Appunti di

Computer Grafica

Ing. Luca Gardelli
lgardelli@deis.unibo.it

DEIS - Dipartimento di Elettronica Informatica e Sistemistica
Università degli Studi di Bologna sede di Cesena
via Venezia 52, 47023, Cesena (FC) Italy
Ph: +39 0547 339244 Fax: +39 0547 339208

Sommario

- Introduzione
- Concetti di base: grafica 2D
- Concetti di base: video e animazioni
- Concetti di base: resa e modellazione 3D

Appunti di

Computer Grafica

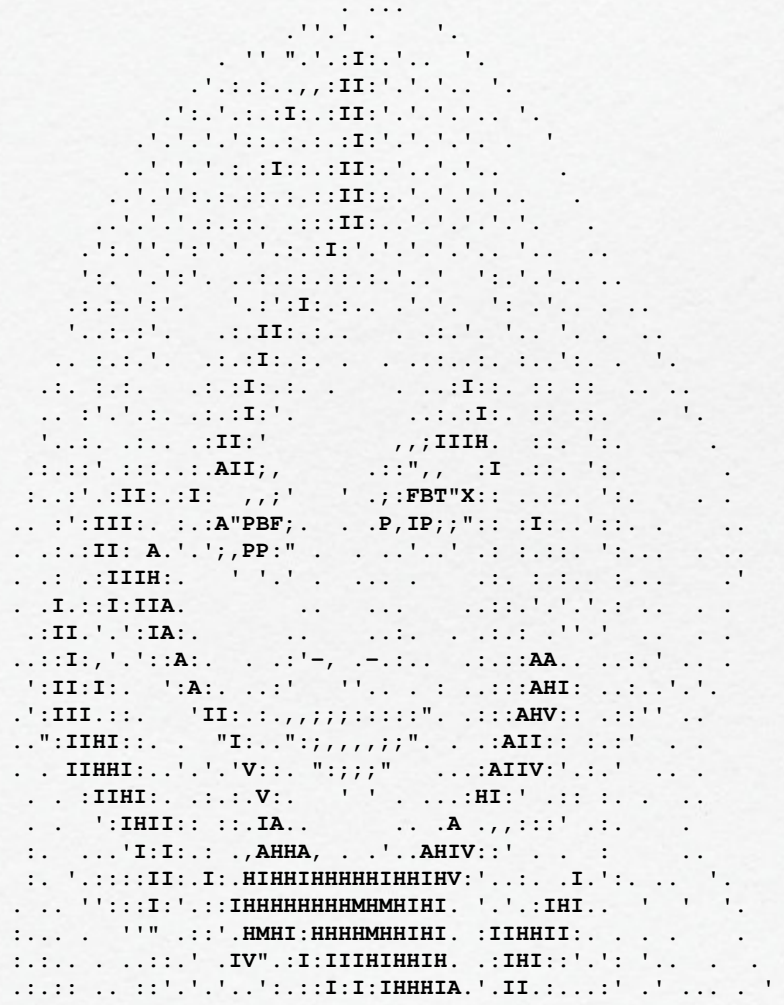
Introduzione

Computer Graphics

- La Computer Graphics (CG) è un campo molto vasto che si avvale di computer per la
  - generazione di rappresentazioni visuali dell'informazione;
  - acquisizione di informazioni visuali dal mondo reale.

CG - preistoria

- I primi tentativi di generare immagini al computer risalgono al 1965 circa
- Ancora non esistevano monitor, tastiera, mouse e i pionieri stampavano immagini formate da caratteri...



CG - storia

- La vera storia della CG si può fare coincidere con la costruzione del primo monitor con tubo catodico (monocromatici)..
- .. le prime applicazioni grafiche tardarono poco (e.g. Sketchpad l'antenato dei moderni CAD)
- I monitor CRT sono ancora utilizzati: ora sono molto più sofisticati e precisi e offrono una qualità superiore a quelli LCD.

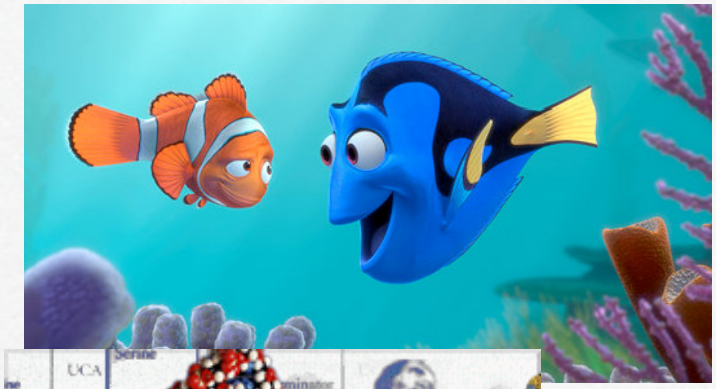
CG - background

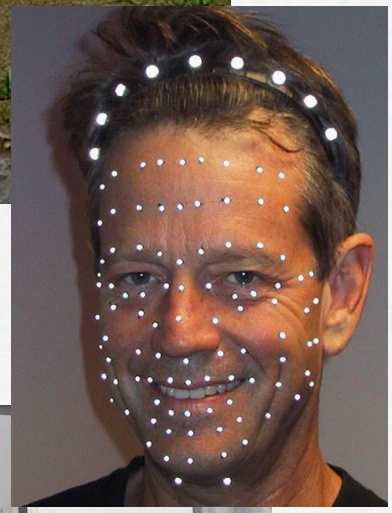
- Integra competenze provenienti da altre discipline quali
  - elettronica;
  - matematica;
  - informatica / intelligenza artificiale;
  - ricerca operativa / ottimizzazione.

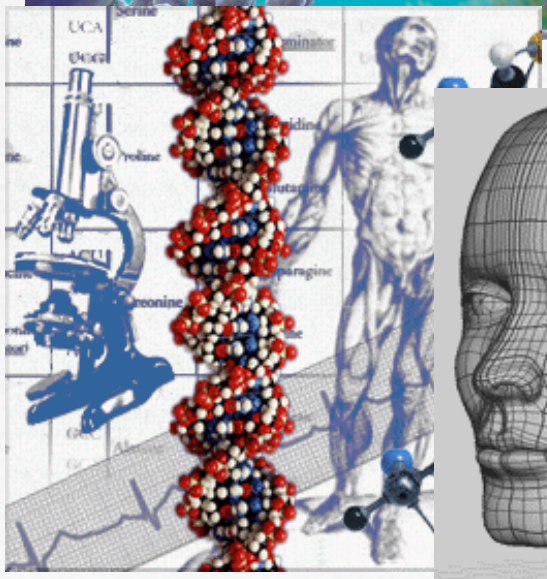
CG - applicazioni

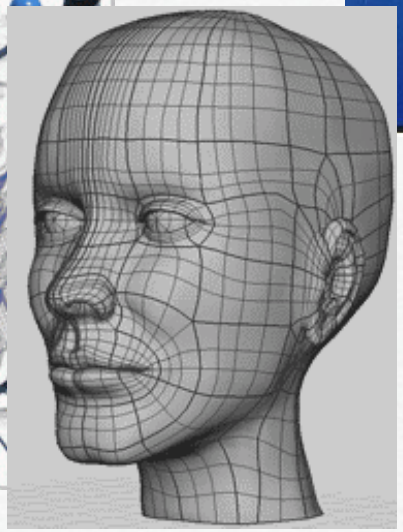
- Le applicazioni sono molteplici, ma si possono raggruppare in
  - acquisizione, modifica, sintesi di immagini 2D: pittura digitale, fotoritocco, grafica pubblicitaria, effetti speciali..
  - visualizzazione di modelli geometrici 3D
    - real-time: videogiochi, CAD, medicina, web..
    - non real-time: film, pubblicità, architettura..
  - acquisizione, modifica di video: film, web..

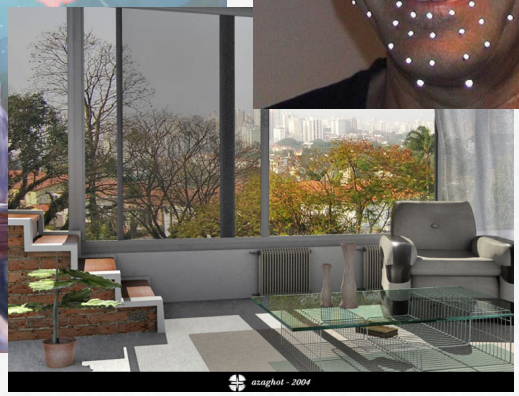
CG - applicazioni



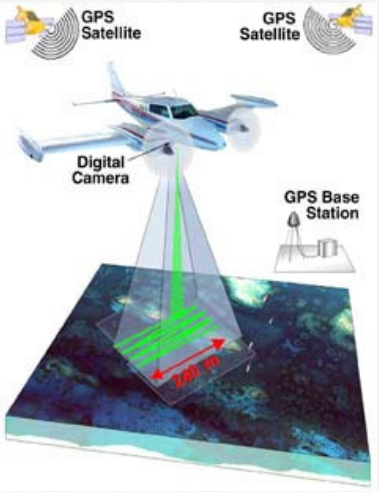








CG - hardware


GPS Satellite
GPS Satellite
Digital Camera
GPS Base Station

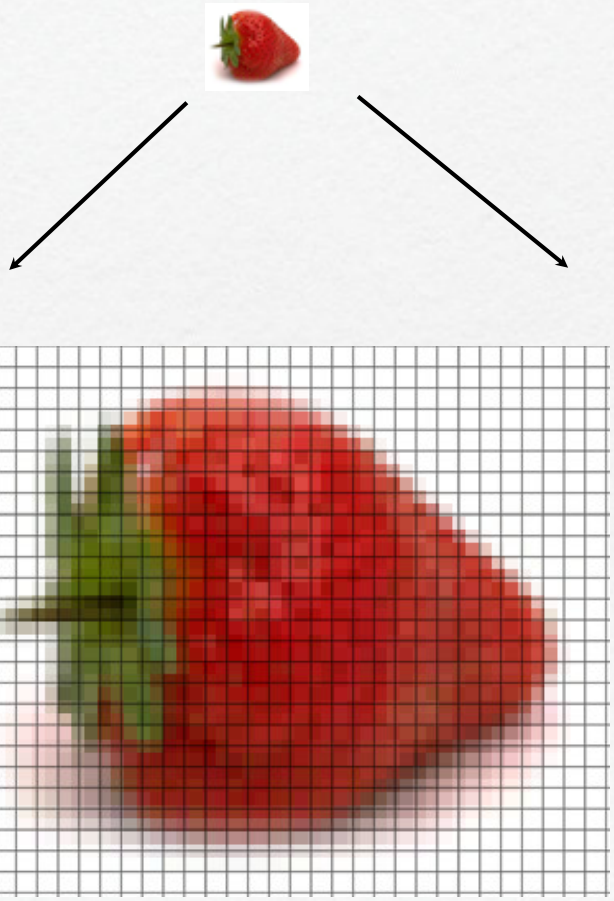
Appunti di

Computer Grafica

Concetti di base: Grafica 2D

Pixel

- I monitor attuali rappresentano le immagini come matrice di pixel (tecnologia raster)
- Il pixel è dunque l'elemento più piccolo rappresentabile: ha forma quadrata ed ha un colore uniforme



