



Tamara Munzner

Lighting/Shading II

Week 6, Fri Feb 16

<http://www.ugrad.cs.ubc.ca/~cs314/vjan2007>

Correction/News

- Homework 2 was posted Wed
 - due Fri Mar 2
- Project 2 out today
 - due Mon Mar 5

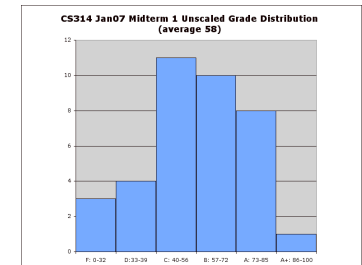
2

News

- midterms returned
- project 2 out

3

Midterm Grading



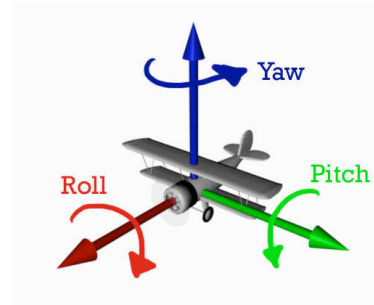
4

Project 2: Navigation

- five ways to navigate
 - Absolute Rotate/Translate Keyboard
 - Absolute Lookat Keyboard
 - move wrt global coordinate system
 - Relative Rolling Ball Mouse
 - spin around with mouse, as discussed in class
 - Relative Flying
 - Relative Mouselook
 - use both mouse and keyboard, move wrt camera
- template: colored ground plane

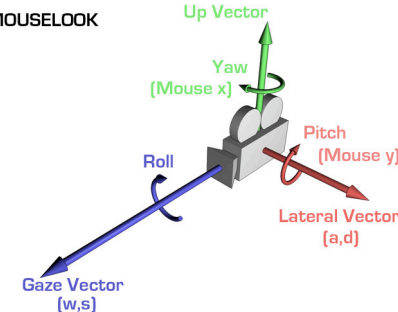
5

Roll/Pitch/Yaw



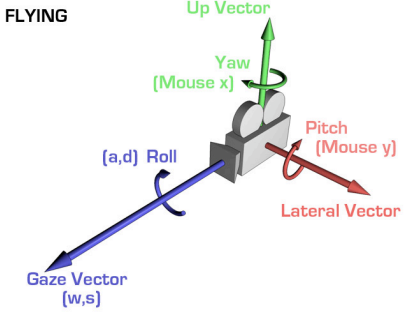
6

MOUSELOOK



7

FLYING



8

Demo

9

Hints: Viewing

- don't forget to flip y coordinate from mouse
 - window system origin upper left
 - OpenGL origin lower left
- all viewing transformations belong in modelview matrix, not projection matrix

10

Hint: Incremental Relative Motion

- motion is wrt current camera coords
 - maintaining cumulative angles wrt world coords would be difficult
 - computation in coord system used to draw previous frame (what you see!) is simple
 - at time k, want $p' = I_{k,k-1} \dots I_{2,1} p_1 C_p$
 - thus you want to premultiply: $p' = C_p$
 - but postmultiplying by new matrix gives $p' = C_p$
 - OpenGL modelview matrix has the info! sneaky trick:
 - dump out modelview matrix with `glGetDoublev()`
 - wipe the stack with `glLoadIdentity()`
 - apply incremental update matrix
 - apply current camera coord matrix
 - be careful to leave the modelview matrix unchanged after your display call (using push/pop)

11

Caution: OpenGL Matrix Storage

- OpenGL internal matrix storage is columnwise, not rowwise

a	e	i	m
b	f	j	n
c	g	k	o
d	h	l	p

 - opposite of standard C/C++/Java convention
 - possibly confusing if you look at the matrix from `glGetDoublev()`!

