- INTRODUZIONE
- DRAWING
- EVENT MANAGEMENT
- VIEWING
- DOUBLE BUFFERING
- Z-BUFFERING
- LIGHTING

LIGHTING

- e` una approssimazione del comportamento della luce nel mondo reale
- permette di visualizzare la scena in modo realistico
- utilizzando il lighting il colore degli oggetti dipende da:
  - le normali sui vertici
  - le sorgenti luminose
  - le proprieta` di materiale degli oggetti
  - il modello di illuminazione

NORMALI SUI VERTICI

- l'incidenza della luce sulle primitive viene determinata dalle normali sui vertici
- alternative possibili
  - calcolare esplicitamente le normali sui vertici del modello che si descrive
  - appoggiarsi ad una libreria che fornisca primitive solide dotate di normali precalcolate (es. glut)

ESERC: LIGHTING

- modificare esercizio precedente
- eliminare definizioni colori
- aggiungere:

```
glEnable( GL_LIGHTING );
glEnable( GL_LIGHT0 );
```

  prima del drawing
- consiglio:
  - porre nella scena una **`glutSolidSphere`**`()`
- N.B.: ogni modalita` abilitata con:

```
glEnable( <MODE> );
```

  puo` essere disabilitata con:

```
glDisable( <MODE> );
```

SORGENTI LUMINOSE

- per ogni sorgente e` possibile definire una serie di proprieta`

```
glLightfv(light_num,property,parameters);
```

- **`light_num`** va da `GL_LIGHT0` a `GL_LIGHT7`

- **`property`** (colore della sorgente)
  - `GL_AMBIENT` determina il contributo alla luce ambientale, espresso come un vettore rgba
  - `GL_DIFFUSE` determina il colore della luce stessa, espresso come un vettore rgba
  - `GL_SPECULAR` determina il colore riflesso dagli highlight, espresso come un vettore rgba (tipicamente e` lo stesso colore della luce diffusa)

SORGENTI (cont.)

- **`property`** (geometria della sorgente)
  - GL_POSITION e` la posizione della sorgente espressa come un vettore xyzw. Se w=0 la luce e` direzionale (a distanza infinita) e xyz e` il vettore direzione, altrimenti e` posizionale e xyzw sono le coordinate omogenee.
  - GL_SPOT_DIRECTION e` la direz. di una spotlight (luce posizionale) espressa come un vettore xyz
  - GL_SPOT_CUTOFF e` il semiangolo di apertura della spotlight, nel range 0.0 .. 90.0. Il valore speciale 180.0 significa tutte le direzioni
  - GL_SPOT_EXPONENT determina la distribuzione della luce nel cono

SORGENTI (cont.)

- **`property`** (attenuazione luce emessa)
  - GL_CONSTANT_ATTENUATION e` il fattore di attenuazione costante
  - GL_LINEAR_ATTENUATION e` il fattore di attenuazione lineare
  - GL_QUADRATIC_ATTENUATION e` il fattore di attenuazione quadratica

(solo per luci posizionali)

- ogni sorgente definita va abilitata

```
glEnable( light_num )
```


ESERC.: SORGENTI

- ridefinire `GL_LIGHT0` come una luce POSIZIONALE, posta in una posizione a piacere purche` FISSA rispetto alla scena
  - notare come in posizione opposta alla luce gli oggetti rimangono oscurati
- soluzione

```
void draw( )
{
GLfloat l0pos[] = { 0.0, 50.0, 0.0, 1.0 };
        ...
        glEnable( GL_LIGHTING );
        ...

        /* light sources */

        glLightfv(GL_LIGHT0,GL_POSITION,l0pos);
        glEnable( GL_LIGHT0 );
        ...
}
```

ESERC.: SORGENTI (cont.)

- ridefinire `GL_LIGHT0` come una spot light puntata verso la scena e con un angolo di apertura opportuno
- soluzione

```
void draw( )
{
GLfloat l0pos[] = { 0.0,  200.0, 0.0, 1.0 };
GLfloat l0dir[] = { 0.0,   -1.0, 0.0, 1.0 };
GLfloat l0cut[] = { 20.0 };

  ...

  /* light sources */

  glLightfv(GL_LIGHT0,GL_POSITION,      l0pos);
  glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,l0dir);
  glLightfv(GL_LIGHT0,GL_SPOT_CUTOFF,   l0cut);
  glEnable (GL_LIGHT0 );

  ...
}
```


ESERC.: SORGENTI (cont.)

