

Grafica Computazionale

Lab: Lighting & Shading

Fabio Ganovelli

fabio.ganovelli@gmail.com

a.a. 2006-2007



Lighting in OpenGL

- Diciamo a OpenGL di usare il lighting:

.....

```
glEnable(GL_LIGHTING)  
// da qui in poi glColorxx non ha  
// "quasi" effetto. Per ora togliamo il  
// quasi
```

- Ora dobbiamo specificare
 - Le luci
 - Le proprietà del materiale



Le Luci

- OpenGL tratta un numero di luci variabili da 8 a `GL_MAX_LIGHT`
- Le luci sono numerate da 0 a `GL_MAX_LIGHT`:
 - `glLight0, glLight1, glLight2, ..., oppure`
 - `glLight0, glLight0+1, glLight0+2, ...`
- Accendiamo una luce
`glEnable(GL_LIGHT0);`
- Spegnamo la luce
`glDisable(GL_LIGHT0);`



Le luci: Colore

```
void glLightfv(  
GLenum light,           // quale luce  
GLenum pname,          // quale attributo  
const GLfloat *params  // valore (4 float)  
);
```

Esempio:

```
GLfloat a[4]={1.0,0.0,1.0,1.0};  
GLfloat s[4]={1.0,1.0,1.0,1.0};  
void glLightfv(GL_LIGHT0, GL_AMBIENT, a );  
void glLightfv(GL_LIGHT0, GL_SPECULAR, a );
```

OpenGL: posizione

- Di ogni luce, settiamo la posizione:

```
glLightfv(GL_LIGHT0, GL_POSITION, v);
```

- Se luce posizionale,
 $v = \{x, y, z, 1\}$
- Se luce direzionale, ("distante all'infinito")
 $v = \{x, y, z, 0\}$
- Coordinate affini!



Le luci: spotlight e attenuation

- Di ogni luce, possiamo anche settare:

– se voglio effetto spotlight:

default:

```
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, v);      (0, 0, -1)
glLightf (GL_LIGHT0, GL_SPOT_CUTOFF, v);        180
glLightf (GL_LIGHT0, GL_SPOT_EXPONENT, v);      0.0
```

– se voglio attenuazione con la distanza:

default:

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a); 1
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);   0
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c); 0
```

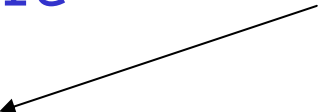


Il materiale

■ Similarmente:

```
void glMaterialfv(  
    GLenum face,    // GL_FRONT, GL_BACK, GL_FRONT_AND_BACK  
    GLenum pname,   // quale attributo  
    const GLfloat *params // valore  
);  
glMaterialf (face, GL_SHININESS, intval);
```

L'esponente del coseno



■ Esempio

```
GLfloat am[4]={1.0,0.0,1.0,1.0};  
void glMaterialfv(GL_FRONT, GL_AMBIENT, am );
```

Le normali rimangono normali nella Transform?

- Solo se la modellazione-vista è rigida

$$\text{modellazione-vista} = V * M$$

rotazioni,
traslazioni
(quindi sempre rigida)

rotazioni,
traslazioni
e forse scalature
(quindi non sempre rigida)



Le normali rimangono normali nella Transform?

- Morale: se uso **scalature** nella ModelView devo **rinormalizzare** le normali prima del Lighting
- chiedo ad OpenGL di farlo:

```
glEnable(GL_NORMALIZE);
```

o di non farlo: (default)

```
glDisable(GL_NORMALIZE);
```



Normali come attributi

- Proprio come il colore:

```
glBegin(GL_TRIANGLES);  
  glNormal3fv( n );  
  glVertex3fv( v0 );  
  glVertex3fv( v1 );  
  glVertex3fv( v2 );  
glBegin(GL_END);
```

flat shading !

```
glBegin(GL_TRIANGLES);  
  glNormal3fv( n0 );  
  glVertex3fv( v0 );  
  glNormal3fv( n1 );  
  glVertex3fv( v1 );  
  glNormal3fv( n2 );  
  glVertex3fv( v2 );  
glBegin(GL_END);
```

Gouraud shading
(o forse Phong)



Normali come attributi

- Scorciatoia:
 - se setto:

```
glShadeModel (GL_FLAT) ;
```

gli attributi non vengono interpolati
ma rimangono costanti nella faccia
(utile per le triangle strip e i triangle fan),
finchè non rimetto

```
glShadeModel (GL_SMOOTH) ;
```



OpenGL: materiali!

- Settiamo tutti i parametri dei materiali:

- i colori:

```
glMaterialfv(face, GL_AMBIENT, colorvec);  
glMaterialfv(face, GL_EMISSION, colorvec);  
glMaterialfv(face, GL_DIFFUSE, colorvec);  
glMaterialfv(face, GL_SPECULAR, colorvec);
```

- l'esponente speculare:

```
glMaterialf (face, GL_SHININESS, intval);
```

"GL_FRONT"

scorciatoia: esiste anche
GL_AMBIENT_AND_DIFFUSE
che setta entrambi
i colori allo stesso valore

Color – Material: glColorXX strikes back!

attivazione:

```
glEnable(GL_COLOR_MATERIAL);
```

USO:

```
glColorMaterial(face, mode);
```

es: se si mette

```
glColorMaterial(GL_FRONT, GL_DIFFUSE);
```

allora come colore diffuso del materiale si

userà (l'altrimenti ignorato) **colore corrente**

settato con vecchio **glColorXX**



Lab.03.lighting.exe