Risoluzione

- La risoluzione misura il numero di pixel per superficie del monitor: ppi - pixels per inch, oppure dpi - dots per inch
- Maggiore la risoluzione, maggiore la qualità dell'immagine rappresentata
  - carta da quotidiano 75 dpi
  - computer display 72 a 96 dpi - in aumento
  - carta da rivista 150 dpi
  - riviste grafiche 300 dpi
  - fotografia >300 dpi

Modelli di colore

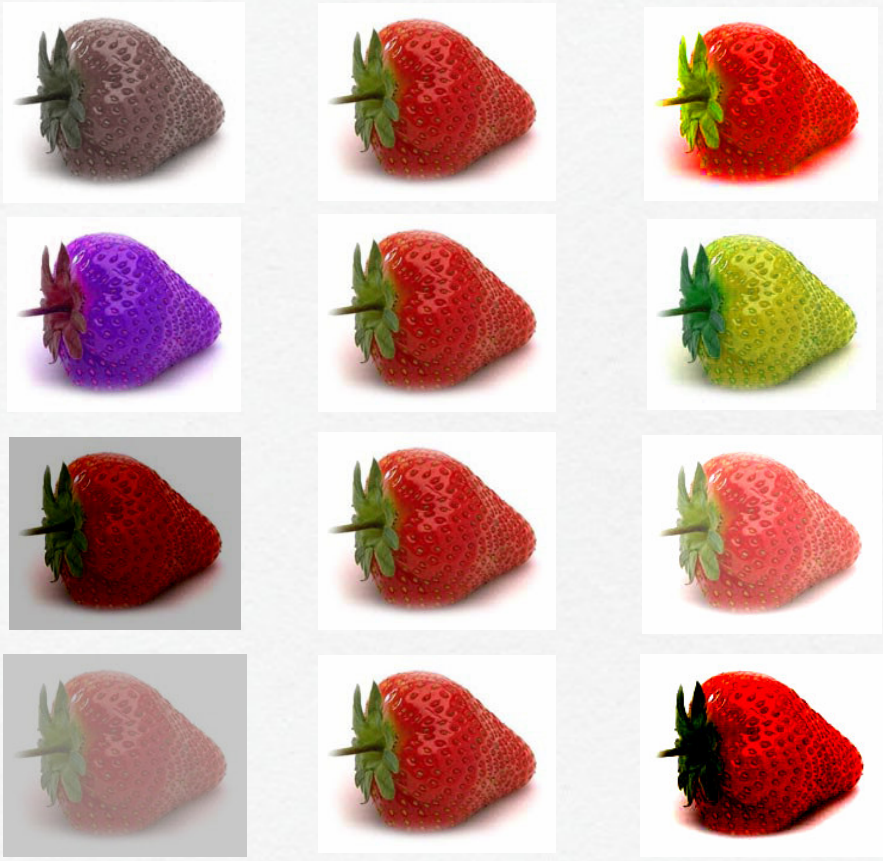
- Ad ogni pixel è associato un colore: esistono differenti modelli di colore: RGB, RYB, HSB, CMYK..
- Si caratterizzano per i colori / grandezza utilizzata
- Nel disegno si usa il modello RYB sottrattivo, Rosso Giallo, Blu (tricromia)
- Nella stampa su carta si usa il modello CMYK sottrattivo, Ciano Magenta Giallo e Nero (quadricromia)

Modelli di colore

- I monitor e le TV sono pensati in termini di RGB additivo, Rosso Verde Blu
- Le schede grafiche di ultima generazione gestiscono anche la trasparenza, RGBA A-Alpha
- Gli artisti/fotografi invece spesso ragionano per tinta, saturazione e luminosità (HSB - Hue Saturation Brightness)
- Per il web spesso si usano immagini con tabelle di colori (color indexed) piuttosto che uno spazio di colori “continuo”

Metriche

- Saturazione (saturation)
- Tinta (hue)
- Luminosità (brightness)
- Contrasto (contrast)



Aliasing

- Aliasing è quel fenomeno - spesso indesiderato - per cui i pixel sui bordi esterni di un'immagine diventano "visibili"
- Anti-Aliasing è un meccanismo che riduce tale effetto propagando il colore ai pixel vicini - una sorta di "sfumatura"

Profondità di colore

- La profondità (color depth) indica il numero bit utilizzati per rappresentare il colore
- Valori tipici sono: 1 monocromatico, 2 quadricromia, 8, 16, 24 true color, 32 (alpha), 48

Canali e formati colore

- L’informazione del colore è memorizzata su tre canali distinti, Rosso Verde e Blu + eventuale trasparenza
- Un formato di colore (color format) si caratterizza per il numero di bit associati ad ogni canale
- Ad esempio RGB 24 bit (8,8,8) mentre RGB 16 bit (6,5,5), (5,6,5), (5,5,6)...

Memorizzazione

- Le immagini vengono memorizzate in differenti formati (compressi o meno) in base all'uso
- Immagini ad alta qualità vengono compresse in modo tale da minimizzare la perdita di informazione a discapito delle dimensioni
- Per la distribuzione su web e altri media si tollerano anche le perdite..
- Formati tipici sono JPEG, GIF, PNG, TIFF, TGA, BMP

Appunti di

Computer Grafica

Video e animazioni

Il frame

- Nei film/animazioni l'illusione del movimento è data dalla presentazione di immagini in sequenza
- Tali immagini prendono il nome di frame o fotogrammi

Double buffering

- Il frame buffer è una zona di memoria che rappresenta direttamente i pixel visualizzati
- Se le operazioni di disegno sono “complicate” si verifica un fenomeno di sfarfallio – flickering
- Il double buffering è una tecnica che impiega due buffer – di cui uno nascosto – per ridurre il flickering: le operazioni di disegno vengono eseguite su quello nascosto e poi copiate direttamente sul frame buffer

Frame rate

- La velocità con cui si presentano i frame prende il nome di frame rate e si misura in fps - frames per second
- Di solito per movimenti fluidi si ritengono necessari almeno 24 fps
- Alcune applicazioni (per motivi di prestazioni) usano rate minori, e.g. videotelefono, webcam, video streaming
- Gli standard attuali prevedono: cinema 24 fps, tv americana 30 fps (NTSC), tv europea 25 fps (PAL)..

Video Bit Rate

- Un parametro per valutare la qualità di un video è la sua bit rate, cioè il flusso di bit per secondo (si indica con bps)
- Ad esempio 10 secondi di animazione a 2000 Kbps richiedono circa 2,5 MB
- A parità di bit rate la qualità dipende dall'efficienza dell'algoritmo utilizzato..

Distribuzione CD/DVD

- La distribuzione di contenuti video tramite CD e DVD rende disponibile una capacità di memorizzazione alta e quindi permette di avere una VBR alta
- Formati comuni sono l'MPEG nelle varie versioni (DVD - MPEG2), AVI - Audio Video Interleave, QuickTime, DIVX..

Distribuzione web

- La distribuzione di contenuti video sul web richiede un'attenzione particolare per quanto riguarda i limiti di banda
- Si utilizza l'approccio streaming, cioè si inviano le informazioni durante la riproduzione del video
- Supportano lo streaming QuickTime, Real Media, e Window Media Video

Grafica vettoriale

- I vincoli di banda che impone il web hanno portato allo sviluppo di altre tecniche di animazione “più leggere”.
- Nella grafica vettoriale i frame non sono definiti da immagini, ma da un insieme di primitive di disegno - linee.. - le cui caratteristiche possono variare nel tempo - colore, forma, posizione.
- Per ulteriori dettagli si rimanda a Macromedia Flash e lo standard SVG del W3C

Appunti di

Computer Grafica

Modellazione e Resa 3D

Panoramica

- La grafica 3D si suddivide tipicamente in due sotto-aree
  - interattiva (real-time): legata alla visualizzazione di geometrie 3D in tempo reale - e.g. CAD, videogiochi..
  - non-interattiva: prodotta con strumenti interattivi, ma distribuita in modo tale che non è possibile l’interazione - e.g. film, pubblicità..

Panoramica

- Il campo della grafica 3D è estremamente vasto e richiede competenze delle arti tradizionali..
  - Disegno, pittura, teoria del colore, anatomia artistica..
  - Animazione e figura in movimento..
  - Scultura, tecniche dei materiali..
  - Fotografia, cinema, regia..
  - Illuminazione, dinamica e resa dei materiali..
  - ..

