

## Costruzione di Interfacce Lezione 9 Esercitazione Shading

[cignoni@isti.cnr.it](mailto:cignoni@isti.cnr.it)  
<http://vcg.isti.cnr.it/~cignoni>

## Sempre piu' difficile

- ❖ Aggiungiamo nell'ordine
  - ❖ L'anello che e' formato da una doppia striscia
  - ❖ Una pallina che rotola sull'anello
  - ❖ L'anello che ruota su se stesso

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

2

## Doppio Anello

- ❖ Semplice basta sostituire il blocco con un doppio blocco.

```
void DrawDoubleBlock(float xsz, float ysz, float zsz, float distance)
{
    glPushMatrix();
    glTranslatef(0, distance, 0);
    DrawBlock(xsz, ysz, zsz);
    glTranslatef(0, -2*distance, 0);
    DrawBlock(xsz, ysz, zsz);
    glPopMatrix();
}
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

3

## Pallina che ruota sull'anello

- ❖ Approssimiamo la pallina con un cubetto :)
- ❖ Facciamo una funz che disegna la pallina sopra un anello di moebius in funzione del tempo.
- ❖ Il codice e' molto simile a disegnare l'anello stesso.

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

4

## Rotolamento pallina 1

```
void DrawRing(int step, float r, int twist)
{
    float angle=360.0f/step;
    float angletwist=(180.0f*twist)/step;
    for(int i=0;i<step;++i)
    {
        glPushMatrix();
        glRotatef(i*angle,0,1,0);
        glTranslatef(r,0,0);
        glRotatef(i*angletwist,0,0,1);
        DrawDoubleBlock(1.5, .2, .5, .4);
        glPopMatrix();
    }
}
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

5

## Rotolamento pallina 2

```
void DrawMoebiusRollingBall(float r, int twist, float angle, float radius)
{
    float TwistPerDeg=(twist*180.0)/360.0f;
    glPushMatrix();
    glRotatef(angle,0,1,0);
    glTranslatef(r,0,0);
    glRotatef(angle*TwistPerDeg,0,0,1);
    DrawTranslatedRollingBall(radius, angle);
    glPopMatrix();
}
void DrawTranslatedRollingBall(float radius, float angle)
{
    const MysteriousConstant = -3;
    glPushMatrix();
    glTranslatef(0, radius*1.5, 0);
    glRotatef(angle*MysteriousConstant,1,0,0);
    DrawBlock(radius, radius, radius);
    glPopMatrix();
}
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

6

## Shading

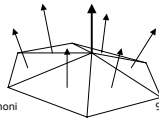
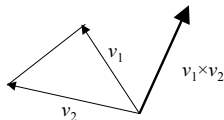
- ❖ Dal wireframe ai modelli ombreggiati:
  - ❖ Non e' solo fatica di OpenGL
  - ❖ Si deve passare informazione in piu'
  - ❖ Si deve costruire la superficie in maniera accurata

## Ricetta x disegnare shaded

- ❖ Definire i vettori normali per ogni vertice dell'oggetto
- ❖ Creare e abilitare una o più luci
- ❖ Scegliere un metodo di illuminazione e di shading
- ❖ Definire le proprietà dei materiali

## Definire le normali

- ❖ Per faccia
  - ❖ prodotto vettore di due edge della faccia (attenti alla regola della mano destra)
- ❖ Per vertice
  - ❖ sfruttando la conoscenza della geometria dell'oggetto (e.g. una sfera)
  - ❖ mediando le normali tra le facce che incidono su un dato vertice



## Definizione di Luci in OpenGL

Per ogni luce occorre definire:

- ❖ Posizione  
`glLightfv(GL_LIGHT0, GL_POSITION, position);`
- ❖ Colore componente diffusa e speculare  
`glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);`  
`glLightfv(GL_LIGHT0, GL_SPECULAR, specular);`
- ❖ abilitare ogni luce  
`glEnable(GL_LIGHT0);`
- ❖ e poi abilitare il calcolo dell'illuminazione  
`glEnable(GL_LIGHTING);`

## Definizione di luci in OpenGL

**Nota:**

- ❖ il numero di luci è limitato (`GL_MAX_LIGHT=8` su MS OpenGL)
- ❖ Le luci possono essere poste all'infinito (coordinate omogenee)  
`v=(0,0,1,0);`  
`glLightfv(GL_LIGHT0, GL_POSITION, v);`
- ❖ Possono essere spot (definibile direzione, angolo e velocità di cut-off)  
`glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, v);`  
`glLightfv(GL_LIGHT0, GL_SPOT_CUTOFF, v);`  
`glLightfv(GL_LIGHT0, GL_SPOT_EXPONENT, v);`
- ❖ Possono avere o no attenuazioni in distanza.  
`glLightfv(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1);`  
`glLightfv(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 1);`

## Default Luci opengl

- ❖ Di default in opengl la luce 0 e' creata come direzionale lungo la z neg.
- ❖ Basta abilitare la luce e il lighting e la normalizzazione automatica delle normali  
`glEnable(GL_LIGHT0);`  
`glEnable(GL_LIGHTING);`  
`glEnable(GL_NORMALIZE);`
- ❖ Dove?
  - ❖ Da qualche parte, anche una sola volta.
  - ❖ Importante DOPO che il contesto opengl sia stato creato (non nei costruttori...)

## Tipi di Shading in OpenGL

### ❖ Flat Shading

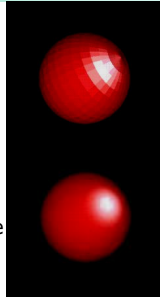
- ❖ tutto il poligono ha lo stesso colore (una sola normale per triangolo)

