Costruzione di Interfacce
Lezione 27
Estensioni Opengl, Multitexturing,

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

Estensioni Opengl

- Opengl è guidato dal consorzio Opengl ARB (Architecture Review Board) formato dalle principali ditte coinvolte nel 3d e che si accorda su come opengl deve evolvere.
- Due meccanismi principali
- Release ufficiali delle specifiche
  - 1.1, 1.2… ecc. All'incirca ongi anno o due.
  - Adesso siamo a 1.4 (quest'estate) e la maggior parte dei driver implementano 1.3.
- Estensioni

Estensioni Opengl

- Gli implementatori dei driver possono aggiungere funzionalità proprie (cubemaps, vertex shaders ecc) e non standard, sotto forma di estensioni.
- Alcune estensioni possono poi diventare standard e essere incluse nelle specifiche ufficiali (gli implementatori solo liberi di implementarle o meno)

- I nomi delle funzioni e token che fanno parte delle estensioni contengono prefissi o suffissi che identificano l'origine e la diffusione dell'estensione
  - glVertexArrayRangeNV() dove nv indica che è un'estensione proposta dall'NVidia.
  - TEXTURE_CUBE_MAP_ARB, arb indica che è una estensione approvata dall ARB

Come si usano

- Quali estensioni supporta il nostro driver?
  - glGetString(GL_EXTENSIONS);
- Come si fa a usare un token?
  - Bisogna avere un .h con le definizioni di tutti I token aggiornate alle ultime estensioni dichiarate.
- Come si fa ad usare una funzione?
  - wglGetProcAddress("nome funzione");

Opengl 1.2 e 1.3

- Purtroppo gli include e le lib inclusi con .net sono relativi a Opengl 1.1.
- Come si accede alle feature di opengl1.3 che sono ormai supportate dai driver delle schede grafiche?
- Occorrono:
  - glext.h e wglext.h
  - http://oss.sgi.com/projects/ogl-sample/registry
  - Che contengono tutti i token e I tipi delle funzioni di fino a opengl1.4 e tutte le estensioni esistenti.

Strada semplice

- Poichè è noioso e poco portabile controllare e inizializzare tutte le funzioni che servono si usa ***glew*** che fornisce un interfaccia protabile a tutto il problema (glew.sourceforge.net)
- Sostituire
  - #include <GL/gl.h>
- con
  - #include <gl/glew.h>
  - Ricordarsi di ***non*** includere mai gl.h prima di glew

GLEW

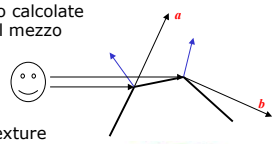
- linkare con glew32.lib e aggiungere la dll di glew nel path
- chiamare non appena si ha un contesto opengl **glewInit**
- E tutte le funzioni appaiono magicamente (quelle che ci sono…)
- Per controllare che ci sia una funzione

```
if (GL_TRUE == glewGetExtension("GL_ARB_fragment_program"))
  { /* ARB_fragment_program is supported. */ }
```

- ricordarsi di fare qualcosa anche se non ci sono le ext…

Sphere Environment Mapping

- Funziona ma ha dei limiti
- Le coordinate texture vengono calcolate solo ai vertici e interpolate nel mezzo
- Sul bordo di un oggetto puo' capitare che coordinate calcolate per I vertici di un triangolo corrispondano a punti molto distanti nella texture
- Interpolando linearmente si riattraversa erroneamente tutta la texture la soluzione corretta avrebbe dovuto interpolare in maniera non lineare



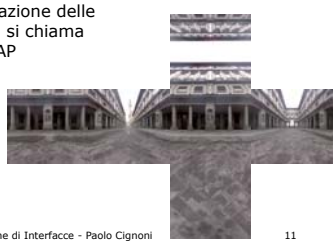
Cube Maps

- Una soluzione più corretta viene dall'uso delle cube maps
- Lo spazio intorno all'oggetto viene mappato, invece che su una sfera, sulle sei facce di un cubo
  - View independent
  - Distorsione minima
  - Robusto
  - È un estensione di opengl…

- Si indicizzano con **TRE** tex coord invece che due (il punto del cubo colpito dal vettore (s,t,r)

CubeMap

- Occorre caricare 6 texture, ognuna con un target differente, eg:
- `glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z_ARB, 0,3,w,h,0,GL_RGB,GL_UNSIGNED_BYTE,ilGetData());`
- La modalità di generazione delle texcoord è su s,t,r e si chiama GL_REFLECTION_MAP



Classe CICubeMap

- Incapsula un po' delle cose noiose riguardo alle cubemaps
  - Loading di 6 texture con un basename comune appoggiandosi a QT
  - CICubeMap.Load(uffizi.jpg) carica 6 immagini chiamate:
    - Uffizi_negx.jpg
      Uffizi_negy.jpg
      Uffizi_posx.jpg
      …

