OpenGL
Lighting and Texturing

Jacobo Bibliowicz
University of Toronto
February 11, 2008

Why use lighting and textures?

- Add realism.
- Does not increase geometric complexity!
- Models the real world.

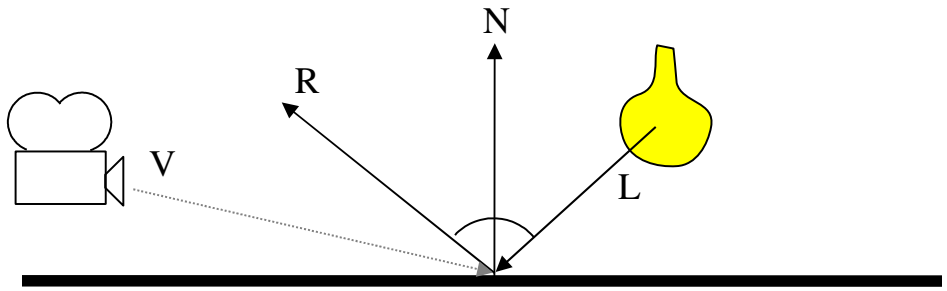
Lighting in OpenGL

- OpenGL provides:
  - Phong lighting (materials).
  - Gouraud shading (color interpolation).

- Phong materials are composed of a diffuse (Lambertian) term and a specular term.

$$C = k_d\{N \cdot L\} + k_s\{R \cdot V\}^n$$

What goes where?



$$C = k_d\{N \cdot L\} + k_s\{R \cdot V\}^n$$

- Material properties: k_d, k_s, n
- Surface properties: N
  - With the light: L, R
  - With the camera: V

Specifying Materials in OpenGL

- One function for all material properties!

```
glMaterialf{v}(GLenum face, TYPE param);
```

- Example:

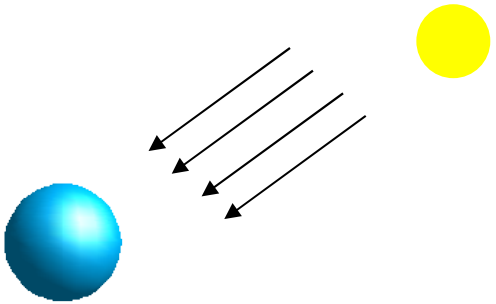
```
GLfloat diff[] = {1.0, 0.0, 0.0, 1.0};
Glfloat spec[] = {0.0, 1.0, 1.0, 1.0};

glMaterialfv(GL_FRONT, GL_DIFFUSE, diff);
glMaterialfv(GL_FRONT, GL_SPECULAR, spec);
glMaterialf(GL_FRONT, GL_SHININESS, 15.0);
```

OpenGL Lights

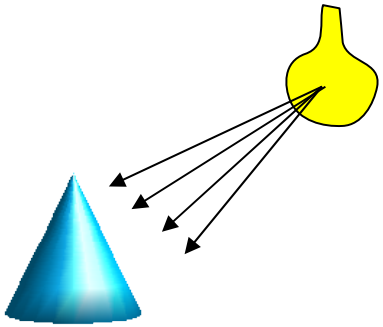
- Lights are placed in the scene only once.
- OpenGL supports a number of lighting effects:
  - Directional v. Positional lights.
  - Spot lights.
  - Attenuation.
  - Ambient lighting.

Directional v. Positional Lights

- A directional light is assumed to be infinitely far away (ex. sun)

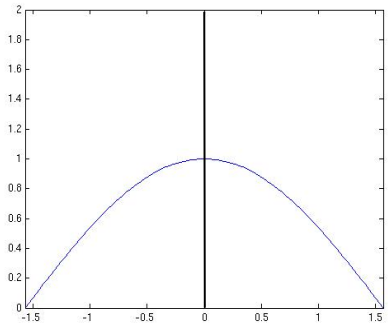


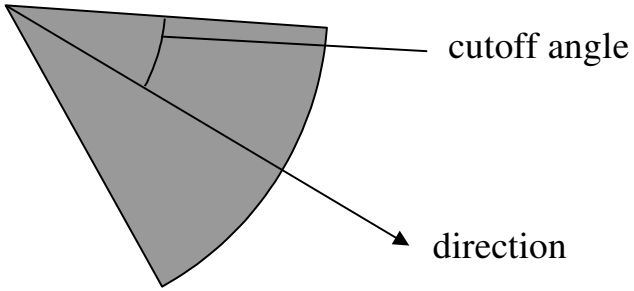
- A positional light is not so far! (ex. lightbulb)



Spot lights

- Cone shaped lighting.
- Shape determined by
  - cutoff angle
  - direction (assume point light)
  - cosine exponent (shape of fall-off curve).





Attenuation, Ambient Lighting

- Attenuation
  - Reduction in light intensity due to distance from the light.
- Ambient lighting
  - Refers to background lighting caused by many lights.
  - Light scattered by environment
    - Unable to determine its direction.

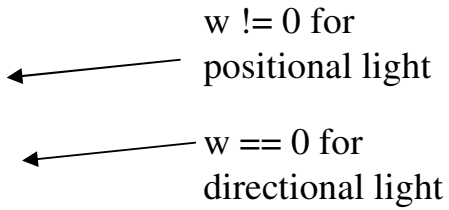
Specifying OpenGL lights

- Once again, one function! :)

  glLight{if}{v}(<light>, <property>, <val>)

- Example:

```
GLfloat pos[] = {5.0, -3.0, 8.0, 1.0};
GLfloat pos2[] = {1.0, -1.0, 1.0, 0.0};
Glfloat diff[] = {1.0, 1.0, 1.0, 1.0};
Glfloat dir[] = {0.0, -3.0, 3.0};

glLightfv(GL_LIGHT0, GL_POSITION, pos2);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);
```



Specifying OpenGL lights

```
glLightfv(GL_LIGHT1, GL_POSITION, pos);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diff);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, dir);

glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 30.0);

...
```

Ambient lighting

- Both lights and materials support an ambient color:

```
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
```

- OpenGL also provides a GLOBAL ambient term.

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);
```

- Objects can glow: emissive materials

```
glMaterialfv(GL_FRONT, GL_EMISSION, em);
```

Mathematics of Lighting

- The final color of a vertex is computed as follows:

C = <GLOBAL ambient>*<mat ambient>
+ <mat emission>;
for each light L
C += <L attenuation>*<spotlight effects>
[<L ambient><mat ambient>
+ <L diffuse>*<mat diffuse>*<diffuse term>
+ <L spec>*<mat spec>*<spec term>];

Putting it all together...

- Steps to perform lighting in OpenGL
  1. Create and position your lights (glLight()).
  2. Enable lighting and individual lights

     ```
     glEnable(GL_LIGHT0);  glEnable(GL_LIGHT1);
     glEnable(GL_LIGHTING);
     ```

  3. For each rendered object
     1. Define material properties (glMaterial()).
     2. Provide normals for each rendered vertex!