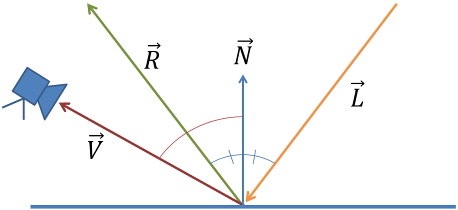
Lights & Materials in OpenGL

Ensimag 3D Graphics, 2014

Local Illumination



$$I = I_e + K_a + \sum_{i \in lights} I_i\left(K_d \vec{L_i}.\vec{N} + K_s(R_i.V)^n\right)$$

Where:

- I_e the emissive light (not in Phong model, OpenGL only)
- K_a is the ambiant term
- I_i is the intensity of the light i
- $K_d \vec{L_i}.\vec{N}$ is the diffuse term
- $K_s(R_i.V)^n$ is the specular term

Local Illumination

Pros:

- Easy to compute

Cons:

- No casted shadows
- Objects can't be light sources

Light Sources

Enable lighting:

```
glEnable(GL_LIGHTING)
```

Turn on one of the predefined lights (seven predefined):

```
glEnable(GL_LIGHT0)
```

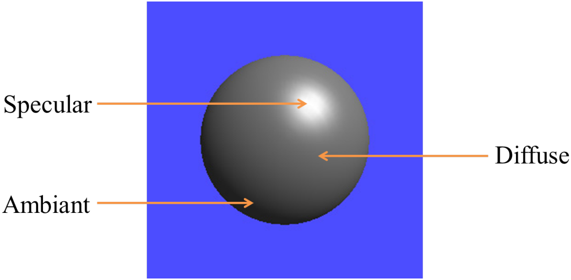
Then change its parameters:

```
glLightf(GLenum light, GLenum pname, GLfloat[*] p)
```

- *light* is the source's name: `GL_LIGHT0, GL_LIGHT1, ...`
- *pname* is the parameter's name : `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR`, `GL_POSITION`, ...
- *p* is the new value of the parameter: `(r, g, b, alpha)`, ...

Light Parameters 1/4

Example:



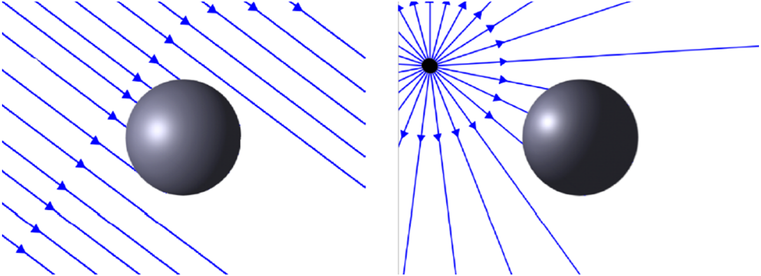

```
glClearColor(light_blue);
glLightfv(GL_LIGHT0, GL_AMBIENT,  dark_grey);
glLightfv(GL_LIGHT0, GL_DIFFUSE,  white);
glLightfv(GL_LIGHT0, GL_SPECULAR, white);
```

Light Parameters 2/4

pname = `GL_POSITION`

p = $x,\ y,\ z,\ w$

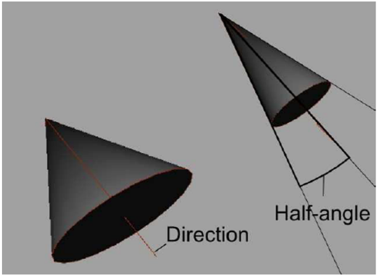
- if w=0: directional light, (x, y, z) = direction
- else: point light, (x, y, z) = position

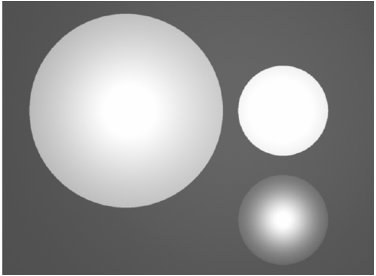


Light Parameters 3/4

pname = `GL_SPOT_CUTOFF` or `GL_SPOT_DIRECTION` or `GL_SPOT_EXPONENT` :

- `GL_SPOT_CUTOFF`: **p** = cone half-angle (in degree)
- `GL_SPOT_DIRECTION`: **(x, y, z, w)** = direction
- `GL_SPOT_EXPONENT`: **p** = attenuation of the light intensity





Light Parameters 4/4

pname = GL_*_ATTENUATION where * is CONSTANT, LINEAR, QUADRATIC
The attenuation factor of the light at distance t is thus:

$$a(t) = \frac{1}{k_c + k_l t + k_q t^2}$$

Materials - 1/3

Defines how an object *reflects* the different components of the lights. Exemple: **(1.0, 0.5, 0.0)** reflects 100% of the red incoming light, 50% of the green and none of the blue light.