12

Reading for Wed/Today/Next Time

- FCG Chap 9 Surface Shading
- RB Chap Lighting

13

Review: Computing Barycentric Coordinates

- 2D triangle area
 - half of parallelogram area
 - from cross product
- $$A = A_{P_1} + A_{P_2} + A_{P_3}$$
- $$\alpha = A_{P_1} / A$$
- $$\beta = A_{P_2} / A$$
- $$\gamma = A_{P_3} / A$$
- weighted combination of three points [demo]
-

14

Review: Light Sources

- directional/parallel lights
 - point at infinity: $(x,y,z,0)^T$
- point lights
 - finite position: $(x,y,z,1)^T$
- spotlights
 - position, direction, angle
- ambient lights



15

Lighting I




16

Light Source Placement

- geometry: positions and directions
- standard: world coordinate system
 - effect: lights fixed wrt world geometry
 - demo: <http://www.xmission.com/~nate/tutors.html>
- alternative: camera coordinate system
 - effect: lights attached to camera (car headlights)
- points and directions undergo normal model/view transformation
- illumination calculations: camera coords



17

Types of Reflection

- *specular* (a.k.a. *mirror* or *regular*) reflection causes light to propagate without scattering. 
- *diffuse* reflection sends light in all directions with equal energy. 
- *mixed* reflection is a weighted combination of specular and diffuse. 

18

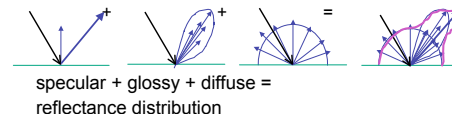
Types of Reflection

- *retro-reflection* occurs when incident energy reflects in directions close to the incident direction, for a wide range of incident directions. 
- *gloss* is the property of a material surface that involves mixed reflection and is responsible for the mirror like appearance of rough surfaces. 

19


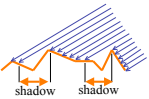

Reflectance Distribution Model

- most surfaces exhibit complex reflectances
 - vary with incident and reflected directions.
 - model with combination



20

Surface Roughness

- at a microscopic scale, all real surfaces are rough 
- cast shadows on themselves 
- "mask" reflected light: 

21

Surface Roughness

- notice another effect of roughness:
 - each "microfacet" is treated as a perfect mirror.
 - incident light reflected in different directions by different facets.
 - end result is mixed reflectance.
 - smoother surfaces are more specular or glossy.
 - random distribution of facet normals results in diffuse reflectance.

22

Physics of Diffuse Reflection

- ideal diffuse reflection
 - very rough surface at the microscopic level
 - real-world example: chalk
 - microscopic variations mean incoming ray of light equally likely to be reflected in any direction over the hemisphere
 - what does the reflected intensity depend on?



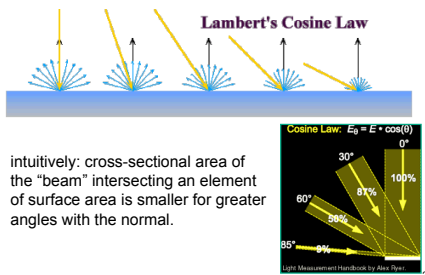
23

Lambert's Cosine Law

- ideal diffuse surface reflection
 - the energy reflected by a small portion of a surface from a light source in a given direction is proportional to the cosine of the angle between that direction and the surface normal
- **reflected** intensity
 - independent of **viewing** direction
 - depends on surface orientation wrt light
- often called **Lambertian surfaces**

24

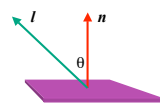
Lambert's Law



25

Computing Diffuse Reflection

- depends on **angle of incidence**: angle between surface normal and incoming light
 - $I_{diffuse} = k_d I_{light} \cos \theta$
- in practice use vector arithmetic
 - $I_{diffuse} = k_d I_{light} (\mathbf{n} \cdot \mathbf{l})$
- **always normalize vectors used in lighting!!!**
 - \mathbf{n} , \mathbf{l} should be unit vectors
- scalar (B/W intensity) or 3-tuple or 4-tuple (color)
 - k_d : diffuse coefficient, surface color
 - I_{light} : incoming light intensity
 - $I_{diffuse}$: outgoing light intensity (for diffuse reflection)



26

Diffuse Lighting Examples

- Lambertian sphere from several lighting angles:

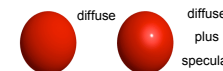


- need only consider angles from 0° to 90°
 - [demo] Brown exploratory on reflection
 - http://www.cs.brown.edu/exploratories/freeSoftware/repository/edu/brown/cs/exploratories/applets/reflection2D/reflection_2d_java_browser.html

27

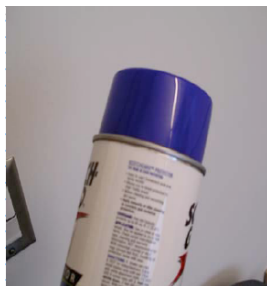
Specular Reflection

- shiny surfaces exhibit specular reflection
 - polished metal
 - glossy car finish
- specular highlight
 - bright spot from light shining on a specular surface
- view dependent
 - highlight position is function of the viewer's position



