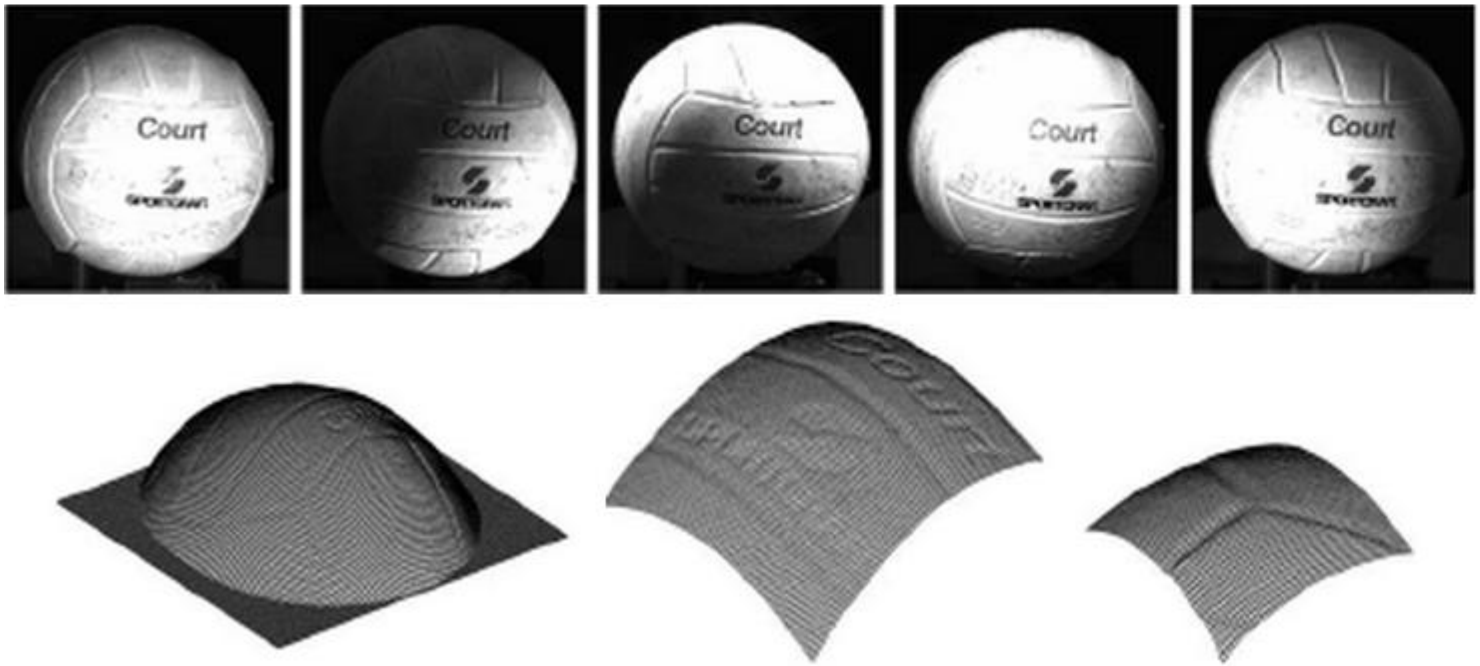
$$L_1 = \mathrm{a}\ (\overrightarrow{s_1} \cdot \vec{n}),\ L_2 = \mathrm{a}\ (\overrightarrow{s_2} \cdot \vec{n}),\ L_3 = \mathrm{a}\ (\overrightarrow{s_3} \cdot \vec{n})$$

Photometric Stereo

- What if we had three images, with different light source locations?
  – For each pixel we have three constraints and we can solve a linear system for **n** and a

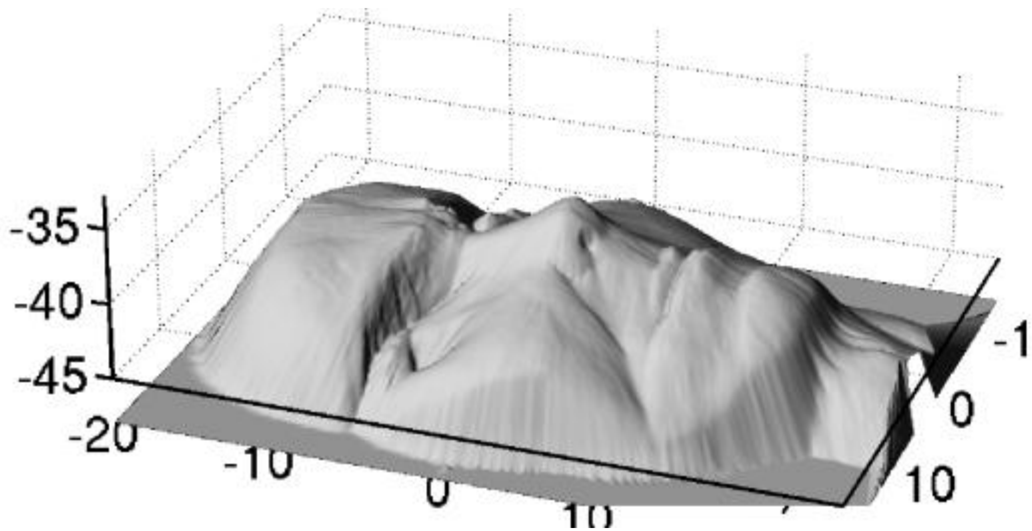
$$L_1 = a\,(\overrightarrow{s_1} \cdot \vec{n}),\ L_2 = a\,(\overrightarrow{s_2} \cdot \vec{n}),\ L_3 = a\,(\overrightarrow{s_3} \cdot \vec{n})$$

- More images would be better
  – Overconstrained linear system, but real images are noisy…


Court
Court
Court
Court
Court

Aside: Shape from Shading

- Can we find the normal/albedo from a **single image**?



- Underconstrained problem
  – Notice dark eyes & eyebrows vs side of head