Properties

- ambient color: `GL_AMBIENT`
- diffuse color: `GL_DIFFUSE`
- specular color: `GL_SPECULAR`
- shininess (scalar): `GL_SHININESS`
- emissive color of material: `GL_EMISSION`
- `GL_AMBIENT_AND_DIFFUSE`, ...

Materials - 2/3

Two ways to define the material properties of an object:

1. Define each property individually:

```
glMaterialfv(GL_FRONT_AND_BACK,
             GL_AMBIENT, ambientColor);

glMaterialfv(GL_FRONT,
             GL_SPECULAR, specularColor);

glMaterialfv(GL_FRONT,
             GL_SHININESS, shininess);
```

Materials - 3/3

2. Or:
   - Specify which property(ies) to set :

```
glColorMaterial(GL_FRONT_AND_BACK,
                GL_AMBIENT_AND_DIFFUSE);
```

   - Then use :

```
glColor4f(0.8f, 0.3f, 0.2f);
```

   - Require `GL_COLOR_MATERIAL` (disable it for `glMaterial*` !!)
   - Beware: cannot be set between `glBegin()` and `glEnd()`
   - May be easier when a single parameter is change from one object to the other

Do not mix the two models!

Blending

How OpenGL deals with the α value

- Do a combination of the currently computed color (source) with the color in the framebuffer (destination)
  1. multiply the source color by the source factor : $(R_s S_r, G_s S_g, B_s S_b, A_s S_a)$
  2. multiply the destination color by the destination factor : $(R_d D_r, G_d D_g, B_d D_b, A_d D_a)$
  3. the final blended color is the sum the two components
- Must enable blending `glEnable(GL_BLEND)` !

Blending

The blending equation is specified by :

```
glBlendFunc(srcFactor, destFactor)
```

where factor can be :

`GL_ZERO`	$(0,0,0,0)$
`GL_ONE`	$(1,1,1,1)$
`GL_SRC_COLOR`	(R_s, G_s, B_s, A_s)
`GL_ONE_MINUS_SRC_COLOR`	$(1,1,1,1) - (R_s, G_s, B_s, A_s)$
`GL_SRC_ALPHA`	(A_s, A_s, A_s, A_s)
`GL_ONE_MINUS_SRC_ALPHA`	$(1,1,1,1) - (A_s, A_s, A_s, A_s)$
`GL_DST_COLOR`	(R_d, G_d, B_d, A_d)
`GL_CONSTANT_COLOR`	(R_c, G_c, B_c, A_c)
`GL_CONSTANT_ALPHA`	(A_c, A_c, A_c, A_c)
...	

More complex effects by changing the blending operator:
`glBlendEquation(mode)` with `GL_FUNC_ADD` (default), `GL_FUNC_SUBSTRACT`, `GL_FUNC_MIN`, `GL_LOGIC_OP`, ...

Blending

Example (most common parameters) :

```
glEnable(GL_BLEND);

// set the first color
glColor4f(0.0f, 1.0f, 0.0f, 1.0);
// override the dst with src value
glBlendFunc(GL_ONE, GL_ZERO);
drawObject1();

// set the second color
glColor4f(1.0f, 1.0f, 0.0f, 0.25f);
// alpha of the src color, 1-alpha of the dest color
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
drawObject2();

//     => 0.75 of the dst color, .25 of src one

glDisable(GL_BLEND);
```

Fog

Principle: The further the object is, the closer the color perceived is to the fog's color
In practice: blend the color of the object with the color of the fog according to a factor f

$$RGBA = f \times RGBA_{object} + (1 - f) \times RGBA_{fog}$$

- f depends on the depth of the object:
  $f = e^{-density \times z}$ $\Rightarrow$ `GL_EXP`
  $f = e^{-(density \times z)^2}$ $\Rightarrow$ `GL_EXP2`
  $f = \dfrac{end - z}{end - start}$ $\Rightarrow$ `GL_LINEAR`

Fog: Example

```
glEnable(GL FOG);
GLfloat fogColor[4] = {0.5, 0.5, 0.5, 1.0 };
glFogi(GL FOG MODE, GL EXP);
glFogfv(GL FOG COLOR,fogColor);
glFogf(GL FOG DENSITY,0.35);
```