



# WebGL

Lezione 19: 24 Maggio 2011

# Cronologia: Grafica 3D nell'Hardware

---

- ❑ In principio (giurassico informatico) postazioni specializzate
- ❑ La Silicon Graphics si afferma come produttrice di workstation grafiche in serie (molto costose!)
- ❑ Hardware 3D estremamente limitato entra nelle case nei tardi anni '80
- ❑ Dal 2000 in poi le potenzialità degli acceleratori 3D aumentano drasticamente, scendono i prezzi

# Cronologia: Grafica 3D nel Software

---

- ❑ Programmi specializzati, principalmente per ricerca e uso scientifico
- ❑ Uso industriale (progettazione, validazione, ...), industria cinematografica
- ❑ Produzioni low-cost, primi videogiochi, progetti culturali sperimentali
- ❑ Intrattenimento (giochi, pubblicità, ...), beni culturali, supporto alla visualizzazione, ...

# Fruizione dei contenuti 3D

---

- Prima del boom di Internet i contenuti 3D erano usati esclusivamente in software specializzati
  - ... che facevano "solo" quello
  - ... 3D non rientrava nel "multi" di *multimedia*
  
- La (decisamente più *multimediale*) natura delle web application ha spinto gli sviluppatori a creare software per integrare la visualizzazione di modelli 3D in pagine html

# Web e 3D

---

## □ Prime apparizioni

- Oggetti ActiveX per Internet Explorer (Microsoft)
- Implementazioni proprietarie
- In pratica, codice "esterno" alla pagina web

## □ Plugin / Estensioni multiplatforma

- Moduli software utilizzabili da sistemi operativi e browser diversi
- Ancora, codice esterno alla pagina

# Web e 3D

---

- Problemi delle soluzioni ad-hoc/proprietarie
  - Scoraggiano gli sviluppatori
  - Il software deve essere installato dagli utenti
  
- Necessità di uno standard
  - Che sia abbastanza flessibile da soddisfare le diverse esigenze di sviluppatori e utenti
  - Che non abbia prerequisiti di nicchia
  - Che si integri facilmente ed elegantemente con le tecnologie esistenti

# La tecnologia dei documenti ipertestuali

---

- HTML: Hyper Text Markup Language
  - Nasce con la possibilità di riferire altri documenti
  
- Multimedialità
  - Permette l'integrazione di informazioni rappresentate in maniera drasticamente diversa (testo, immagini, video, audio)
  
- CSS: Cascading Style Sheet
  - Si aggiunge la possibilità di definire (con un apposito linguaggio standardizzato) l'aspetto del documento

# La tecnologia dei documenti ipertestuali

---

## □ Scripting

- Inserire algoritmi all'interno di documenti
- JavaScript il linguaggio adottato come standard in documenti html

## □ Primi usi del JavaScript

- Effetti grafico/stilistici (menu, contenuti dinamici, ...)
- Comunicazioni con il server
- Odiosità varie tipo "quando clicco sul link si apre un favoloso popup" etc etc etc etc etc etc etc etc

## □ Successivi utilizzi del JS decisamente più complessi



# Standardizzare la Grafica 3D per il Web

---

- Definire uno standard per il 3D su web che utilizzi tecnologie largamente diffuse
  - Ha il vantaggio di avere un ampio bacino di sviluppatori
  
- OpenGL, il punto di partenza
  - Specifica di funzionalità grafiche multiplatforma
  - Utilizzata dai cellulari alle workstation
  - Vastissima comunità di sviluppatori
  
- Da OpenGL, definire un'API JavaScript per sfruttare l'HW 3D *dall'interno* di pagine html
  - Perplessità sulle prestazioni ...

# I tempi sono maturi

---

- I linguaggi di scripting sono considerati “lenti” rispetto a quelli compilati (C/C++ etc.)
  - Vero.
  - Vero, ma le prestazioni sono migliorate sensibilmente
  - Vero, ma, come sempre, dipende da quello che dobbiamo fare
  
- *Cosa* dobbiamo fare?
  - Sicuramente non processing “pesante”
  - Sicuramente visualizzazione
  
- *Come* lo facciamo?
  - Delegando i compiti più complessi all’HW 3D

# WebGL

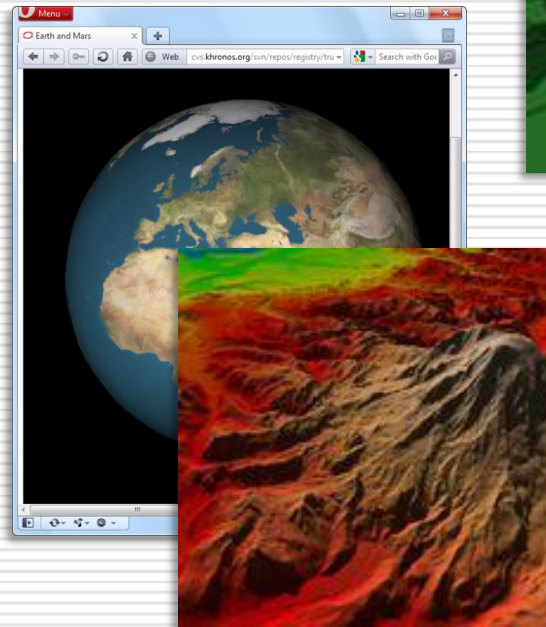
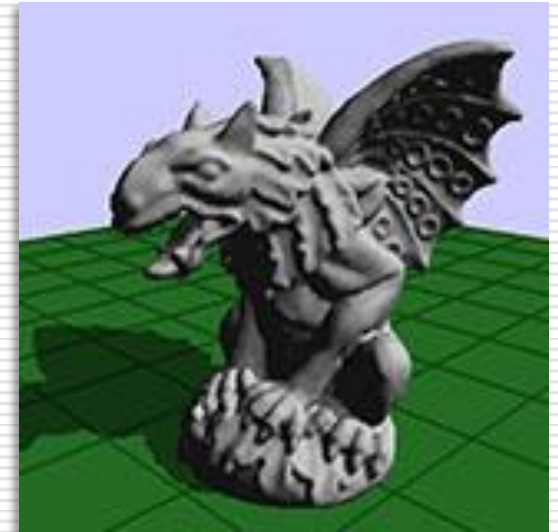
---