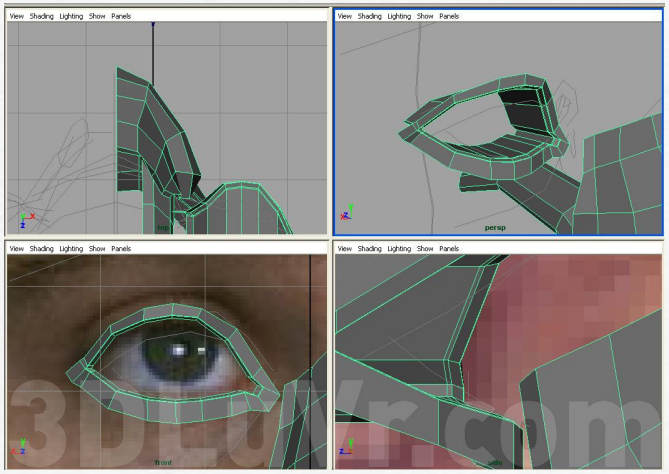
Panoramica

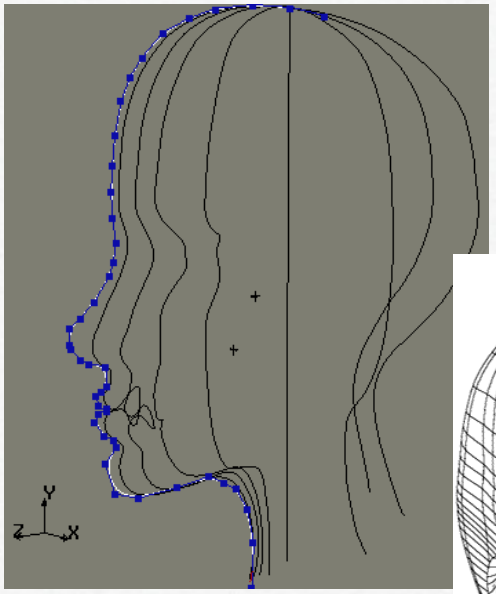
- Un “prodotto” di grafica 3D passa attraverso le seguenti fasi artistiche
  - Modeling
  - Texturing | Rigging | Skinning
  - Animation
  - Lighting
  - Rendering
  - Compositing & Post-production

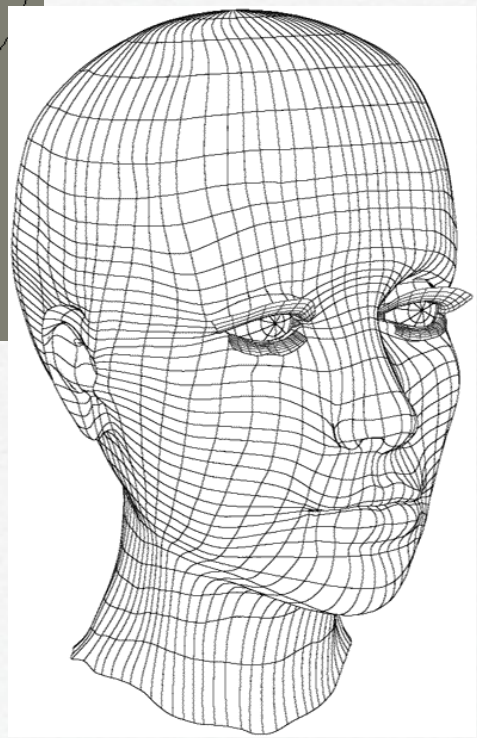
Modeling

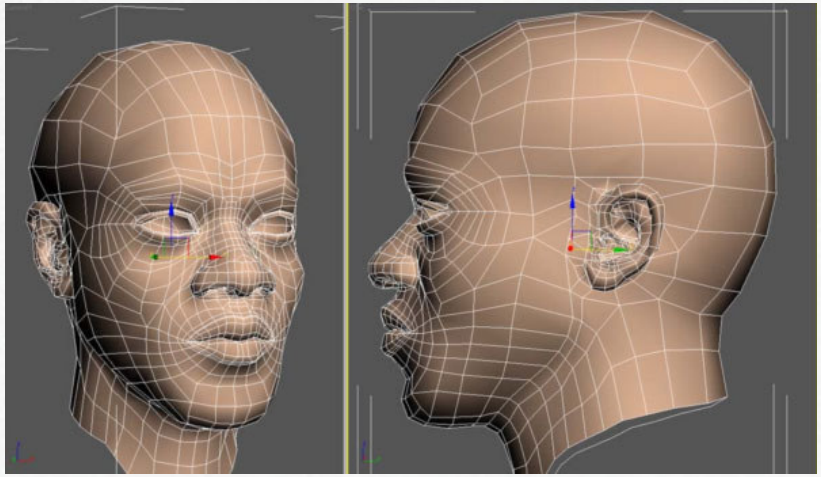
- E’ la fase in cui si crea una geometria 3D di un oggetto da rappresentare
- Il modello prodotto è un insieme di vertici, segmenti e triangoli oppure di curve e superfici parametriche
- Si può partire da disegni, sculture oppure direttamente dalle primitive geometriche, ma i risultati sono più “artificiali”

Modeling





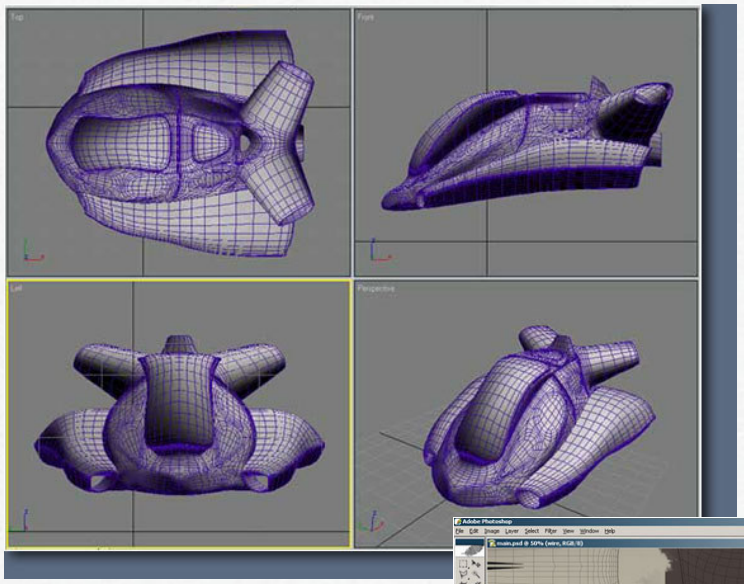


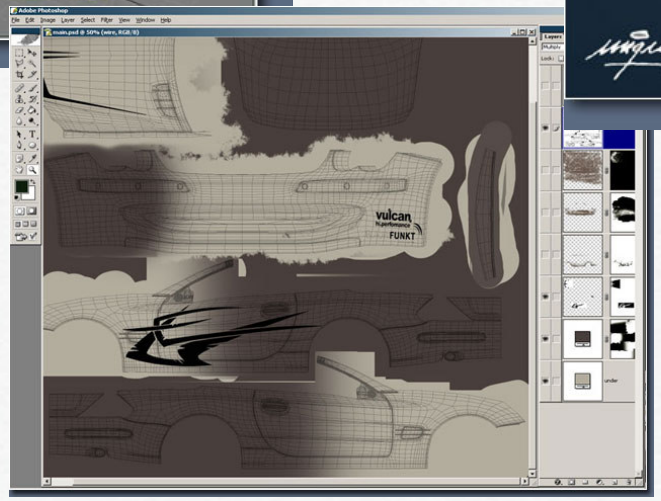


Texturing

- In questa fase le geometrie vengono “colorate” con texture, cioè immagini che rappresentano dei materiali
- Le texture possono essere ottenute da foto, disegni, oppure in modo parametrico -e.g. vetro, marmo..
- Occorre tradurre le coordinate della geometria e le coordinate delle texture (UV Mapping)

Texturing

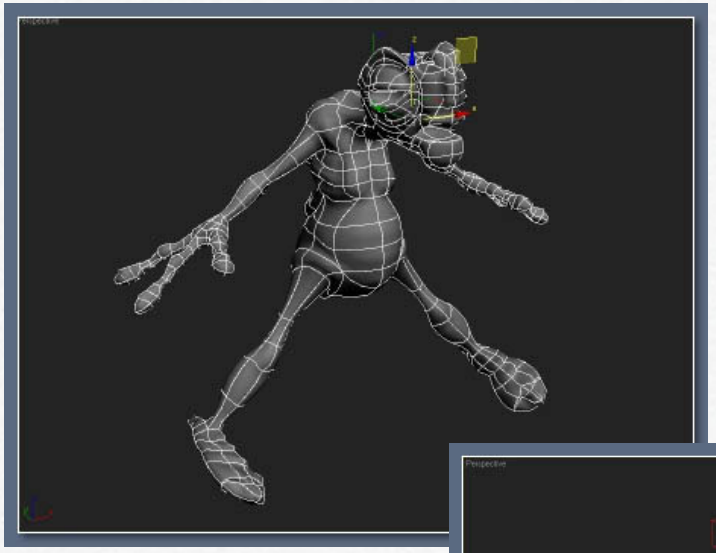


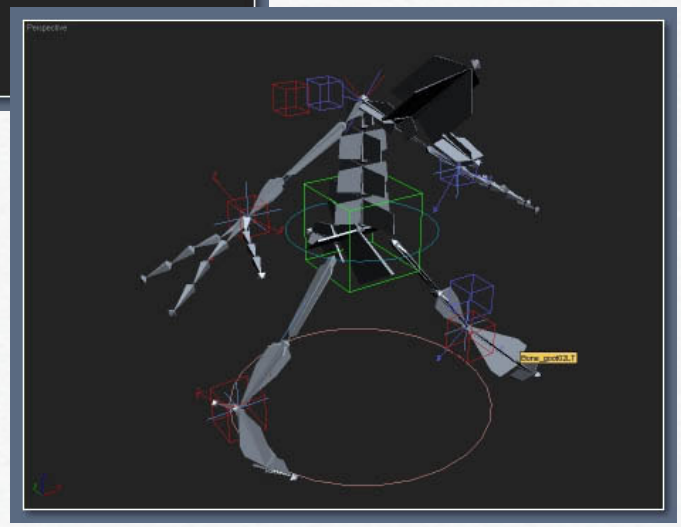


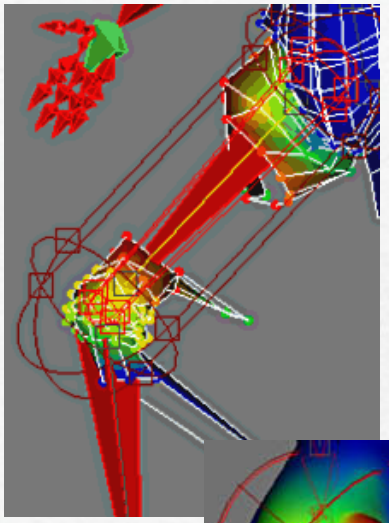
Rigging e Skinning

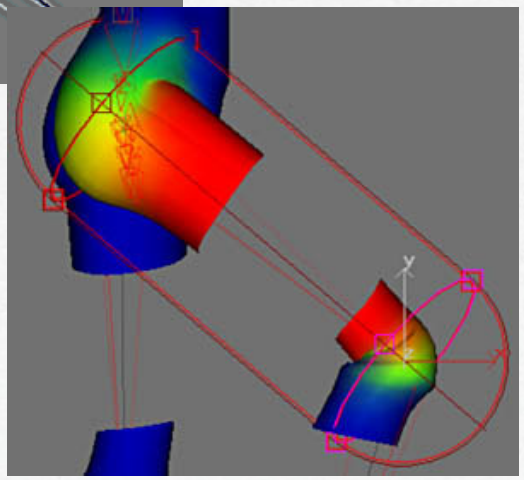
- Il rigging è la fase in cui si creano le ossature di personaggi e veicoli
- In questo modo è possibile deformare le geometrie in un modo più vicino alla realtà
- Lo skinning è quella fase in cui si parametrizza l'effetto del rigging sulla geometria - e.g. muscoli..