- ridefinire `GL_LIGHT0`
  - luce ambient., diffusa, speculare: verde
- definire `GL_LIGHT1`
  - posizione differente
  - luce ambient, diffusa, speculare: cyan (verde+blu)

ESERC.: SORGENTI (cont.)

```
void draw( )
{
GLfloat l0pos[] = { 0.0,  200.0, 0.0, 1.0 };
GLfloat l0dir[] = { 0.0, -200.0, 0.0, 1.0 };
GLfloat l0cut[] = { 30.0 };
GLfloat l0amb[]  = { 0.0, 1.0, 0.0, 1.0 };
GLfloat l0diff[] = { 0.0, 1.0, 0.0, 1.0 };
GLfloat l0spec[] = { 0.0, 1.0, 0.0, 1.0 };

GLfloat l1pos[] = {  200.0, 0.0, 0.0, 1.0 };
GLfloat l1dir[] = { -200.0, 0.0, 0.0, 1.0 };
GLfloat l1cut[] = { 10.0 };
GLfloat l1amb[]  = { 0.0, 1.0, 1.0, 1.0 };
GLfloat l1diff[] = { 0.0, 1.0, 1.0, 1.0 };
GLfloat l1spec[] = { 0.0, 1.0, 1.0, 1.0 };
  ...
  glLightfv(GL_LIGHT0,GL_POSITION,      l0pos );
  glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,l0dir );
  glLightfv(GL_LIGHT0,GL_SPOT_CUTOFF,   l0cut );
  glLightfv(GL_LIGHT0,GL_AMBIENT,       l0amb );
  glLightfv(GL_LIGHT0,GL_DIFFUSE,       l0diff);
  glLightfv(GL_LIGHT0,GL_SPECULAR,      l0spec);
  glEnable (GL_LIGHT0 );

  glLightfv(GL_LIGHT1,GL_POSITION,      l1pos );
  glLightfv(GL_LIGHT1,GL_SPOT_DIRECTION,l1dir );
  glLightfv(GL_LIGHT1,GL_SPOT_CUTOFF,   l1cut );
  glLightfv(GL_LIGHT1,GL_AMBIENT,       l1amb );
  glLightfv(GL_LIGHT1,GL_DIFFUSE,       l1diff);
  glLightfv(GL_LIGHT1,GL_SPECULAR,      l1spec);
  glEnable (GL_LIGHT1 );
  ...
}
```


PROPRIETA` DI MATERIALE

- per ogni faccia e` possibile definire una serie di proprieta` di materiale

```
glMaterialfv(face,property,parameters);
```

- **face** puo` essere `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`
- **property**
  - `GL_AMBIENT` determina la percentuale riflessa della luce ambiente, espressa come un vettore rgba
  - `GL_DIFFUSE` determina la percentuale riflessa della luce diffusa, espressa come un vettore rgba
  - `GL_AMBIENT_AND_DIFFUSE` determina la percentuale riflessa di entrambe le luci ambiente e diffusa, espresse tramite lo stesso vettore rgba

MATERIALE (cont.)

- **`property`**
  - `GL_SPECULAR` determina la percentuale riflessa dagli highlight della luce speculare, tramite un vettore rgba
  - `GL_EMISSION` determina il colore della luce emessa, tramite un vettore rgba
  - `GL_SHININESS` e` il "coefficiente di lucidita`" del materiale, espresso in un range 0.0 .. 128.0

- assegnando dei valori opportuni, e` possibile simulare il comportamento di materiali reali

ESERC.: MATERIAL

- ridefinire entrambe le luci come luce bianca:
  - ambient 0.2, 0.2, 0.2, 1.0
  - diffuse 1.0, 1.0, 1.0, 1.0
  - specular 1.0, 1.0, 1.0, 1.0

- soluzione

```
...
GLfloat l0amb[]  = { 0.2, 0.2, 0.2, 1.0 };
GLfloat l0diff[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat l0spec[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat l1amb[]  = { 0.2, 0.2, 0.2, 1.0 };
GLfloat l1diff[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat l1spec[] = { 1.0, 1.0, 1.0, 1.0 };
...
```

ESERC.: MATERIAL (cont.)

