A Steroid V0.2

Paolo Cignoni
cignoni@iei.pi.cnr.it
http://vcg.iei.pi.cnr.it/~cignoni

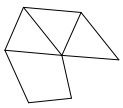
26 ottobre 1999

A Steroid
Seconda Versione

- Da 2D a 3D
- Nuove Classi per memorizzare, caricare, visualizzare una mesh di triangoli
- Classe Ship3D
- Classe Ast3D

Rappresentazione Superfici

- Ogni superficie può essere approssimata con facce planari (piu' se ne usa migliore l'approssimazione)
- Facce triangolari sono la scelta più comune
- Elementi di mesh poligonale:
  - facce (face)
  - spigoli (edge)
  - vertici (vertex)



Poligoni in OpenGL

- Sono definiti per vertici:

```
glBegin(mode);
   glVertex3f(x1,y1,z1);
   glVertex3f(x2,y2,z2)
   ...
glEnd();
```

- Per ogni vertice possiamo anche definire normale e colore (se cambiano)

Poligoni in OpenGL

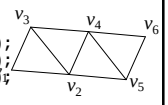
-
```
glBegin(mode);
    glVertex3f(x1,y1,z1);
    glVertex3f(x2,y2,z2)
    ...
glEnd();
```

- il parametro mode può assumere i seguenti valori:
- `GL_POINTS,`
- `GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP,`
- `GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN`
- `GL_QUADS, GL_QUAD_STRIP, GL_POLYGON`

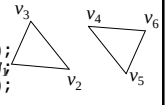
Triangle Strip

- Modo efficiente per disegnare una serie di triangoli:

```
glBegin(GL_TRIANGLE_STRIP);
   glVertex3fv(v1); glVertex3fv(v2);
   glVertex3fv(v3); glVertex3fv(v4);
   glVertex3fv(v5); glVertex3fv(v6);
glEnd();
```




```
glBegin(GL_TRIANGLES);
   glVertex3fv(v1); glVertex3fv(v2);
   glVertex3fv(v3); glVertex3fv(v4);
   glVertex3fv(v5); glVertex3fv(v6);
glEnd();
```

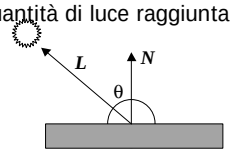


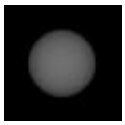
Illuminazione della Superfice

- Luce emessa
- Riflessione diffusa
- Riflessione speculare
- Luce ambiente

Componente Diffusa

- Modello Lambert: la superficie riflette la luce nella stessa misura in ogni direzione.
- Superficie perfettamente opaca
- La luminosità percepita dipende solo dalla quantità di luce raggiunta dalla superficie



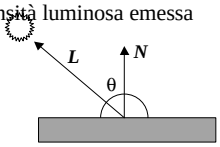


Componente Diffusa

- Modello Lambertiano

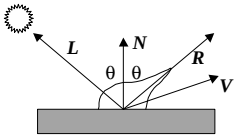
$$I_D=K_D(N\bullet L)\ I_L$$

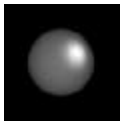
- I_D intensità luminosa percepita
- K_D coefficiente di riflessione proprio del materiale
- I_L intensità luminosa emessa



Componente Speculare

- Oggetti lucidi (plastica, metalli)
- La luce non è diffusa uniformemente in tutte le direzioni ma segue la legge di riflessione



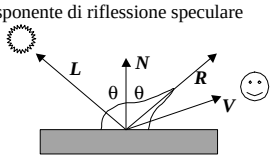


Componente Speculare

- Modello Phong

$$I_S=K_S(V\bullet R)^n\ I_L$$

- K_S coefficiente di riflessione speculare del materiale
- V direzione di vista
- n esponente di riflessione speculare



Componente Emissiva

- Quantità di luce emessa dall'oggetto stesso
- (e.g. tubi al neon ecc.)

Componente Ambientale

- Quantità di luce riflessa dall'oggetto e dovuta alle interriflessioni tra oggetti
- Per semplicità si assume che sia uniforme
  - in tutte le direzioni
  - in tutti i punti della scena
  - E' un'approssimazione molto grezza!