Rigging e Skinning









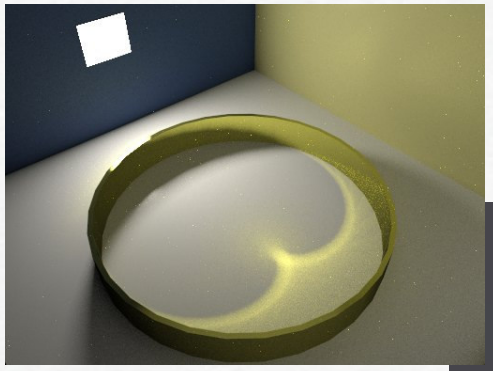
Animation

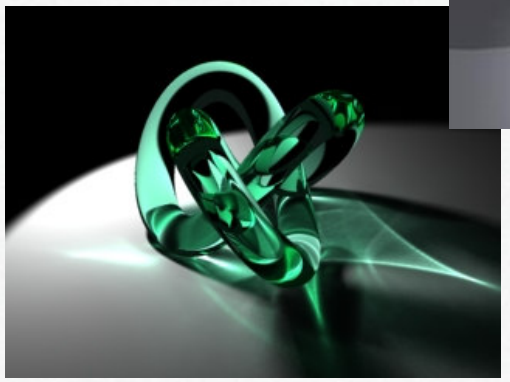
- In questa fase si definiscono i movimenti dei personaggi, dell'ambiente e le inquadrature di massima
- Utilizzando le bones - in realtà è molto più complesso - si definiscono le trasformazioni chiave della geometria
- Queste trasformazioni vengono interpolate per produrre il movimento!
- Si guadagna molto rispetto all'animazione tradizionale in cui va disegnato ogni frame!

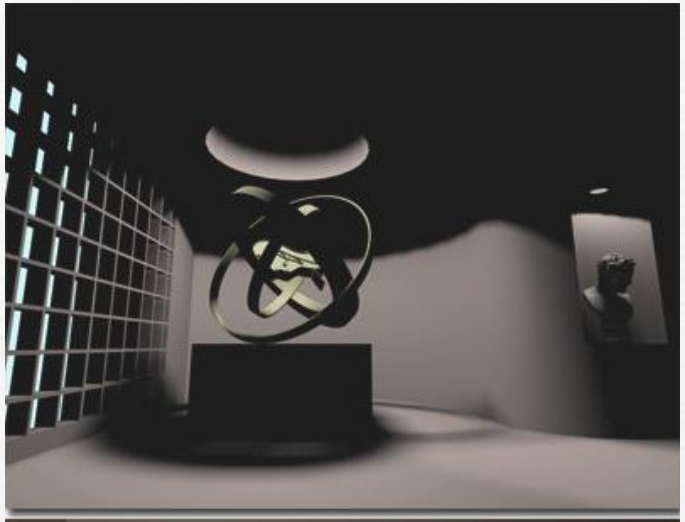
Lighting

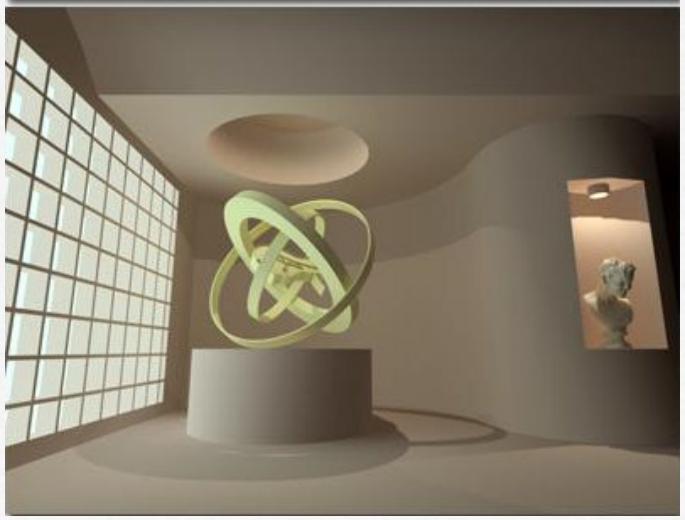
- Si definisce il tipo di illuminazione della scena prodotta da luci e ambiente
- Numero di luci, posizione, tipo, colore, ombre..
- Grado di realismo, ray- tracing, radiosity, global illumination

Lighting









Rendering

- Questa fase non è interattiva!
- Una volta decisi i parametri - qualità, formato.. - si avvia un processo che elabora le informazioni risultanti dalle fasi precedenti e producendo i filmati / le immagini “finali”.
- Un singolo fotogramma può richiedere anche parecchie ore se si vuole raggiungere il foto-realismo. (180h per il mammut dell’Era glaciale!)
- E’ più legata alla tecnologia e la distribuzione del calcolo che a aspetti artistici!

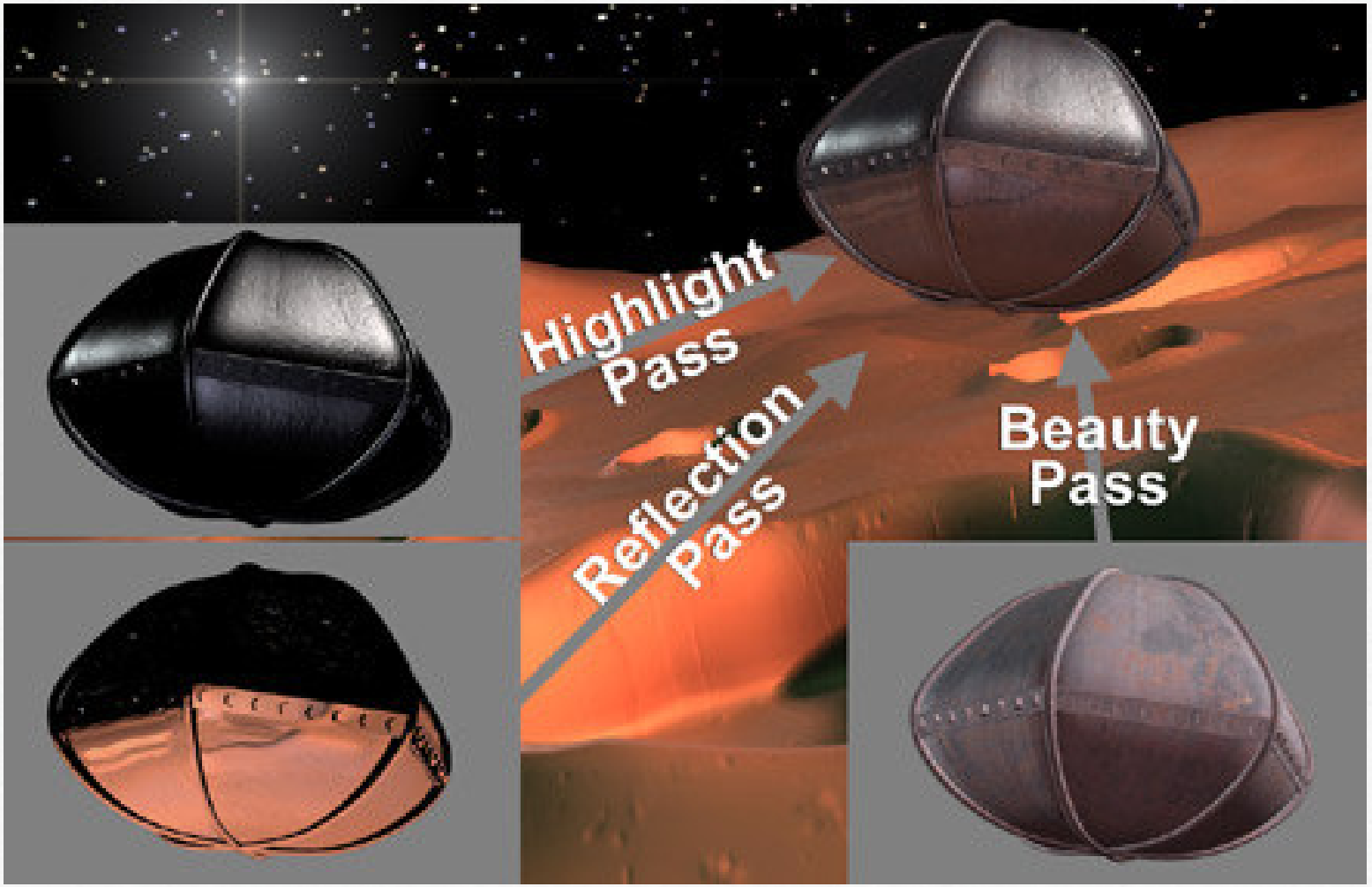
Rendering Shrek 2

- *The data center that resulted consisted of 500 HP servers (1,000 processors) connected to DreamWorks' studio 20 miles away in Redwood City, Calif., via a secure fiber optic link.*
- Ad una media di 10 h/CPU per frame Shrek 2 richiederebbe circa 69000 gg/CPU per essere renderizzato!

Compositing & Post-Production

- In questa fase le immagini vengono montate per formare un unica sequenza
- Spesso il rendering produce diverse immagini per un singolo frame -multipass
- Queste immagini vengono sovrapposte per regolare la luminosità, riflessioni, ombre..
- Inoltre si scelgono quali sequenze mantenere, le inquadrature - regia
- Si integrano audio e sottotitoli, titoli e titoli di coda..