28

Specular Highlights



Michiel van de Panne

29

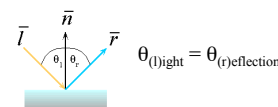
Physics of Specular Reflection

- at the microscopic level a specular reflecting surface is very smooth
- thus rays of light are likely to bounce off the microgeometry in a mirror-like fashion
- the smoother the surface, the closer it becomes to a perfect mirror

30

Optics of Reflection

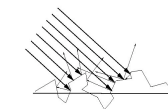
- reflection follows *Snell's Law*:
 - incoming ray and reflected ray lie in a plane with the surface normal
 - angle the reflected ray forms with surface normal equals angle formed by incoming ray and surface normal



31

Non-Ideal Specular Reflectance

- Snell's law applies to perfect mirror-like surfaces, but aside from mirrors (and chrome) few surfaces exhibit perfect specularly
- how can we capture the "softer" reflections of surface that are glossy, not mirror-like?
- one option: model the microgeometry of the surface and explicitly bounce rays off of it
- or...



32

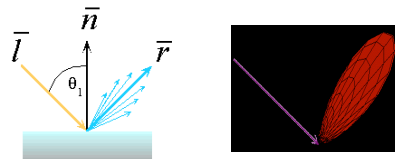
Empirical Approximation

- we expect most reflected light to travel in direction predicted by Snell's Law
- but because of microscopic surface variations, some light may be reflected in a direction slightly off the ideal reflected ray
- as angle from ideal reflected ray increases, we expect less light to be reflected

33

Empirical Approximation

- angular falloff



- how might we model this falloff?

34

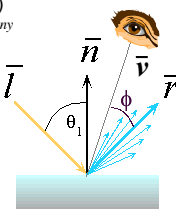
Phong Lighting

- most common lighting model in computer graphics

(Phong Bui-Tuong, 1975)

$$I_{\text{specular}} = k_s I_{\text{light}} (\cos \phi)^{n_{\text{shiny}}}$$

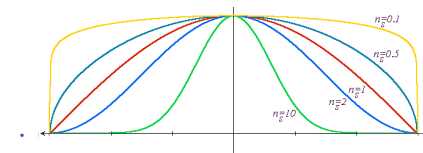
- n_{shiny} : purely empirical constant, varies rate of falloff
- k_s : specular coefficient, highlight color
- no physical basis, works ok in practice



35

Phong Lighting: The n_{shiny} Term

- Phong reflectance term drops off with divergence of viewing angle from ideal reflected ray



Viewing angle – reflected angle

36

Phong Examples

varying I



varying n_{shiny}



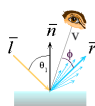
37

Calculating Phong Lighting

- compute cosine term of Phong lighting with vectors

$$I_{\text{specular}} = k_s I_{\text{light}} (\mathbf{v} \cdot \mathbf{r})^{n_{\text{shiny}}}$$

- \mathbf{v} : unit vector towards viewer/eye
- \mathbf{r} : ideal reflectance direction (unit vector)
- k_s : specular component
 - highlight color
- I_{light} : incoming light intensity



- how to efficiently calculate \mathbf{r} ?

38

Calculating R Vector

$\mathbf{P} = \mathbf{N} \cos \theta$ = projection of \mathbf{L} onto \mathbf{N}



39

Calculating R Vector

$\mathbf{P} = \mathbf{N} \cos \theta$ = projection of \mathbf{L} onto \mathbf{N}

$$\mathbf{P} = \mathbf{N} (\mathbf{N} \cdot \mathbf{L})$$



40

Calculating R Vector

$\mathbf{P} = \mathbf{N} \cos \theta |\mathbf{L}| |\mathbf{N}|$ projection of \mathbf{L} onto \mathbf{N}

$\mathbf{P} = \mathbf{N} \cos \theta$ \mathbf{L}, \mathbf{N} are unit length

$$\mathbf{P} = \mathbf{N} (\mathbf{N} \cdot \mathbf{L})$$



41

Calculating R Vector

$\mathbf{P} = \mathbf{N} \cos \theta |\mathbf{L}| |\mathbf{N}|$ projection of \mathbf{L} onto \mathbf{N}