```
glShadeModel(GL_FLAT);
```

### ❖ Gouraud Shading

- ❖ i colori dei vertici sono interpolati linearmente per tutto il poligono (una normale per ogni vertice)

```
glShadeModel(GL_SMOOTH);
```



## Proprietà Materiali

- ❖ I materiali possono essere specificati per la front, la back o entrambe le face di un dato poligono

- ❖ (il parametro **face** può assumere i seguenti valori **GL\_FRONT**, **GL\_BACK**, **GL\_FRONT\_AND\_BACK**)

- ❖ Si specificano le varie componenti

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

## Esempio disegnare un cilindro

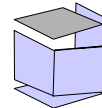
- ❖ Non vogliamo usare oggetti predefiniti
- ❖ Cilindro simmetrico rispetto all'Asse Y
- ❖ Occorre definire una sola strip di triangoli



- ❖ Le normali sono perpendicolari all'asse Y

## Blocco shaded

- ❖ Occorre definire anche le normali quando si disegna il cubetto
- ❖ Disegniamo il cubetto come una quad strip e due quad.



- ❖ Per ogni quad diamo una normale

## SolidBlock

```
void DrawSolidBlock(float xss, float yss, float zss)  
{  
    glBegin(GL_QUAD_STRIP);  
    glNormal3f(0f, 0f, -1.0f);  
    glVertex3f(xss, yss, zss);  
    glVertex3f(-xss, -yss, zss);  
    glNormal3f(0f, 0f, 0f);  
    glVertex3f(xss, yss, -zss);  
    glVertex3f(-xss, -yss, -zss);  
    glNormal3f(0f, 0f, 1.0f);  
    glVertex3f(xss, yss, zss);  
    glVertex3f(-xss, -yss, zss);  
    glEnd();  
  
    glBegin(GL_QUAD);  
    glNormal3f(0f, 1.0f, 0f);  
    glVertex3f(-xss, yss, zss);  
    glVertex3f(xss, yss, zss);  
    glVertex3f(xss, -yss, zss);  
    glVertex3f(-xss, -yss, zss);  
    glEnd();  
  
    glNormal3f(0f, -1.0f, 0f);  
    glVertex3f(-xss, -yss, -zss);  
    glVertex3f(xss, -yss, -zss);  
    glVertex3f(xss, yss, -zss);  
    glVertex3f(-xss, yss, -zss);  
    glEnd();  
}
```

## Zbuffer

- ❖ Non basta, perche' sia visualizzato correttamente occorre usare lo zbuffer

- ❖ Chiedere un contesto opengl con lo zbuffer

- ❖ Meccanismo in generale OS dependent

- ❖ In sdl prima di fare `SDL_SetVideoMode`

```
SDL_GL_SetAttribute(SDL_GL_DEPTH_SIZE, 24);
```

- ❖ Abilitare l'operazione di depthtest durante la rasterizzazione