Compositing & Post-Production



Appunti di

Computer Grafica

QTJava

QuickTime for Java

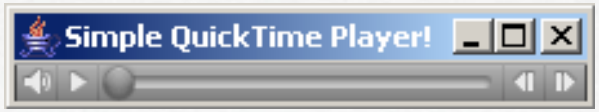
- E’ una libreria sviluppata da Apple che supporta lo sviluppo di applicazioni multimediali www.apple.com
- Le funzionalità principali sono
  - Riproduzione (anche streaming) di file audio, midi e video in diversi formati
  - Conversione audio/video in diversi formati
  - Video editing
  - Registrazione di file audio
  - Virtual Reality..

Per iniziare...

- Per poterne fruire è necessario avere installato QuickTime includendo anche la libreria per Java, più gli strumenti di sviluppo di base (JDK, eclipse..)
- QuickTime al momento dell'installazione copia il file QTJava.zip nel cartella /Windows/System32: tale file va incluso nel progetto come un Jar
- In più occorre scaricare la documentazione delle API – che contiene anche numerosi esempi

Simple Quicktime Player

- Al fine di prendere confidenza con la libreria consideriamo un'applicazione
- Simple Quicktime Player è un esempio di applicazione realizzata a partire da alcuni esempi forniti con la documentazione
- Lo scopo è quello di riprodurre filmati e brani audio



Simple Quicktime Player

- Ogni applicazione che fa uso della libreria deve aprire una sessione
- Nel caso in cui vengano generate delle eccezioni a run-time la sessione deve essere chiusa
- Al termine dell'applicazione la sessione va chiusa
- Il resto dell'applicazione è dentro il metodo `createNewMovieFromURL()`

```
try {
 //Apre la sessione
QTSession.open();
createNewMovieFromURL
("file://C:/1.mp3");

} catch (QTException
e) {
 //Chiude la sessione
e.printStackTrace();
QTSession.close ();
}
```

Simple Quicktime Player

- Si crea un oggetto che contiene le info del file da riprodurre
- Si crea – da tali info – un oggetto che rappresenta il file da riprodurre
- Si crea il controller con cui si agisce sulla riproduzione (play, pause..)
- Si deriva un componente e lo si aggiunge al Frame

```
public void createNewMovieFromURL
(String theURL){
try {
 DataRef urlMovie = new
DataRef(theURL);
m = Movie.fromDataRef
(urlMovie,StdQTConstants.newMovieAc
tive);
mc = new MovieController (m);
 if (qtc == null){
 qtc =
QTFactory.makeQTComponent(mc);
 add((Component)qtc);
} else {
 qtc.setMovieController(mc);
 }
} catch (QTException err) {}
}
```

Appunti di

Computer Grafica

OpenGL e JOGL

OpenGL

- OpenGL è un API per lo sviluppo di applicazioni grafiche 2D e 3D portabili www.opengl.org
- Dalla sua nascita – 1992 – OpenGL è lo standard più supportato dalle industrie – e.g. Videogiochi, CAD/CAM..
- L'architettura di OpenGL è progettata in modo da essere efficiente ed indipendente dal sistema operativo (portabilità)...
- ..per questo non include nessun comando per la creazione di finestre.

GLU – OpenGL Utility Library

- OpenGL Utility Library (GLU) è un libreria che utilizzando le OpenGL, fornisce delle funzionalità di più alto livello.
- Questa libreria include
  - Operazioni su matrici
  - Primitive di disegno

GLX – OpenGL Extension

- GLX è una libreria che estende le capacità dei sistemi a finestre per fornire supporto alle OpenGL.
- Esiste una versione di tale libreria per i sistema operativi più diffusi..
  - Microsoft Windows : wgl
  - Unix & Co. : glx
  - Apple : agl
  - IBM OS/2 : pgl

Sintassi dei comandi OpenGL

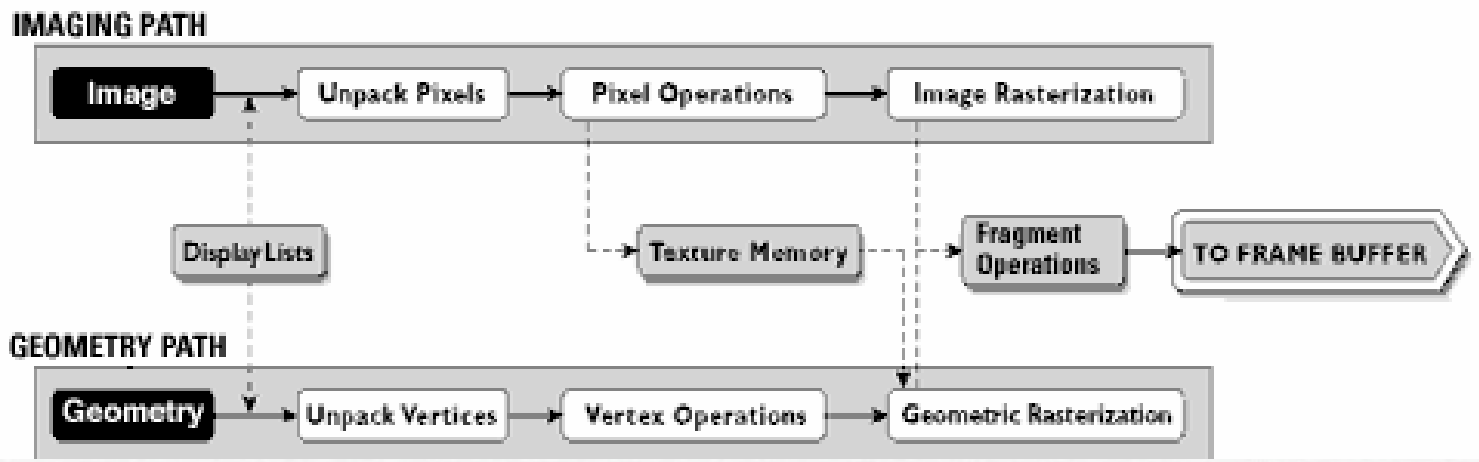
- Ogni comando inizia con *gl* e ogni lettera iniziale è maiuscola, e.g. glClearColor()
- Le costanti iniziano con *GL*, e.g. GL_BUFFER_COLOR_BIT
- Alcuni comandi indicano anche il numero di parametri ed il tipo, e.g. glVertex3f() si aspetta 3 parametri di tipo float.
- Una *v* finale indica che è richiesto un puntatore a vettore.

Sintassi e Tipi

Suffisso	Tipo	Tipo OpenGL	Equivalente C
b	8-bit intero	GLbyte	signed char
s	16-bit intero	GLshort	short
i	32-bit intero	GLint, GLsizei	int, long
f	32-bit virgola mobile	GLfloat, GLclampf	float
d	64-bit virgola mobile	GLdouble, GLclampd	double
ub	8-bit intero senza segno	GLubyte, GLboolean	unsigned char
us	8-bit intero senza segno	GLushort	unsigned short
ui	8-bit intero senza segno	GLuint, GLenum, GLbitfield	unsigned int, unsigned long
		GLvoid	

OpenGL pipeline

- OpenGL opera sia su immagini che su geometria, componendo i risultati per produrre l'immagine finale



Display list

- Sia che i dati siano pixel o geometria
  - possono essere elaborati subito : *immediate mode;*
  - possono essere memorizzati nella display list: *retained mode*.
- Quando è eseguita la display list, i dati vengono inviati come se fossero stati spediti dall'applicazione

Evaluators

- Tutte le primitive geometriche sono descritte da vertici.
- Le curve e le superfici parametriche possono essere descritte in altri modi (e.g. punti di controllo).
- Gli *evaluators* traducono questa descrizione in un insieme di vertici.

Per-vertex operations

- I vertici vengono convertiti in primitive.
- Alcune operazioni che vengono eseguite in questo stadio sono
  - Trasformazione delle coordinate dalla matrice di visualizzazione
  - Generazione delle coordinate delle texture
  - Calcolo dell'illuminazione

Glossario: clipping

- Clipping è il processo di eliminazione di porzioni di geometria che giace nel piano opposto a quello di vista.
  - Il clipping di vertici consiste semplicemente nel non disegnarli
  - Il clipping di linee può richiedere la suddivisione in ulteriori linee

Primitive assembly

- Questo stadio si comporta in modo differente in base alla primitiva (e.g. il clipping)
- Altre elaborazioni riguardano
  - Calcolo della profondità
  - Calcolo dell’antialiasing

Pixel Operations

- I pixel vengono tradotti nel formato più appropriato.
- I dati sono elaborati (e.g. scalatura) in base ad una mappa di pixel.

Texture assembly

- Questa fase vengono applicate le texture alla geometria per renderla più realistica.
- Alcuni dispositivi potrebbero avere capacità di elaborazione delle texture per ottimizzare le prestazioni.
- Le texture possono essere immagazzinati in una zona di memoria dedicata oppure in memoria centrale (meccanismi di priorità).

Rasterization

- E’ il processo che converte dati geometrici e pixel in frammenti.
- Ogni frammento corrisponde ad un pixel nel framebuffer.

Fragment operations

- Prima di scrivere i frammenti nel framebuffer vengono eseguite alcune operazioni (che possono essere disabilitate).
- Queste operazioni includono
  - Calcolo dei texel
  - Calcolo della nebbia
  - Antialiasing
  - Test di profondità, trasparenza...

Da OpenGL a JOGL..

- JOGL si inserisce in un contesto di tecnologie open source creato dal Game Technology Group di Sun MicroSystems. https://jogl.dev.java.net/
- JOGL è stato sviluppato con l'obiettivo di fornire alle applicazioni scritte in java, l'accesso all'accelerazione grafica hardware.
- Il nucleo di JOGL consiste in un "binding" delle funzioni di OpenGL tramite Java Native Interface
- In JOGL sono comprese – oltre ad OpenGL – anche le librerie GLU e GLUT, nonchè tutte le estensioni dei produttori di schede grafiche

JOGL – per iniziare..