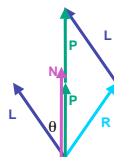
$\mathbf{P} = \mathbf{N} \cos \theta$ \mathbf{L}, \mathbf{N} are unit length

$$\mathbf{P} = \mathbf{N} (\mathbf{N} \cdot \mathbf{L})$$

$$2\mathbf{P} = \mathbf{R} + \mathbf{L}$$

$$2\mathbf{P} - \mathbf{L} = \mathbf{R}$$

$$2(\mathbf{N} (\mathbf{N} \cdot \mathbf{L})) - \mathbf{L} = \mathbf{R}$$



42

Phong Lighting Model

- combine ambient, diffuse, specular components

$$I_{\text{total}} = k_s I_{\text{ambient}} + \sum_{i=1}^{\# \text{lights}} (k_d (\mathbf{n} \cdot \mathbf{l}_i) + k_s (\mathbf{v} \cdot \mathbf{r}_i)^{n_{\text{shiny}}})$$

- commonly called *Phong lighting*

- once per light
- once per color component

- reminder: normalize your vectors when calculating!

43

Phong Lighting: Intensity Plots

Phong	ρ_{ambient}	ρ_{diffuse}	ρ_{specular}	ρ_{total}
$\phi = 60^\circ$				
$\phi = 25^\circ$				
$\phi = 0^\circ$				

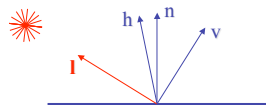
44

Blinn-Phong Model

- variation with better physical interpretation

• Jim Blinn, 1977
 $I_{\text{out}}(\mathbf{x}) = k_s (\mathbf{h} \cdot \mathbf{n})^{n_{\text{shiny}}} \cdot I_{\text{in}}(\mathbf{x})$; with $\mathbf{h} = (\mathbf{l} + \mathbf{v})/2$

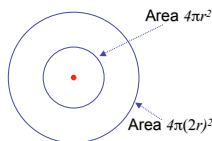
- \mathbf{h} : halfway vector
 - \mathbf{h} must also be explicitly normalized: $\mathbf{h} / |\mathbf{h}|$
 - highlight occurs when \mathbf{h} near \mathbf{n}



45

Light Source Falloff

- quadratic falloff
- brightness of objects depends on power per unit area that hits the object
- the power per unit area for a point or spot light decreases quadratically with distance



46

Light Source Falloff

- non-quadratic falloff
- many systems allow for other falloffs
- allows for faking effect of area light sources
- OpenGL / graphics hardware
 - I_0 : intensity of light source
 - \mathbf{x} : object point
 - r : distance of light from \mathbf{x}

$$I_{\text{in}}(\mathbf{x}) = \frac{1}{ar^2 + br + c} \cdot I_0$$

47

Lighting Review

- lighting models
 - ambient
 - normals don't matter
 - Lambert/diffuse
 - angle between surface normal and light
 - Phong/specular
 - surface normal, light, and viewpoint

48

Lighting in OpenGL

- light source: amount of RGB light emitted
 - value represents percentage of full intensity e.g., (1.0,0.5,0.5)
 - every light source emits ambient, diffuse, and specular light
- materials: amount of RGB light reflected
 - value represents percentage reflected e.g., (0.0,1.0,0.5)
- interaction: multiply components
 - red light (1,0,0) x green surface (0,1,0) = black (0,0,0)

49

Lighting in OpenGL

```
glLightfv(GL_LIGHT0, GL_AMBIENT, amb_light_rgba );
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif_light_rgba );
glLightfv(GL_LIGHT0, GL_SPECULAR, spec_light_rgba );
glLightfv(GL_LIGHT0, GL_POSITION, position);
glEnable(GL_LIGHT0);

glMaterialfv( GL_FRONT, GL_AMBIENT, ambient_rgba );
glMaterialfv( GL_FRONT, GL_DIFFUSE, diffuse_rgba );
glMaterialfv( GL_FRONT, GL_SPECULAR, specular_rgba );
glMaterialfv( GL_FRONT, GL_SHININESS, n );
```

- warning: glMaterial is expensive and tricky
 - use cheap and simple glColor when possible
 - see OpenGL Pitfall #14 from Kilgard's list

<http://www.opengl.org/resources/features/KilgardTechniques/oglpitfall/>

50