```
glEnable(GL_DEPTH_TEST);
```

## Cambiamo di colore

- ❖ Per cambiare di colore agli oggetti quando l'illuminazione e' abilitata occorre cambiare materiale:
  - ❖ glMaterialfv( ....)
  - ❖ glColor ma solo se stiamo usando glColorMaterial

## glMaterialf

```
-
float diffrd[4]= (1.0f,0.4f,0.4f,1);
float diffgreen[4]=0.4f,1.0f,0.4f,1);
float spec[4]={1,1,1,1};
float shin = 4;

glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE , diffrd );
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR , spec);
glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS , shin);
DrawMoebiusRing(18,5,1);

// per il cubetto cambio soltanto la componente diffuse
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE , diffgreen );
DrawMoebiusRollingBall(5,1, CurAngle, .5);
-
```

## glColorMaterial

- ❖ Occorre
  - ❖ Abilitare il meccanismo glColorMaterial
  - ❖ glEnable(GL\_COLOR\_MATERIAL)
  - ❖ Dire i prossimi comandi glColorxx quale componente del materiale cambiano
  - ❖ Cambiare colore delle componenti scelte come desiderato

## glColorMaterial

```
float spec[4]={0.3, 0.3, 0.3,1};
float shin = 4;
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR , spec);
glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS , shin);

glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE );
glEnable(GL_COLOR_MATERIAL);

glColor3f(1, .4, .4);
DrawMoebiusRing(18,5,1);

glColor3f(.4, .4, 1);
DrawMoebiusRollingBall(5,1, CurAngle, .5);
```

## Posizione delle luci

- ❖ La posizione effettiva della luce in spazio di camera viene calcolata applicando la matrice corrente nel momento in cui tale posizione viene specificata.

## Luci e sistemi di riferimento

- ❖ Luce come il default di opengl

```
float lightpos[4]={0,0,1,0};
glLoadIdentity();
glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
gluLookAt(2,5,12,0,0,0,1,0);
glRotatef(-CurAngle, 0,1,0);

glColor3f(1, .4, .4);
DrawMoebiusRing(18,5,1);

glColor3f(.4, .4, 1);
DrawMoebiusRollingBall(5,1, CurAngle, .5);
```

## Luci e sistemi di riferimento

### ❖ Luce laterale solidale all'osservatore

```
float lightpos[4]={1,0,0,0};
glLoadIdentity();
glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
gluLookAt(2,5,12,0,0,0,1,0);
glRotatef(-CurAngle,0,1,0);

glColor3f(1,.4,.4);
DrawMoebiusRing(18,5,1);

glColor3f(.4,.4,1);
DrawMoebiusRollingBall(5,1,CurAngle,.5);
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

25

## Luci e sistemi di riferimento

### ❖ Luce solidale all'oggetto che ruota

```
float lightpos[4]={0,0,1,0};
glLoadIdentity();
gluLookAt(2,5,12,0,0,0,1,0);
glRotatef(-CurAngle,0,1,0);
glLightfv(GL_LIGHT0, GL_POSITION, lightpos);

glColor3f(1,.4,.4);
DrawMoebiusRing(18,5,1);

glColor3f(.4,.4,1);
DrawMoebiusRollingBall(5,1,CurAngle,.5);
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

26

## Luci e sistemi di riferimento

### ❖ Luce non direzionale solidale con l'osservatore

```
float lightpos[4]={0,0,0,1};
glLoadIdentity();
glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
gluLookAt(2,5,12,0,0,0,1,0);
glRotatef(-CurAngle,0,1,0);

glColor3f(1,.4,.4);
DrawMoebiusRing(18,5,1);

glColor3f(.4,.4,1);
DrawMoebiusRollingBall(5,1,CurAngle,.5);
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

27

## Luci e sistemi di riferimento

### ❖ Luce nel centro dell'oggetto che ruota

```
float lightpos[4]={0,0,0,1};
glLoadIdentity();
gluLookAt(2,5,12,0,0,0,1,0);
glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
glRotatef(-CurAngle,0,1,0);

glColor3f(1,.4,.4);
DrawMoebiusRing(18,5,1);

glColor3f(.4,.4,1);
DrawMoebiusRollingBall(5,1,CurAngle,.5);
```

17 Ott 2003

Costruzione di Interfacce - Paolo Cignoni

28