- ❑ API per la grafica 3D in JavaScript
- ❑ Definito a partire dalle specifiche OpenGL|ES 2.0
- ❑ <http://webgl.org>



# WebGL in use

---



# WebGL

---

- Basandosi su OpenGL|ES 2.0 è a basso livello
  - ☹ Difficile(?) e tediosa da usare
  - 😊 Alte prestazioni
  
- Esistono diverse librerie per semplificarne l'utilizzo, ognuna mirata a un diverso target di utenti (es. [www.khronos.org/c3dl/](http://www.khronos.org/c3dl/) )
  - Dichiarative
  - Procedurali
  - Per la visualizzazione scientifica
  - Per effetti 2D
  - ...

# SpiderGL

---

- ❑ Libreria sviluppata al Visual Computing Lab
- ❑ Procedurale
- ❑ Semplifica l'uso di WebGL
- ❑ Flessibile
  
- ❑ <http://spidergl.org>
  
- ❑ Altre risorse e librerie
  - Learning WebGL – <http://www.learningwebgl.com>
  - GLGE – <http://www.glge.org>
  - SceneJS – <http://scenejs.org>

# Richiami sulla pipeline grafica

---

Application

→ Specifica la geometria e i materiali

Vertex Processing

→ Calcola gli attributi per vertice

Primitive Processing

→ Esegue computazioni per primitiva

Rasterizer

→ Individua i pixel da "accendere"

Fragment Processing

→ Calcola il colore del singolo pixel

Output Merger

→ Rimuove superfici nascoste

Framebuffer

→ Contiene l'immagine da visualizzare

# Disegnare un modello 3D

---

- ❑ Specificare la forma (geometria)
- ❑ Specificare le proprietà ottiche (materiali)
- ❑ Definire la posizione e l'orientamento dell'oggetto
- ❑ Definire le sorgenti di luce
- ❑ Definire il punto di vista



# Più in dettaglio

---

- Specificare la geometria
  - Attributi dei vertici (posizione, colore, ...)
  - Connettività (per formare triangoli)
- Specificare i materiali
  - Texture (immagini da “disegnare” sui poligoni)
  - Come il materiale reagisce alla luce
- Posizionare l’oggetto nel mondo
  - I vertici sono, in origine, nello *spazio di modellazione*
  - Posizione e orientamento nel mondo vengono espressi tramite matrici di trasformazione (trasformano le coordinate dallo *spazio di modellazione* allo *spazio mondo*)

# Più in dettaglio

---

- Definire le sorgenti di luce
  - La loro posizione nel mondo
  - Il loro colore
  
- Definire il punto di vista
  - L'immagine generata è quella che proviene da una ipotetica macchina fotografica
  - Sempre con matrici di trasformazione, definiamo posizione e orientamento della fotocamera

# Computazioni per Vertice e per Pixel

---

- Programmi in linguaggio GLSL
  
- Vertex Shaders
  - Trasformano i vertici dallo spazio di modellazione allo spazio di proiezione
  - Computano attributi ausiliari (colore, ...)
  
- Fragment Shaders
  - Ricevono in input gli attributi (interpolati) emessi dai vertex shaders
  - Calcolano il colore finale del pixel

# In SpiderGL

---

## □ Inizializzazione

- Creare ed inizializzare una Mesh
- Creare i Vertex e Fragment shader

## □ Definire le matrici per posizionare la mesh e la telecamera

- Possono essere definiti una sola volta in fase di inizializzazione o ad ogni frame in caso di animazioni o cambiamenti del punto di vista

## □ Demo!

# Conclusioni

---

- WebGL porta il 3D nel web, grandi potenzialità
- MA (problemi)
  - Compatibilità coi browser?
  - Tradeoff tra facilità d'uso e potenzialità
  - Interazione e processing

# Next in line...

---

Next lesson:

- Wrap-up

Contacts:

Matteo Dellepiane

c/o ISTI-CNR Via G. Moruzzi 1

56124 Pisa (PI)

Tel. 0503152925

E-Mail: [dellepiane@isti.cnr.it](mailto:dellepiane@isti.cnr.it)

Personal website: <http://vcg.isti.cnr.it/~dellepiane/>

VCG website: <http://vcg.isti.cnr.it>

---