- Per utilizzare JOGL occorre scaricare dal sito web ufficiale
  - la libreria jogl.jar da includere nei progetti
  - il binding per il sistema operativo utilizzato – supportati Windows, Solaris, Linux e MacOSX
  - le librerie native vanno inserite nella cartella di sistema appropriata – e.g. /windows/system32
- Si consiglia anche di fare riferimento anche a
  - Demo di jogl
  - NeHe OpenGL tutorials http://nehe.gamedev.net
  - Codice JOGL dei NeHe tutorials http://pepijn.fab4.be/nehe/

JOGL: Framework di lavoro

- Ogni programma – non banale – che utilizza JOGL dovrà almeno svolgere due compiti
  - Creare una finestra di output compatibile con OpenGL
  - Gestire gli eventi di base di OpenGL, cioè in pratica creare una classe che implementi l'interfaccia GLEventListener

JOGL: Framework di lavoro

- JOGL si integra perfettamente con Swing e AWT quindi si potranno utilizzare le classi Frame e JFrame!
- Avendo un frame i passi successivi sono
  - ottenere un GLCanvas (o GLJPanel)
  - impostare il GLEventListener
  - creare un Animator – gestisce un thread separato
  - gestire la chiusura della finestra

Esempio 1: Framework di lavoro

```
package demos.lgardelli.lesson001;

import net.java.games.jogl.*;
import java.awt.Frame;
import java.awt.event.*;

public class Lesson001 {

 private static Animator animator = null;
 public static void main(String[] args) {

        Frame frame = new Frame("JOGL : Lesson 1");
        GLCanvas canvas =
                GLDrawableFactory.getFactory().createGLCanvas(new
GLCapabilities());
        canvas.addGLEventListener(new Renderer());
        frame.add(canvas);
        frame.setSize(800, 600);
        animator = new Animator(canvas);
```

Esempio 1: Framework di lavoro

```
...
        frame.addWindowListener(
            new WindowAdapter(){
                public void windowClosing(WindowEvent
    e){
                    animator.stop();
                    System.exit(0);
                }
            }
        );
        frame.setVisible(true);
        animator.start();
        canvas.requestFocus();
    }

}
```

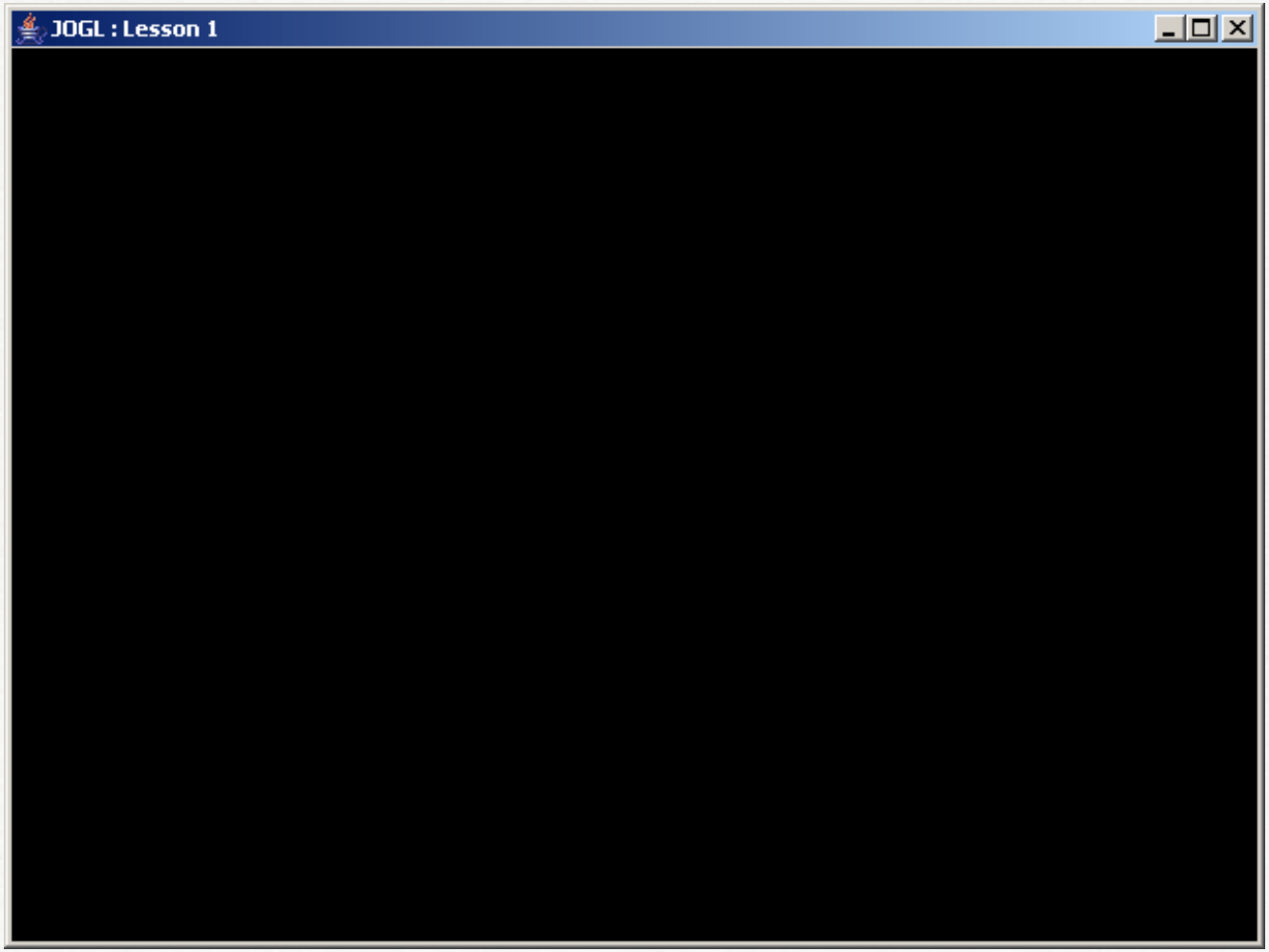
Esempio 1: Framework di lavoro

- Nel secondo passo bisogna creare la classe che implementa GLEventListener
  - Init: tutte le operazioni necessarie all'inizializzazione di OpenGL – in linea di principio dovrebbe essere chiamata una volta sola
  - Display: tutte le operazioni che devono essere ripetute per ogni frame
  - Reshape: tutte le operazioni per gestire il ridimensionamento della finestra
  - DisplayChanged: tutte le operazioni necessarie al passaggio da una modalità video all'altra – non necessaria

Esempio 1: Framework di lavoro

```
package demos.lgardelli.lesson001;
import net.java.games.jogl.*;

public class Renderer implements GLEventListener{

 public void init(GLDrawable drawable){
        final GL gl = drawable.getGL();
        gl.glClearColor(0.0f,0.0f,0.0f,0.0f);
        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
 }

 public void display(GLDrawable drawable){}

 public void reshape(GLDrawable drawable, int x, int y, int width,int height){}

 public void displayChanged(GLDrawable drawable, boolean modeChanged,
        boolean deviceChanged){}
}
```

Esempio 1: Framework di lavoro



Esempio 2: Vertici e Colori

1. Inizializzazione
2. Rotazioni e traslazioni
3. Creazione di un cubo colorato
4. Operazioni di ridimensionamento della finestra
5. Impostare una vista prospettica

Esempio 2: init

- // ottenere un'istanza della libreria
- final GL gl = drawable.getGL();
- // si definisce la modalità colorazione:
- // gradienti tra i colori dei vertici
- gl.glShadeModel(GL.GL_SMOOTH);
- //si definisce il colore con cui riempire lo sfondo
- gl.glClearColor(0.0f,0.0f,0.0f,0.0f);
- //si inizializza il Depth Buffer
- gl.glClearDepth(1.0f);
- //si abilita il test di profondità
- gl.glEnable(GL.GL_DEPTH_TEST);

Esempio 2: reshape

- //ottiene un'istanza della libreria OpenGL
- **final GL gl = drawable.getGL();**
- //ottiene un'istanza della libreria GLU
- **final GLU glu = drawable.getGLU();**
- //calcola l'aspetto della finestra
- **if (height <= 0) height = 1;**
- **final float h = (float)width / (float)height;**
- //imposta la finestra
- **gl.glViewport(0, 0, width, height);**
- //definisce i parametri della visuale prospettica
- **gl.glMatrixMode(GL.GL_PROJECTION);**
- **gl.glLoadIdentity();**
- **glu.gluPerspective(45.0f, h, 1.0, 20.0);**
- **gl.glMatrixMode(GL.GL_MODELVIEW);**
- **gl.glLoadIdentity();**

