Libreria Grafica OpenGL parte II

www.opengl.org

Illuminazione in OpenGL

Come si è detto, ogni vertice ha un colore (glColor( )) e ogni punto interno ad un poligono ha un colore dato dall'interpolazione del colore dei vertici della primitiva; ma se si abilita l'illuminazione …

Abilitiamo l'illuminazione (per default è disabilitata):

```
glEnable (GL_LIGHTING);
```

una volta abilitata, il colore di ogni vertice dipende da:

- **Proprietà delle sorgenti luminose (o luci)**
- **Proprietà del materiale di cui è composto l'oggetto**
- **Dal modello di illuminazione (basato sul modello di Phong che necessita delle normali)**

Definizione Luci

```
glEnable(GL_LIGHT0);  //abilita luce 0
glEnable(GL_LIGHT1);  //abilita luce 1
....
```

Quante?

Il numero esatto lo troviamo nella costante GL_MAX_LIGHT

glLight[fv](<nome luce>,<nome param.>,<valore param.>);

nome luce: GL_LIGHT0, GL_LIGHT1, ... , GL_LIGHT7

nota: **GL_LIGHT*k*** vale **GL_LIGHT0+*k***. Utile per i cicli **for**

nome param. : GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION, GL_SPOT, ecc.

valore param.: puntatore al valore RGBA, posizione,...

Nota: la posizione delle luci è soggetta a trasformazioni come tutte le altre primitive geometriche (matrice ModelView)

Tipi di Luci

glLightfv(GL_LIGHT0, GL_POSITION, v);

- **Direzionale**: localizzata all'infinito in un punto v=[x,y,z,w] con w=0.
- **Puntiforme**:v=[x,y,z,w] con w=1; irradia in tutte le direzioni

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,d);
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,ang);
glLightf(GL_LIGHT0,GL_SPOT_EXPONENT,e);
```

- **Spotlight**: concentra la luce in un cono (posizione, direzione d=[0,0,-1], angolo ang=180, esponente e=0.0)

“nome param.” in glLight

Ogni risorsa luminosa da’ un contributo in termini di componente ambiente, diffusa e speculare al modello di illuminazione. Ciascuna componente ha un’ intensità e un colore (RGB).

GL_AMBIENT

- Specifica l’intensità RGB ambiente

GL_DIFFUSE

- Specifica l’intensità RGB diffusa

GL_SPECULAR

- Specifica l’intensità RGB speculare

Materiale

glMaterial[if]v(<nome faccia>,<nome par.>,<val. par.>);

nome faccia: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

nome par.: GL_AMBIENT, GL_DIFFUSE,
GL_SPECULAR, GL_SHININESS,
GL_AMBIENT_AND_DIFFUSE

val. par.: puntatore al valore RGB

Color-Material in OpenGL

Attivazione:

glEnable(GL_COLOR_MATERIAL);

Uso:

glColorMaterial(<face> , <mode>);

face: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

mode: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_AMBIENT_AND_DIFFUSE

Esempio:

glColorMaterial(GL_FRONT, GL_DIFFUSE);

in questo caso come colore del materiale si userà il colore corrente settato con **glColor3f**

Accendere/spegnere le luci in scena..

- **Abilitare l'illuminazione** (agisce globalmente su tutte le luci)

  glEnable(GL_LIGHTING);
- **Accendere la luce n-esima GL_LIGHTn**

  glEnable(GL_LIGHTn);
- **Spegnere la luce n-esima GL_LIGHTn**

  glDisable(GL_LIGHTn);
- **Definire proprietà di ciascuna luce in scena**

  glLightfv( );
- **Definire proprietà del materiale di ciascuna primitiva**

  glMaterialfv( );

Modello di Illuminazione

glLightModel[fv](<nome par.> , <valore/i par.>);

nome par:

GL_LIGHT_MODEL_AMBIENT **Inizializza la luce ambiente globale (intensità RGBA della scena es: 0.2 0.2 0.2 1.0**

GL_LIGHT_MODEL_TWO_SIDE **considera front e back (=1) o solo front (=0) del materiale**

GL_LIGHT_MODEL_LOCAL_VIEWER **disabilita l'accelerazione del calcolo dell'illuminazione (0 = non considera VRP)**

GL_LIGHT_MODEL_COLOR_CONTROL **ottimizza la luce speculare**

Colore Generato

Il colore prodotto, illuminando un vertice, è calcolato come segue:

Vertex Color = Ke +
Ka * Ia_globale +
Ka * Ia +
Kd * Id * cos_theta +
Ks * Is * (cos_phi)^Se

Ricordiamo il Modello di Phong

$$I = k_a I_a + I_l \left(k_d (L \cdot N) + k_s (R \cdot V)^n \right)$$

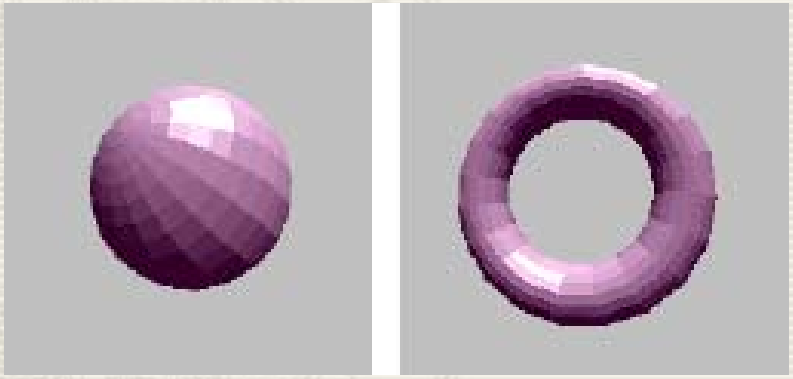
Tutorial Lightposition
(gltutorials)

Tutorial Lightmaterial
(gltutorials)

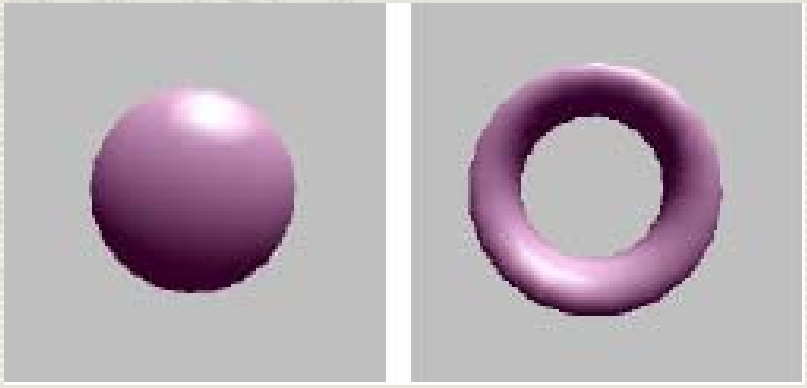
Shading in OpenGL

OpenGL permette due tecniche di shading: flat e smooth (Gouraud).

glShadeModel(GL_FLAT);



glShadeModel(GL_SMOOTH);

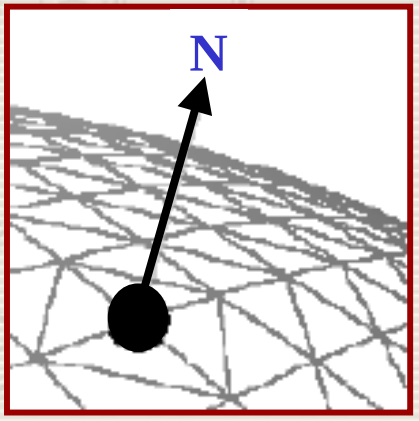


Normali alla Superficie

Le normali devono essere calcolate e assegnate al vertice mediante:

```
glNormal3f( x, y, z );
```

Tale vettore viene usato per tutti i vertici che seguono finché non viene specificata una nuova normale.

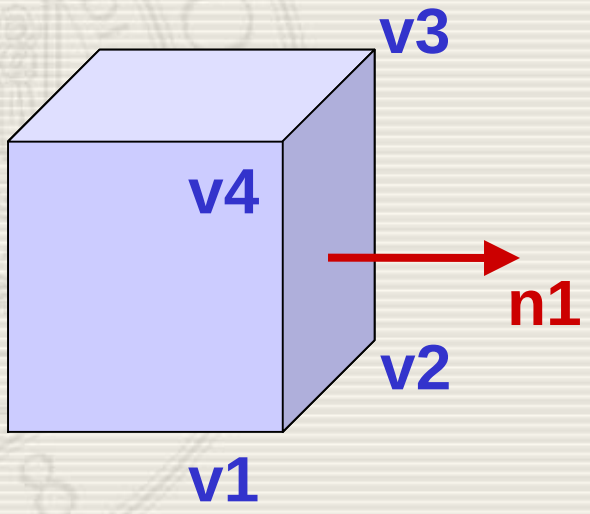


Se si usano scalature nella ModelView bisogna abilitare la rinormalizzazione delle normali:

```
glEnable(GL_NORMALIZE );
```

Esempio: resa di un Cubo (Flat)

```
glShadeModel(GL_FLAT);
glBegin(GL_POLYGON) ;
   glNormal3fv(n1);
   glVertex3fv(v1);
   glVertex3fv(v2);
   glVertex3fv(v3);
   glVertex3fv(v4);
glEnd( );
```



Esempio: resa di un Cubo (Gouraud)

```
glShadeModel(GL_SMOOTH);
glBegin(GL_POLYGON);
  glNormal3fv(n1);
  glVertex3fv(v1);
  glNormal3fv(n2);
  glVertex3fv(v2);
  glNormal3fv(n3);
  glVertex3fv(v3);
  glNormal3fv(n4);
  glVertex3fv(v4);
glEnd( );
```