A Steroid V0.2s e V0.3

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

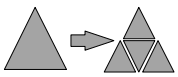
2 novembre 1999

A Steroid v0.2s

- The shape of the asteroids is randomly generated
- The ship class include side movement member functions
- The mesh class can manage colors and smoothed normals

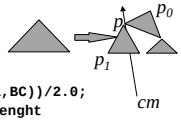
Asteroide Frattale

- Uso struttura mesh
- Si parte da una mesh semplice (un tetraedro) e si suddivide ogni triangolo in quattro triangoli piu' piccoli



Fractal Asteroid

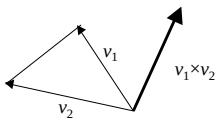
- Ogni nuovo punto *p* viene spostato dalla sua posizione lungo la retta *cm-p* di una quantita' casuale dipendente dalle posizioni di p_1 e p_2



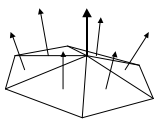

```
Point3f mp=(p0+p1)/2.0;
float r=(Distance(p0,BC)+Distance(p1,BC))/2.0;
float l=Distance(p0,p1);   // Edge Lenght
l=l*strenght;
srand48(seed + p0.Hash() + p1.Hash());
// Displacement depends only on the two vertexes p0,p1
float d=drand48()-0.5f;
mp=BC+Normalize(mp-BC)*(r+d*l/2.0);
```

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



Calcolo Normali Smooth

- Nella nostra struttura Mesh non abbiamo modo di sapere quali facce condividono uno stesso vertice.
  - Per mediare le normali occorre fare un'unificazione dei vertici.
  - Puo' essere fatta in maniera implicita facendo uso dei container associativi delle STL.

STL MAP

- Una map<key,data> è un Associative Container che associa oggetti di tipo Key con oggetti di tipo Data.
- Il modo più semplice di pensare una map è considerarlo come un vettore i cui elementi sono indiciati da oggetti di tipo key invece che interi.
- STL incoraggia la metafora permettendo l'uso delle [] per accedere agli elementi della map.

Stl Map

- Nel nostro caso occorre una
  - map<vertici,normali> NN;
  - e il codice per sommare tutte le normali incidenti in un vertice diventa:

```
map<Point3f,Point3f> NN; //the vertex->normal map
vector<Triangle>::iterator i;
int j;

// accumulate normals per vertex
for(i=T.begin();i!=T.end();++i)
    for(j=0;j<3;++j)
          NN[(*i).v[j]]+=(*i).n[j];
```

A Steroid Terza Versione

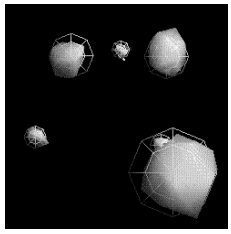
- Bounding objects
- Collision Detection
- Esplosioni

Collision detection

- Occorre capire quando un proiettile colpisce un asteroide e quando un asteroide investe l'astronave
  - si potrebbe risolvere calcolando l'intersezione di ogni singolo triangolo dell'asteroide con ogni singolo triangolo dell'astronave
  - La soluzione esatta del problema non ci importa (in questo caso)
  - Si utilizza al posto dell'astronave e dell'asteroide una loro rappresentazione MOLTO approssimata

Collision Detection

- Per ogni GameObj si tiene una bounding sphere (sfera di contenimento) che racchiude l'oggetto stesso.



Collision Detection

- La collisione si controlla solo tra le varie bounding spheres:

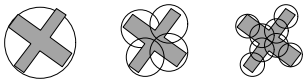
```
bool GameObj::Collide(GameObj *o){
            B.c = p;
            o->B.c = o->p;
            return B.Intersect(o->B);
}
```

dove

```
inline bool Sphere3f::Intersect(
                  Sphere3f const & s) const {
      if((c-s.c).Norm()< r+s.r) return true;
      else return false;
  }
```

Collision Detection

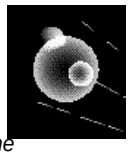
- Un'approssimazione migliore puo venire dall'utilizzo di gerarchie di bounding object, che approssimano l'oggetto in maniera sempre migliore



- Nel gioco useremo solo bounding sphere semplici (per ora).

Esplosioni

- Realizzate tramite un disco *trasparente* al centro e giallo opaco ai bordi, che cresce. Realizzato con una singola triangle_fan
  - All'interno dell'esplosione si vedono gli altri oggetti
  - piu' esplosioni si sommano *bene*



Gestione trasparenza in opengl

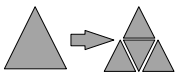
- In opengl il colore e' una quadrupla ***<r,g,b,a>***
- dove ***a*** alpha o opacita'
  - alpha==1 totalmente opaco
  - alpha==0 totalmente trasparente
- quando si disegna un triangolo trasparente il suo colore viene mixato con il colore del fondo. Opengl permette di decidere come viene mescolato:
  - `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)`
- fa si che il colore risultante da un triangolo con alpha = a sia colore_tri * a + colore_fondo *(1-a)

Problema ordinamento

- cambiando l'ordine con cui si disegna gli oggetti semitrasparenti cambia il risultato. Ad es:
  - Se il depth buffering e' abilitato se disegno prima un vetro e poi gli oggetti dietro non vedo i secondi (il depth test ne impedisce il disegno)
- Soluzione corretta disegnando gli oggetti semi trasparenti in ordine di profondita'

Esercizio 1 (*)

- Asteroide Frattale3D
  - Modificare la class AstFract3D : public Ast in maniera tale che la perturbazione del punto sia fatta a caso sul piano perpendicolare all'edge da dividere anziche' nella direzione centro di massa -- splitpoint



Esercizio 2 (**)

- Nuovo proiettile con esplosione finale FireBallBullet
- Il proiettile si comporta normalmente per 3 sec poi esplode lanciando +50 proiettilini a raggiera che pero' durano solo 0.5 sec. (e.g. tipo fuoco d'artificio).
  - hint: i proiettilini secondari non sono nuovi gameobj ma solo semplici righe disegnate dalla FireBallBullet::Draw. in compenso varia il raggio del bounding sphere del bullet in questione...

Esercizio 3 (***)

- Esplosioni con luci
- Ogni esplosione genera una Opengl light posizionata nel punto dell'esplosione per la durata dell'esplosione.
  - hint: in opengl le luci sono limitate (in genere 8). Far sì che le esplosioni usino solo tre luci tenendo un vettorino statico di classe per sapere quale delle tre e' libera.