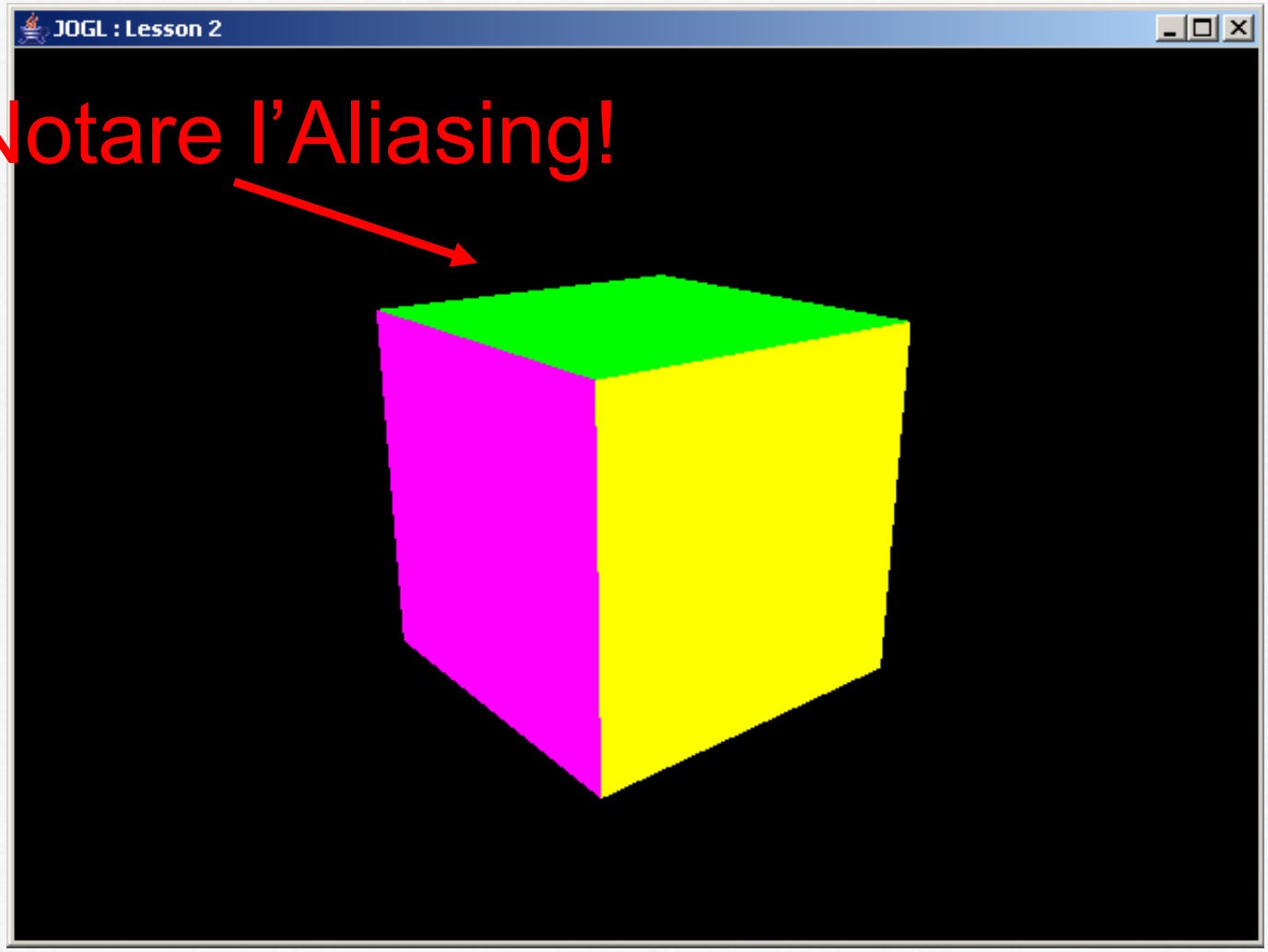
Esempio 2: display

- //ottiene un'istanza della libreria OpenGL
- final GL gl = drawable.getGL();
- //riempie lo sfondo e pulisce il depth buffer
- gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
- //carica la matrice identità
- gl.glLoadIdentity();
- //trasla e ruota il riferimento delle coordinate
- gl.glTranslatef(0.0f, 0.0f, -6.0f);
- gl.glRotatef(20, 1.0f, 0.0f, 0.0f);
- gl.glRotatef(rAngle, 0.0f, 1.0f, 0.0f);
- //inizia a disegnare quadrilateri
- gl.glBegin(GL.GL_QUADS);
- ...

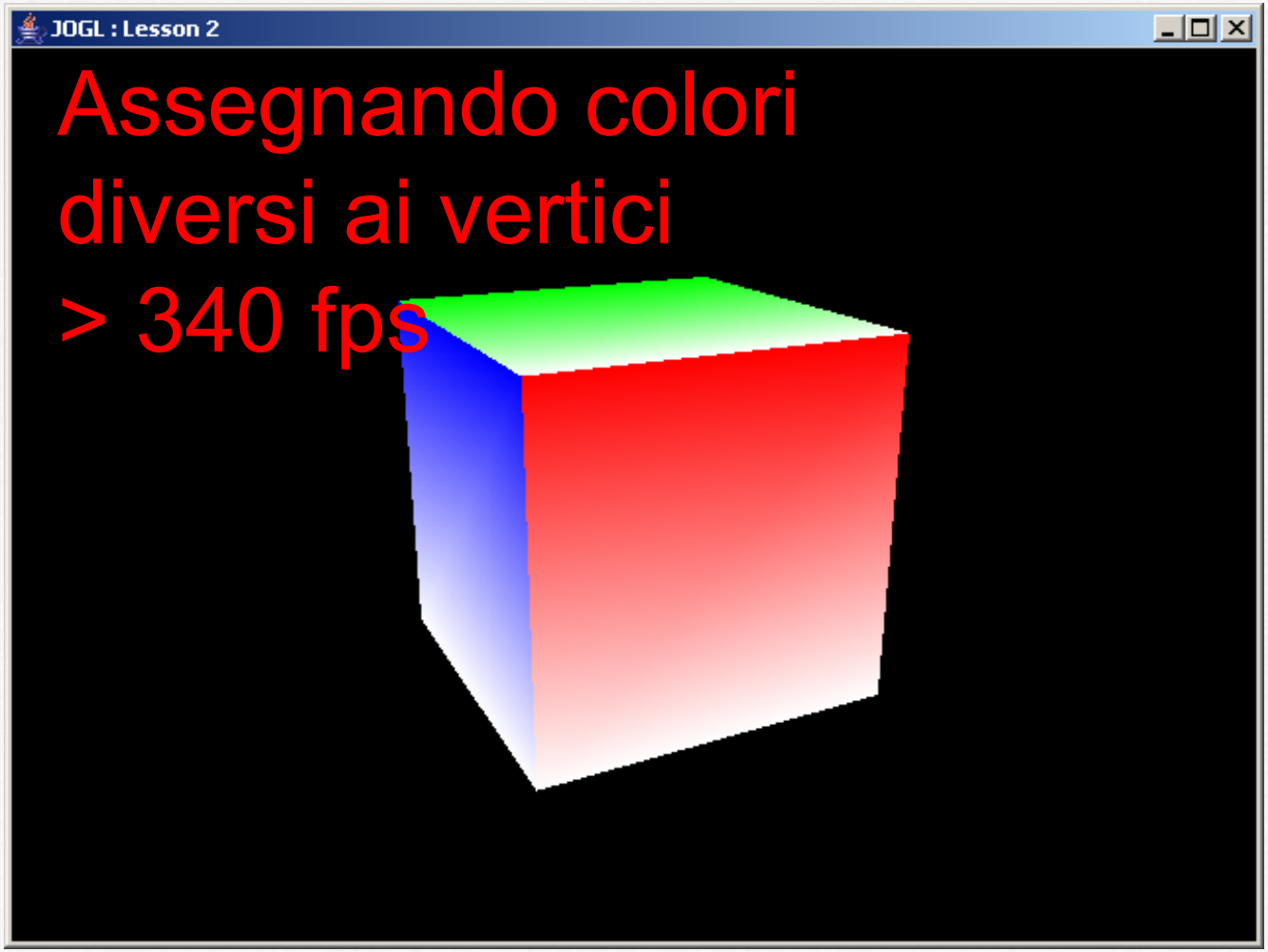
Esempio 2: display

- //definisce il colore deo vertici e definisce i vertici del primo quadrilatero
- gl.glColor3f(0.0f,1.0f,0.0f);//Verde
- gl.glVertex3f(1.0f, 1.0f,-1.0f);
- gl.glVertex3f(-1.0f, 1.0f,-1.0f);
- gl.glVertex3f(-1.0f, 1.0f, 1.0f);
- gl.glVertex3f(1.0f, 1.0f, 1.0f);
- ...
- //termina di disegnare
- gl.glEnd();
- //forza i cambiamenti
- gl.glFlush();
- //aggiorna l'angolo di rotazione
- rAngle += 0.1f;

Esempio 2: risultato



Esempio 2: risultato



Esempio 3: Texturing

- Si vuole creare un cubo che ruota a cui si vuole applicare una texture

1. Inizializzazione
2. Creazione della texture
3. Definizione delle coordinate di texturing

Esempio 3: init

- //occorre abilitare le texture!
- gl.glEnable(GL.GL_TEXTURE_2D);
- //si crea un'immagine: questo non è opengl!
- BufferedImage img = readPNGImage("res/texture.png");
- //incapsula un po' di complessità per la generazione delle texture
- //in questo momento non ci interessano i dettagli
- makeRGBTexture(gl, drawable.getGLU(), img, GL.GL_TEXTURE_2D, false);

- //si definiscono i parametri di filtraggio della texture
- gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MIN_FILTER,GL.GL_LINEAR);

Esempio 3: display

- //si lega la texture ai vertici
- gl.glBindTexture(GL.GL_TEXTURE_2D, texture);
- //inizio a disegnare
- gl.glBegin(GL.GL_QUADS);
- //per ogni vertice definisco le coordinate uv nello spazio
- //della texture sia le coordinate xyz
- gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f(-1.0f, -1.0f, 1.0f);
- gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f(1.0f, -1.0f, 1.0f);
- gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f(1.0f, 1.0f, 1.0f);
- gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f(-1.0f, 1.0f, 1.0f);
- ...