- definire differenti proprieta` di materiale per gli oggetti nella scena in modo che risultino di colori differenti

ESERC.: MATERIAL (cont.)

```
void draw( )
{
...
GLfloat mat1amb[]   = { 0.0, 0.3, 0.0, 1.0 };
GLfloat mat1diff[]  = { 0.0, 0.5, 0.0, 1.0 };
GLfloat mat1spec[]  = { 0.0, 1.0, 0.0, 1.0 };
GLfloat mat1shine[] = { 80.0 };

GLfloat mat2amb[]   = { 0.3, 0.0, 0.0, 1.0 };
GLfloat mat2diff[]  = { 0.5, 0.0, 0.0, 1.0 };
GLfloat mat2spec[]  = { 1.0, 0.0, 0.0, 1.0 };
GLfloat mat2shine[] = { 80.0 };
  ...
  glMaterialfv(GL_FRONT,GL_AMBIENT,  mat1amb  );
  glMaterialfv(GL_FRONT,GL_DIFFUSE,  mat1diff );
  glMaterialfv(GL_FRONT,GL_SPECULAR, mat1spec );
  glMaterialfv(GL_FRONT,GL_SHININESS,mat1shine);

  glPushMatrix();
  glScalef( 60.0, 10.0, 50.0 );
  glutSolidCube( 1.0 );
  glPopMatrix();

  glMaterialfv(GL_FRONT,GL_AMBIENT,  mat2amb  );
  glMaterialfv(GL_FRONT,GL_DIFFUSE,  mat2diff );
  glMaterialfv(GL_FRONT,GL_SPECULAR, mat2spec );
  glMaterialfv(GL_FRONT,GL_SHININESS,mat2shine);

  glPushMatrix();
  glTranslatef( -20.0, 20.0, 0.0 );
  glScalef( 20.0, 20.0, 20.0 );
  glutSolidIcosahedron( );
  glPopMatrix();
  ...
```

MODELLO DI ILLUMINAZIONE

- luce ambientale

  componente della ambient light non dipendente dalle sorgenti definite

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,rgba);
```

  dove **`rgba`** e` un vettore di 4 `GLfloat`

- osservatore locale o a distanza infinita

  a distanza infinita i tempi di calcolo sono inferiori ma i risultati meno realistici

```
glLightModeli( GL_LIGHT_MODEL_LOCAL_VIEWER,
                                    flag );
```

  dove **`flag`** e` `GL_TRUE` o `GL_FALSE`

MODELLO DI ILLUMIN. (cont.)

- one-sided or two-sided lighting

  determina su quale lato delle primitive calcolare l'illuminazione

```
glLightModeli( GL_LIGHT_MODEL_TWO_SIDE,
                                    flag );
```

  dove **`flag`** e` `GL_TRUE` o `GL_FALSE`

ESERC.: LIGHTMODEL

- definire la luce ambientale minima e massima
  - notare come con sufficiente luce ambientale i modelli siano illuminati anche sul lato opposto alle sorgenti luminose (dipendentemente dalle proprieta` di materiale)
- definire l'osservatore a distanza infinita e locale

ESERC.: LIGHTMODEL (cont.)

```
void draw( )
{

...

GLfloat lmodamb[]   = { 0.3, 0.3, 0.3, 1.0 };

  ...

  /* light model */

  glLightModelfv( GL_LIGHT_MODEL_AMBIENT,
                  lmodamb );
  glLightModeli ( GL_LIGHT_MODEL_LOCAL_VIEWER,
                  GL_TRUE );

  ...
}
```

LIGHTING (cont.)

- inoltre:
  - e` preferenziale l'utilizzo della modalita` RGB
  - il lighting deve essere abilitato con:

```
glEnable( GL_LIGHTING );
```

- qualsiasi modalita` abilitata con:

```
glEnable( <MODE> );
```

puo` essere disabilitata con:

```
glDisable( <MODE> );
```


NORMALI (cont.)

- e` possibile definire una normale differente per ogni vertice:

```
void glNormal*( )
```

determina la normale sui vertici definiti successivamente

- le normali devono avere lunghezza unitaria (normali normalizzate):

```
void glEnable( GL_NORMALIZE )
```

richiede la normalizzazione automatica delle normali definite successivamente (costoso)

ESERC.: NORMALI

- visualizzare un poligono definendo esplicitamente le normali
  - togliere dalla scena le primitive glut
  - disegnare un triangolo in posizione di comodo (avente normale nota)
- soluzione

  (utilizziamo per comodita` le funzioni con segnatura 3fv)

