Lights and materials

Estelle Duveau - estelle.duveau@inria.fr

MoSIG1, Introduction to Computer Graphics, 11/03/2009

Planning

- **Lecture :** introduction to OpenGL
  Lab : first steps in OpenGL and modeling - 25/02/2009
- **Lecture/Lab :** transformations and hierarchical modeling - 04/03/2009
- **Lecture :** lights and materials in OpenGL - 11/03/2009
- **Lab :** lights and materials in OpenGL - 18/03/2009
- **Lecture :** textures in OpenGL
  Lab : textures in OpenGL - 25/03/2009
- **Lab :** procedural animation - 01/04/2009
- **Lab :** physical animation : particle systems - 08/04/2009
- **Lab :** physical animation : collisions - 22/04/2009

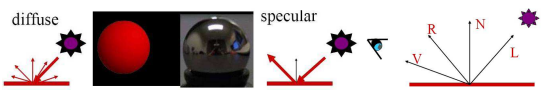
Plan

Plan

Local illumation

Color displayed depends on :

- Position of the surface element
- Its orientation with respect to lights and camera
- The material of the surface



$$I = Ka + \sum Is\,(Kd\,L\,.\,N + Ks\,(R\,.\,V)^n)$$

ambiant diffuse specular

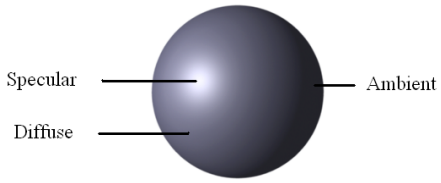
Plan

Light sources

- Enable lighting :
  `glEnable(GL_LIGHTING)`
- Turn on one of the `GL_MAX_LIGHTS` predefined lights :
  `glEnable(GL_LIGHT0)`
- `glLightf(GLenum light, GLenum pname, GLfloat p)`
  - light : source name (`GL_LIGHT0` ... `GL_LIGHT7`)
  - pname : parameter tuned
  - p : value of the parameter

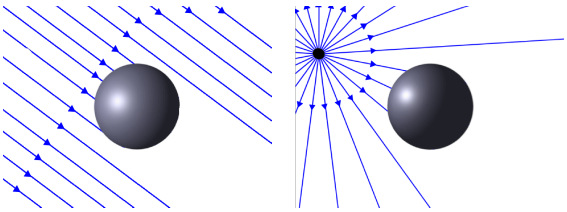
Parameters 1/4

- **pname** = `GL_AMBIENT` **or** `GL_DIFFUSE` **or** `GL_SPECULAR` :
  p = (r, g, b, α)



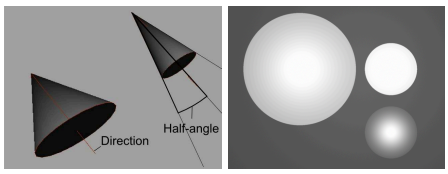
Parameters 2/4

- **pname** = `GL_POSITION` :
  p = (x, y, z, w)
  - if $w = 0$: directional light, (x, y, z) = direction
  - otherwise : point light, (x, y, z) = coordinates in the current reference frame



Parameters 3/4

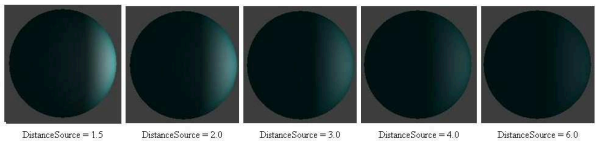
- **pname** = `GL_SPOT_CUTOFF` **or** `GL_SPOT_DIRECTION` **or** `GL_SPOT_EXPONENT` :
  - pname = `GL_SPOT_DIRECTION`,
    p = direction
  - pname = `GL_SPOT_CUTOFF`,
    p = cone half-angle
  - pname = `GL_SPOT_EXPONENT`,
    p = attenuation of the light intensity away from the direction



Parameters 4/4

- **pname** = `GL_*_ATTENUATION` :
  * = `CONSTANT`, `LINEAR`, `QUADRATIC`
  p = constant k_c, k_l, k_q of the attenuation factor :

$$\frac{1}{k_c + k_l d + k_q d^2}.$$



Plan

Materials - 1/2

- **Properties :**
  - ambient color
  - diffuse color
  - specular color
  - shininess *(scalar)*
  - interpolation : `GL_FLAT`, `GL_SMOOTH`
- **Color description :**
  - Red
  - Green
  - Blue
  - α = blending factor

Materials - 2/2

Two ways to define the material properties of an object :

- Property defined by `glColor(...)` :

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

  - Required : `glEnable(GL_COLOR_MATERIAL)`
  - **Beware !** Can not be set between `glBegin()` and `glEnd()`
- Define the other properties with :

```
glMaterialfv(GL_FRONT_AND_BACK,GL_SPECULAR,specular)
glMaterialfv(GL_FRONT_AND_BACK,GL_SHININESS,shininess)
```