Esempio 3: risultato

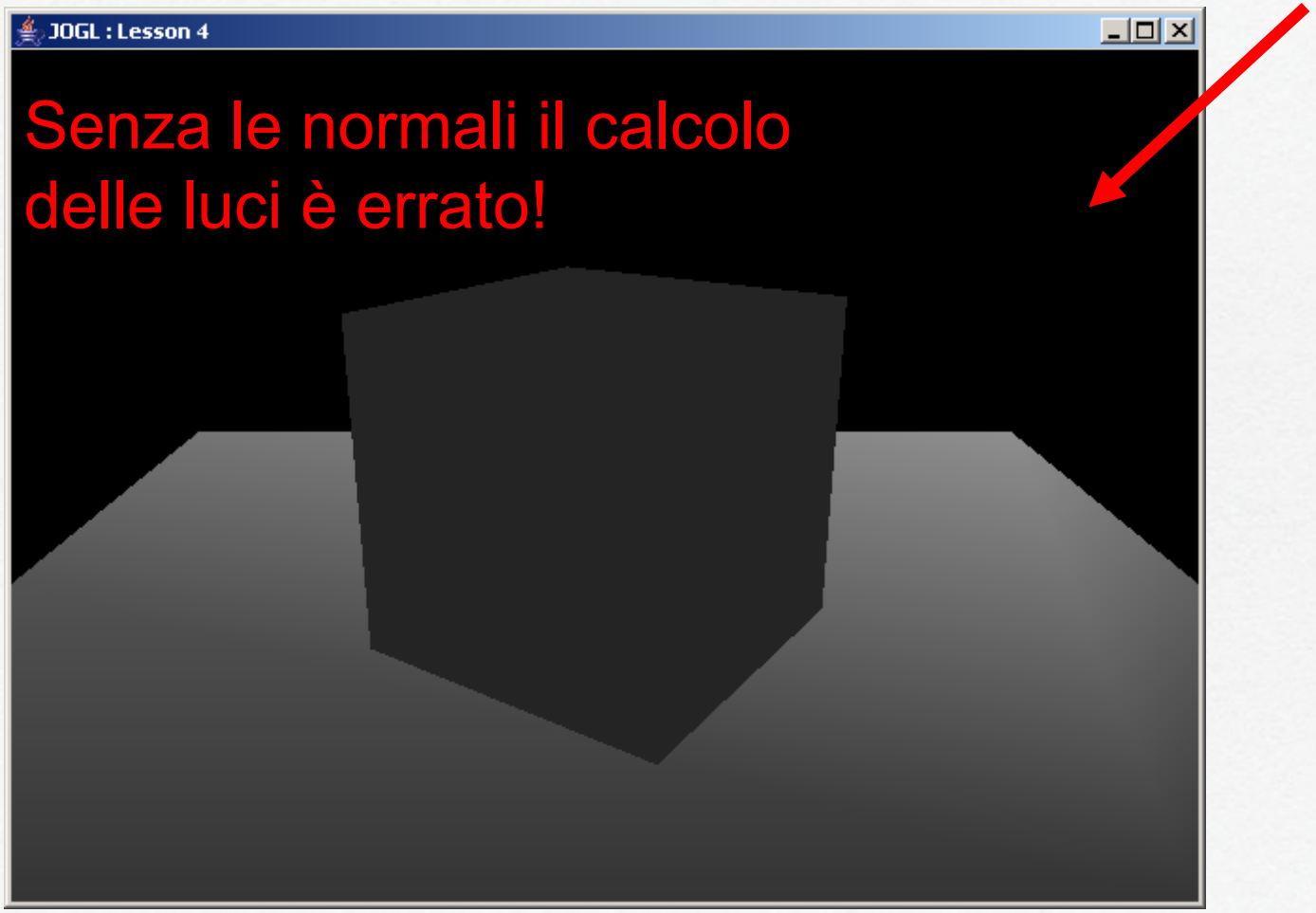
Esempio 4: illuminazione

- Si vuole creare un piano ed un cubo che ruota al di sopra di esso
  - Creazione di una luce
  - Creazione della normale
  - Aggiungere le ombre

Esempio 4: init

- //Definisco il colore ambientale della luce
- **float[] ambient = {0.5f, 0.5f, 0.5f, 1.0f};**
- //Definisco il colore della luce
- **float[] diffuse = {1.0f, 1.0f, 1.0f, 1.0f};**
- //Definisco la posizione della luce
- **float[] position = {8.0f, 8.0f, 0.0f, 1.0f};**
- //Imposto il colore ambientale della luce 1
- **gl.glLightfv(GL.GL_LIGHT1, GL.GL_AMBIENT, ambient);**
- //Imposto il colore della luce 1
- **gl.glLightfv(GL.GL_LIGHT1, GL.GL_DIFFUSE, diffuse);**
- //Imposto la posizione della luce 1
- **gl.glLightfv(GL.GL_LIGHT1, GL.GL_POSITION, position);**
- //Attivo la luce 1
- **gl.glEnable(GL.GL_LIGHT1);**
- //Attivo l'illuminazione
- **gl.glEnable(GL.GL_LIGHTING);**

Esempio 4: risultato



Esempio 4: display

- //Definisco il colore del piano
- gl.glColor3f(1.0f,1.0f,1.0f);//Bianco

- //Definisco il vettore normale al piano
- gl.glNormal3f(0.0f, 1.0f, 0.0f);

- //Definisco i vertici che compongono il quadrilatero
- gl.glVertex3f(4.0f, -1.0f,-4.0f);
- gl.glVertex3f(-4.0f, -1.0f,-4.0f);
- gl.glVertex3f(-4.0f, -1.0f, 4.0f);
- gl.glVertex3f(4.0f, -1.0f, 4.0f);

Esempio 4: risultato

Esempio 5: input

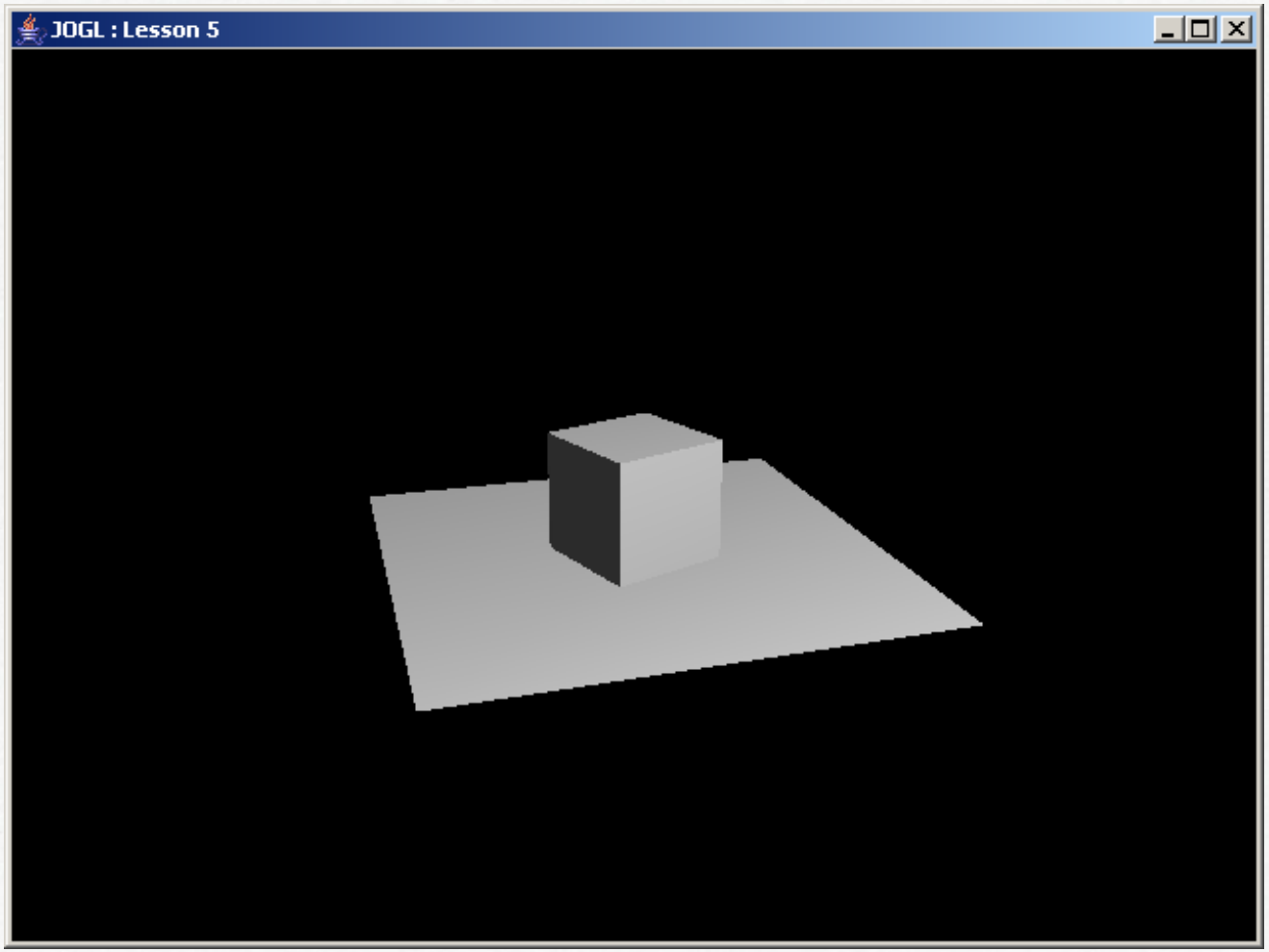
- Si vuole creare un piano ed un cubo che ruota al di sopra di esso
  - Leggere l'input da tastiera
  - Ruotare e avvicinare la cinepresa

Esempio 5: KeyListener

```
public void keyReleased(KeyEvent e){}
public void keyTyped(KeyEvent e){}
public void keyPressed(KeyEvent e){
 switch(e.getKeyCode()){
 case KeyEvent.VK_LEFT:
        tumble += 1.0f;
        break;
 case KeyEvent.VK_RIGHT:
        tumble -= 1.0f;
        break;
 case KeyEvent.VK_UP:
        dolly += 0.5f;
        break;
 case KeyEvent.VK_DOWN:
        dolly -= 0.5f;
        break;
 }
}
```

- ...
- gl.glTranslatef(0.0f, 0.0f, dolly);
- gl.glRotatef(20, 1.0f, 0.0f, 0.0f);
- gl.glRotatef(tumble, 0.0f, 1.0f, 0.0f);
- ...

Esempio 5: risultato



Appunti di

Computer Grafica

Bibliografia e Links

Software per grafica 2D

- Photoshop - www.adobe.com
- PaintShop Pro - www.jasc.com
- Illustrator - www.adobe.com
- Freehand - www.macromedia.com
- Painter - www.corel.com
- GIMP - www.gimp.org (FREE!)

Software per video

- QuickTime - www.quicktime.com
- Final Cut - www.apple.com
- Premiére - www.adobe.com
- VirtualDub - virtualdub.sourceforge.net(FREE!)
- Cleaner - www.discreet.com
- Flash - www.macromedia.com

Software per grafica 3D

- 3D Studio Max - www.discreet.com
- Maya - www.aliaswavefront.com
- Softimage - www.softimage.com
- LightWave - www.newtek.com
- RenderMan - www.pixar.com
- Blender - www.blender3d.org (FREE!)
- Milkshape - www.download.com(FREE!)

Tutorial grafica 3D

- www.3dluvr.com
- www.3drender.com
- www.3dtotal.com
- www.highend3d.com
- www.thegnomonworkshop.com
- www.aliaswavefront.com
- www.3dark.com

Programmazione

- www.gamedev.net
- www.gamasutra.com
- www.flipcode.com
- www.java.net
- java.sun.com
- http://www.jars.com/
- www.opengl.org
- http://www.xmission.com/~nate/index.html
- https://jogl.dev.java.net