```
...
GLfloat  n[3] = {   0.0, 1.0,   0.0 };
GLfloat v1[3] = {   0.0, 0.0,  70.0 };
GLfloat v2[3] = {  50.0, 0.0, -50.0 };
GLfloat v3[3] = { -50.0, 0.0, -50.0 };
...
        glBegin(GL_TRIANGLES);
                glNormal3fv( n  );
                glVertex3fv( v1 );
                glVertex3fv( v2 );
                glVertex3fv( v3 );
        glEnd();
```


CALCOLO NORMALI

- dati v_1, v_2, v_3 vertici non allineati di un poligono

  $[u] = [v_1 - v_2]$ **e** $[w] = [v_2 - v_3]$

  sono due vettori coplanari ma non paralleli

- il prodotto vettoriale di due vettori u, w coplanari ma non paralleli e` una normale al piano

  $[n] = [u] \times [w] =$

  $= [\, u_y w_z - w_y u_z,\ w_x u_z - u_x w_z,\ u_x w_y - w_x u_y \,]$

- determinare la normale di un poligono (triangolo) generico e disegnare il poligono stesso

ESERC.: NORMALI (cont.)

```
GLfloat n[3];
GLfloat u[3];
GLfloat w[3];
GLfloat v1[3] = { 60.0,  0.0,  0.0 };
GLfloat v2[3] = {  0.0, 30.0,  0.0 };
GLfloat v3[3] = {  0.0,  0.0, 50.0 };
...
u[0] = v1[0]-v2[0]; /* un lato */
u[1] = v1[1]-v2[1];
u[2] = v1[2]-v2[2];

w[0] = v2[0]-v3[0]; /* un altro lato */
w[1] = v2[1]-v3[1];
w[2] = v2[2]-v3[2];

n[0] = u[1]*w[2]-w[1]*u[2]; /* la normale */
n[1] = w[0]*u[2]-u[0]*w[2];
n[2] = u[0]*w[1]-w[0]*u[1];

     glEnable( GL_NORMALIZE );
     glBegin(GL_TRIANGLES);
              glNormal3fv( n  );
              glVertex3fv( v1 );
              glVertex3fv( v2 );
              glVertex3fv( v3 );
     glEnd();
```

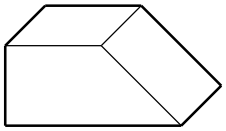

ESERC.: NORMALI (cont.)

- disegnare un poligono avente valori differenti delle normali sui vertici
- soluzione

```
...
 GLfloat n1[3] = { 0.0, 1.0, 0.0 };
 GLfloat n2[3] = { 1.0, 0.0, 0.0 };
 GLfloat v1[3] = {   0.0, 0.0,  70.0 };
 GLfloat v2[3] = {  50.0, 0.0, -50.0 };
 GLfloat v3[3] = { -50.0, 0.0, -50.0 };
...
        glBegin(GL_TRIANGLES);
                 glNormal3fv( n1 );
                 glVertex3fv( v1 );
                 glVertex3fv( v2 );
                 glNormal3fv( n2 );
                 glVertex3fv( v3 );
        glEnd();
```

ESERC.: NORMALI (last)

- disegnare una semplice superficie poliedrica chiusa
  - descrivere la superficie tramite poligoni o triangoli
  - determinare le normali sui vertici dei poligoni / triangoli definiti
  - utilizzare i dati determinati, per effettuare il drawing
- Es.:



- Istanziare ripetutamente la primitiva creata
  - differenti valori di traslazione / rotazione
  - differenti proprieta` di materiale

ALTRI COMANDI

- filling dei poligoni:

  `glPolygonMode``(`**`<FACE>`**`,`**`<MODE>`**`);`

  parametri possibili (**`<FACE>`**):

  `GL_FRONT`
  `GL_BACK`
  `GL_FRONT_AND_BACK`

  parametri possibili (**`<MODE>`**):

  `GL_POINT`
  `GL_LINE`
  `GL_FILL`

- tipo di shading utilizzato:

  `glShadeModel``(` **`<MODE>`** `);`

  parametri possibili:

  `GL_SMOOTH`
  `GL_FLAT`

ESERC.: RIEPILOGO

- disegnare una scena a piacere utilizzando tutte le nozioni viste in precedenza
  - proiezione prospettica
  - double buffering e z-buffering
  - interazione tramite tastiera o mouse
  - lighting