A Steroid V0.3s e V0.4

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

16 novembre 1999

Esplosioni con Luci

- In opengl le luci sono una risorsa preziosa:
  - solo 8 luci (in genere)
  - molte luci degradano la performance specialmente se:
    - non all'infinito (non direzionali)
    - con attenuation

Esplosioni con luci

- Si usano al massimo 3 luci.
- Politica di allocazione delle luci.
- Vettore statico di classe per sapere chi e' il proprietario di ogni luce
- Ogni esplosione se c'e' una luce disponibile se la alloca, e alla fine la spenge e la rende nuovamente disponibile alle altre luci.

A Steroid Versione 0.4

- Riorganizzazione Classi
- Display Lists
- Vertex Array

Riorganizzazione Classi

- GameSession figlia di GameSection
- Nuova classe intro
- Tutte le GlutCallback si rifanno alla variable globale che GameSection corrente
- Passare dall'intro al gioco significa cambiare la GameSection corrente
- variabili globali per tenere una gamesession e una gameintro

Migliorare l'efficenza

- Non tutti i modi per disegnare sono egualmente veloci.
- Function Call Overhead
  - glColor3f();
  - glNormalf();
  - glVertex();
  - Per specificare un triangolo 9 chiamate di funzione.

Display List

- Meccanismo per “cachare” una serie di operazioni di rendering
- Una display list e’ una sequenza di comandi opengl che viene memorizzata per poter poi essere nuovamente eseguita.
- Ogni display list e’ identificata da opengl con un intero

Display List

- Allocazione
  - alloca “n” liste consecutive richiamabili con gli interi dli..dli+n-1

```
dli=glGenLists(n);
```

- Disallocazione

```
glDeleteLists(dli,n);
```

Display List

- Generazione
  - genera la display list dli, mode puo essere
    - GL_COMPILE
    - GL_COMPILE_AND_EXECUTE

```
glNewList(dli,mode);
..Comandi opengl..
glEndList()
```

- Chiamata

```
glCallList(dli);
glCallLists(dli,n);
```

Display List

- Difetti Display List:
  - sono statiche
  - gli oggetti vengono tenuti in memoria due volte.
- Pregi
  - danno la possibilita’ ad opengl di convertire tutti i dati nel formato piu’ conveniente

Vertex Array

- In Opengl 1.1 si puo compattare tutte i dati da passare alle varie primitive opengl in un unico vettore.
  - Si deve dichiarare quali vettori si vuole usare
    - vertex
    - color
    - normal
    - texcoord
  - come sono fatti
  - e abilitarli con

```
glEnableClientState (GL_VERTEX_ARRAY);
glEnableClientState (GL_COLOR_ARRAY);
glEnableClientState (GL_NORMAL_ARRAY);
```

-

Vertex Array

- Per specificare un vettore di vertici

```
glVertexPointer(int n,TYPE,int stride,void *data);
```

  - dove **n** =2,3,4 indica quante coordinate si specifica per ogni vertice
  - **TYPE** puo essere GL_FLOAT, GL_DOUBLE, ecc
  - **stride** indica quanti byte ci sono tra un vertice e il seguente, zero significa che sono paccati strettamente e il vettore contiene solo le coordinate)
  - **data** e’ un puntatore al vettore in questione

Vertex Array

- Similarmente si specificano
  - colori
    ```
    glColorPointer(N,TYPE,stride,data);
    ```
  - normali
    ```
    glNormalPointer(TYPE,stride,data);
    ```

Vertex Array

- Vertici normali e colori possono anche essere tutti in uno stesso vettore:

```
Class Vertex {
  float v[3];
  float n[3];
  float c[3];
};
Data Vertex[n];

glVertexPointer(3,GL_FLOAT,sizeof(Vertex),&(Data.v[0]));
glColorPointer(3,GL_FLOAT,sizeof(Vertex), &(Data.c[0]));
glNormalPointer(3,GL_FLOAT,sizeof(Vertex),&(Data.n[0]));
```

Vertex Array

- Per disegnare utilizzando gli array definiti:
- `glDrawArrays(primitive,int start,int end);`
  - Dove
  - primitive e' uno dei soliti GL_TRIANGLES, GL_LINES, ecc
  - start e end dicono quali elementi del vettore usare.

Vertex Array

- Difetti
  - Possono essere meno efficienti delle display list
- Pregi
  - Non occupano memoria aggiuntiva
  - l'utente puo' cambiare la mesh continuamente.

Risorse

- Font3d
- utility che permette di costruire scritte 3d con un qualunque font TrueType e salvarle in formato raw
- disponibile per win e linux

Esercizio 1 (*)

- Star Background
- aggiungere un po' di stelle sul background della scena per renderela un po' meno piatta

Esercizio 2 (**)

- Esplosioni con Frammenti
- Ogni esplosione lancia una decina di frammenti, ogni frammento e' modellato con un singolo triangolo

Esercizio 3 (***)

- Better Intro
- Aggiungere asteroidi che vengano diretti contro l'osservatore.
- Per non appesantire troppo gli asteroidi quando sono lontani (e quindi piu' piccoli) dovrebbero essere disegnati con minor dettaglio.