Equazione di Shading

- Una sola luce:

$$I = I_E + I_A + I_D + I_S =$$
$$= I_E + I_A + K_D(N \bullet L)\, I_L + K_S(V \bullet R)^n\, I_L$$

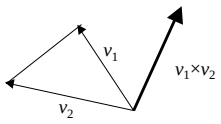
- Per più luci si sommano i vari contributi di I_D+I_S relativi ad ogni luce

Disegnare un oggetto ombreggiato

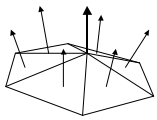
- Definire i vettori normali per ogni vertice dell'oggetto
- Creare e abilitare una o più luci
- Segliere un modello di illuminazione e di shading
- Definire le proprietà dei materiali

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);
```

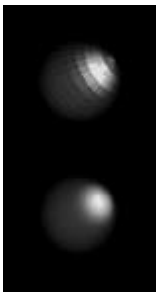
Tipi di Shading in OpenGL

- 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 (e.g quelli di glut)
- Asse Y
- Occorre definire una sola strip di triangoli

- Le normali sono perpendicolari all'asse Y

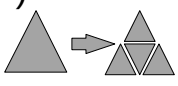
Definizione Triangle e Mesh

- Classe Triangle
  - memorizza le tre coordinate e precalcola la normale
- Classe Mesh
  - Memorizza una mesh di triangoli come un vettore di Triangle
  - Carica una Mesh da file in formato RAW
  - Disegna la mesh

Gestione Luci e Oggetti

- Le luci e le varie modalità di shading sono settate una volta per tutte nella myGLInit()
- Ogni oggetto carica la propria mesh nel costruttore
- Il rendering e' fatto tramite le funzioni della classe Mesh()

Esercizio 1 (***)



- Asteroide Frattale3D
  - Scrivere la
    class AstFract3D : public Ast
    Il cui costruttore ha una procedura che genera una superficie frattale. Partire da un semplice solido (un ottaedro) con facce triangolari e sostituire ricorsivamente ogni triangolo con quattro triangoli piu' piccoli. Al solito i nuovi vertici vengono spostati/perturbati di una piccola quantita' casuale.
  - Attenzione i nuovi vertici sono sempre condivisi quindi si deve fare il perturbamento uguale per ogni edge della mesh (hint fare una rand che dipende dalle coordinate dei vertici dell'edge da splittare).
  - interessante sarebbe quella realizzare quanto sopra tramite una funzione Mesh::Fractalize(int recurs)...

Esercizio 2 (***)

- Nuova Astronave con colore
  - Modificare la classe Mesh per aggiungere la gestione del colore per faccia, sia in loading (modificate il formato raw) che in rendering.
  - Disegnarsi un'astronave ben colorata. Suggerimenti:
    - Usare uno dei vari modellatori 3d (vedi alla pagina risorse) convertire in qualche formato ascii e cercare di convertire a mano il file risultante.
    - Modellarsi l'astronave a mano (carta e matita...)
    - Nota usate un numero basso (max 50) poligoni!!!

Esercizio 3 (***)

- Strafe dell'astronave
- Aggiungere all'astronave la possibilita' di spostarsi lateralmente (strafe a là Doom o Quake): *Premendo ALT + frecce laterali l'astronave deve iniziare ad accellerare lateralmente*
- Modificare la classe base ship per includere la gestione dello strafe e le funzioni callback glut per la gestione della tastiera.

Esercizio 4 (**)

- Debugging
- Ci sono una serie di errori (tutti involontari ma alcuni volontariamente non corretti) che vanno trovati e corretti.

Risorse

- 3D Modeller Freeware and Shareware
  - Ac3d (Win, Linux)
    http://www.comp.lancs.ac.uk/computing/users/andy/ac3dpov.html
  - Blender (Win, Linux)
    http://www.blender.nl/download/download.html
  - sPatch (Only Win) http://www.cableone.net/alyson/spatch.html
  - Extreme Wave (Win, Linux)
    http://agnews.tamu.edu/~jpalmer/ewave/#downloads
  - Rhino3d (Only Win)
    http://www.rhino3d.com/ (Win)

Risorse

- Links
  - http://www.povray.org/links/3D_Programs/Modelling_Programs/
  - http://sgic.linuxberg.com/x11html/gra_3d.html
  - http://www.opengl.org/Products/Applications/Modeling.html
  - http://www-personal.umich.edu/~jeffab/links/modellinks.html
- 3D Converters
  - 3d Win (Win/Linux)
    http://www.stmuc.com/thbaier/index.html
  - CrossRoads (Win)
    http://www.europa.com/~keithr/crossroads/
  - VRML2pov (Win/Linux)
    http://www.chemicalgraphics.com/paul/vrml2pov/index.html