Background

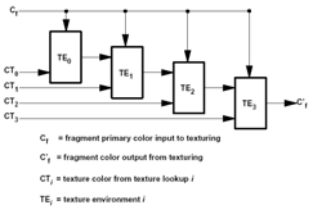
- Esiste anche un'altra modalità di generazione tex che riguarda le normal map
- glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_NORMAL_MAP_ARB);
- Assegna come tex coord di un vertice la normale del vertice interpretata come (r,s,t).
- Puo essere usato per creare uno sfondo convincente

CubeMapped Background

- piazzando un cubo abbastanza grande centrato rispetto all'osservatore (con le normali mediate ai vertici.
- In realtà non importa che il cubo sia grande, basta buttare via tutte le trasf di modellazione e disabilitare la scrittura sul depth buffer.
- Le rotazioni della camera/trackball basta metterle dentro la matrice di texture.
- CICubeMap ha una chiamata per far tutto ciò…

MultiTexture

- In opengl è possibile usare più di una texture contemporaneamente
- Tramite estensione
- Ogni Texture unit ha un proprio stato (environment, parameters, texture coords)



Multiple Texture

```
glActiveTexture(GL_TEXTURE0);
   glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE );
   glDisable(GL_TEXTURE_GEN_S);
   glDisable(GL_TEXTURE_GEN_T);
   glBindTexture(GL_TEXTURE_2D,tiBall);

glActiveTexture(GL_TEXTURE1);
   glEnable(GL_TEXTURE_CUBE_MAP);
   glBindTexture(GL_TEXTURE_CUBE_MAP,cm.ti);
   glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE );
   glEnable(GL_TEXTURE_GEN_S);
   glEnable(GL_TEXTURE_GEN_T);
   glEnable(GL_TEXTURE_GEN_R);
   glTexGeni(GL_S,GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);
   glTexGeni(GL_T,GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);
   glTexGeni(GL_R,GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);
```


MultiTexture

- Le coordinate di texture possono essere diverse per ogni texture unit
- glMultiTexCoord2f(GL_TEXTURE0, s,t)
- glMultiTexCoord2f(GL_TEXTURE1, s,t)

Ripartiamo da zero

- Copio da una vecchia versione (Moebius3ds) i seguenti file
  - main.cpp
  - CIGLWidget.h CIGLWidget.cpp
  - mainform.h mainform.cpp
  - mainform.ui mainform.ui.h
  - QTMoebius.pro
  - la dir images con le icone
- copio il .pro e la dir vcg ultima

- rinomino il pro, lo apro da designer e lo importo in .net
- Aggiungo la parte dello scene graph base senza moebius
- Sposto tutti i file che inziavano con ci in una subdir CI
- Aggiungo la trackball
- Aggiungo qualcosa da disegnare nella glwidget, il solito scenegraph base

Classe Grid e Cell

```
class CICell
{
public:
   bool selected;
   Point3f V[4];
   Point3f n;
   void ComputeNormal();
   void glDraw();

};
```


Classe Grid e Cell

```
class CIGrid
{
public:
    CIGrid() :ti(0){resize(10,10);}
    CIGrid(const int _sx, const int _sy) {resize(_sx,_sy);}
  GLuint ti;
    void setTexture(QString t) {texFileName=t;};
    void loadTexture();
    void resize(const int sx, const int sy)  ;
    inline CICell &G(const int i, const int j){return g[j*sx+i];}
    void glDraw();

private:
    std::vector<CICell> g;
    int sx,sy;
    QString texFileName;
};
```


MMSG

- Solito subclassing di CISG
- ci aggiungo il nodo Grid e (da fare come esercizio) quello CIMd2
  - Aggiungere il nodo significa solo fare una classe wrapper derivata da grid e da CINode e implementare i metodi glDraw ecc.

MMSG

```
class CIGridNode : public CIGrid, public CISGNode
{
public:
    virtual void XMLWrite(QTextStream &xstrm)
    virtual bool XMLRead(QDomElement en, CISG *Base) {return true;};
    virtual void glDraw(const float ){ CIGrid::glDraw(); };
};

// Scene Graph specializzato per tenere anche un unico Grid;
class MMSG : public CISG
{
public :
    virtual CISGNode *Allocate(const QString &classname)
    {
          CISGNode *ret=CISG::Allocate(classname);
          if(classname=="CIGrid") ret=new CIGridNode();

          return ret;
    }
 CIGridNode *g;
};
```


Aggiungo un bottone

- l'icona la disegno con il .net
- aggiungo l'azione
- trascino l'azione sulle parti dell'interfaccia
  - sulla toolbar
  - sul menu'

Aggiungere un dialogo modale

- Disegnare il dialogo da aggiungere nel designer
- includere il .h nel mainform
- istanziare un oggetto del tipo appena creato
- fare partire il dialogo quando serve con la exec.
- I valori stanno nei widget del dialogo secondo la solita filosofia qt.