Plan

Plan

Blending

- Blending = fusion, combination $\rightarrow$ of colors
- **Goal :** create effects (such as transparency) combining colors from a source and from a destination
- Many blending functions possible $\Rightarrow$ many effects possible
- Use of the alpha canal (RBGA)

Blending : principle

Combination of the color of the current pixel (source) with the color already in the framebuffer (destination)

Example :
Seeing an object throught a green glass that lets through 80% of incoming light (opacity = α_{new} = 20%) :

Color to display
= 20% glass color and 80% object color
= 20% new color + 80% previous color
= $\alpha_{new} RGBA_{new} + (1.0 - \alpha_{new}) RGBA_{old}$

Blending functions - 1/3

- color = $(a_1 R_{new} + a_2 R_{old}, b_1 G_{new} + b_2 G_{old}, c_1 B_{new} + c_2 B_{old}, d_1 \alpha_{new} + d_2 \alpha_{old})$
- 2 commands :
  - `glBlendFunc(srcFactor, destFactor)`
  - `glBlendFuncSeparate(srcRGB, destRGB, srcAlpha, destAlpha)`
- Don't forget `glEnable(GL_BLEND)` !

Blending functions - 2/3

Argument	Blending factor
GL_ZERO	$(0.0, 0.0, 0.0, 0.0)$
GL_ONE	$(1.0, 1.0, 1.0, 1.0)$
GL_SRC_ALPHA	$(\alpha_{source}, \alpha_{source}, \alpha_{source}, \alpha_{source})$
GL_ONE_MINUS_SRC_ALPHA	$(1, 1, 1, 1) - (\alpha_{source}, \alpha_{source}, \alpha_{source}, \alpha_{source})$
GL_CONSTANT_COLOR	$(R_{constant}, G_{constant}, B_{constant}, \alpha_{constant})$
GL_CONSTANT_ALPHA	$(\alpha_{constant}, \alpha_{constant}, \alpha_{constant}, \alpha_{constant})$
...	...

For the last 2 cases, constant specified with
`glBlendColor(cstRed,cstGreen,cstBlue,cstAlpha)`

Blending functions - 3/3

- More complex functions than addition :

GL_FUNC_ADD	$k_1 RGBA_{new} + k_2 RGBA_{old}$
GL_FUNC_SUBTRACT	$k_1 RGBA_{new} - k_2 RGBA_{old}$
GL_FUNC_REVERSE_SUBTRACT	$k_2 RGBA_{old} - k_1 RGBA_{new}$
GL_MIN	$\min(k_1 RGBA_{new}, k_2 RGBA_{old})$
GL_MAX	$\max(k_1 RGBA_{new}, k_2 RGBA_{old})$
GL_LOGIC_OP	$RGBA_{new}$ op $RGBA_{old}$

- Define with

```
glBlendEquation(mode)
```

or

```
glBlendEquationSeparate(modeRGB, modeAlpha)
```

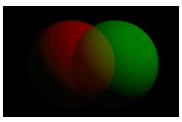

Plan

Transparency

Transparency $= \alpha_{new} RGBA_{new} + (1.0 - \alpha_{new}) RGBA_{old}$

Example :
Blend two objects with the same proportion :

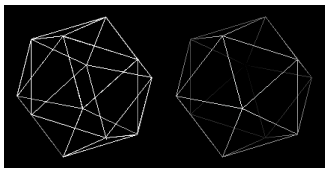
```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,(0,1,0,0.5));
drawSphere()
glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,(1,0,0,0.5));
drawSphere()
```



Plan

Fog : principle

- **The further away an object is from the viewpoint, the closer the color perceived is to the color of the fog**
- Can be used for fog but also smoke, pollution, steam, depth cueing ...
- Depth cueing : reduce the intensity of an object depending on its distance to the viewpoint



Fog : parameters tuning

- Blend the color of the object with the color of the fog according to a factor f :

$$RGBA = fRGBA_{objet} + (1 - f)RGBA_{fog}$$

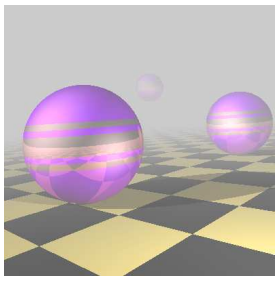
- f depends on the depth z of the object :

$$f = e^{-density.z} \quad \texttt{GL\_EXP}$$
$$f = e^{-(density.z)^2} \quad \texttt{GL\_EXP2}$$
$$f = \frac{end - z}{end - start} \quad \texttt{GL\_LINEAR}$$

- *density*, *start*, *end* → `GL_FOG_DENSITY`, `GL_FOG_START`, `GL_FOG_END`
- Direction of z can be changed with `glFogCoord(...)` ⇒ property of a vertex

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);
```



Plan

Conclusion

- Done :
  - Lights
  - Materials
  - Blending effects
- Highlights :
  - Diffuse/Specular/Ambient/Shininess
  - Many effects quite easily
- To do :
  - Lab session : experiment notions
  - More complex modeling of materials : textures