

# **Corso di *Grafica Computazionale***

***Real-Time Rendering  
Introduzione all'hardware grafico***

**Docente:  
Massimiliano Corsini**

**Laurea Specialistica in Ing. Informatica**

**Facoltà di Ingegneria**

**Università degli Studi di Siena**

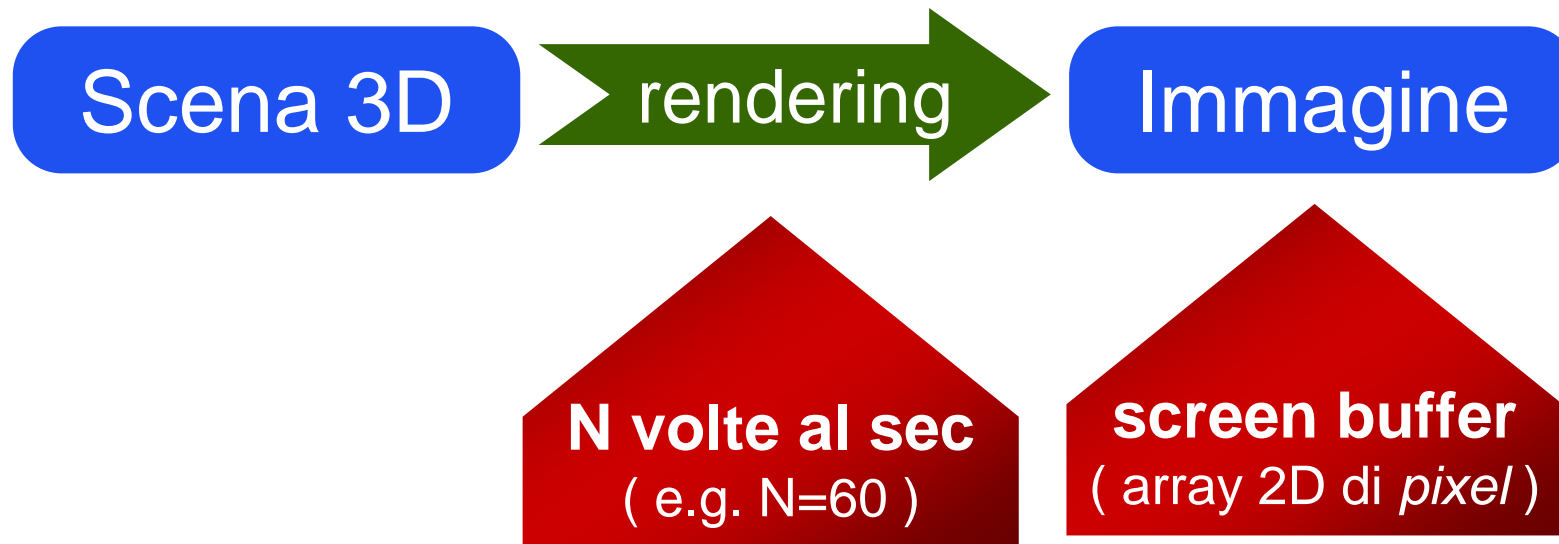


# Hardware Grafico

Facoltà di  
Ingegneria

- Caratterizzato dalla cosiddetta *pipeline di rendering*
- La *pipeline di rendering* è la serie di stages di elaborazione che i dati della scena attraversano per diventare immagine
- La moderna pipeline di rendering è *programmabile* → *vertex shaders, pixel shaders*
- *Parallelismo estremo* per garantire alte prestazioni





**Double-buffering** → permette di eliminare effetti di flickering  
Un buffer viene usato per disegnare, un altro viene mostrato,  
al momento opportuno si scambiano rapidamente.



# Transfer-Rate

- **pixel** = 32 bit = 4 bytes ("pixel depth")
- **screen buffer** = 1024 x 768 pixels ("screen resolution")
- **frame rate** = 60 Hz ("fps")
- **total (fill-rate)** = 4 x 1024 x 768 x 60 byte al sec

**188 MegaBytes / sec**



- Vantaggio di disporre di hardware specializzato per la grafica: **efficienza**
  - computazioni più ripetute hard-wired nel chipset
  - resto dell'applicazione libera di utilizzare la CPU e RAM base
  - sfruttamento del **parallelismo** implicito nel problema di rendering
    - sotto forma di parallelismo di pipeline
    - sotto forma di parallelismo *in ogni fase* del pipeline
- Svantaggio: **rigidità**
  - vincola l'approccio usato per fare rendering...



# Paradigma: Rasterization-based

Facoltà di  
Ingegneria

- Raytracing
- Rasterization based
- Image based (per es. light field)
- Radiosity
- Photon Mapping
- ...

## RASTERIZZAZIONE DI TRIANGOLI



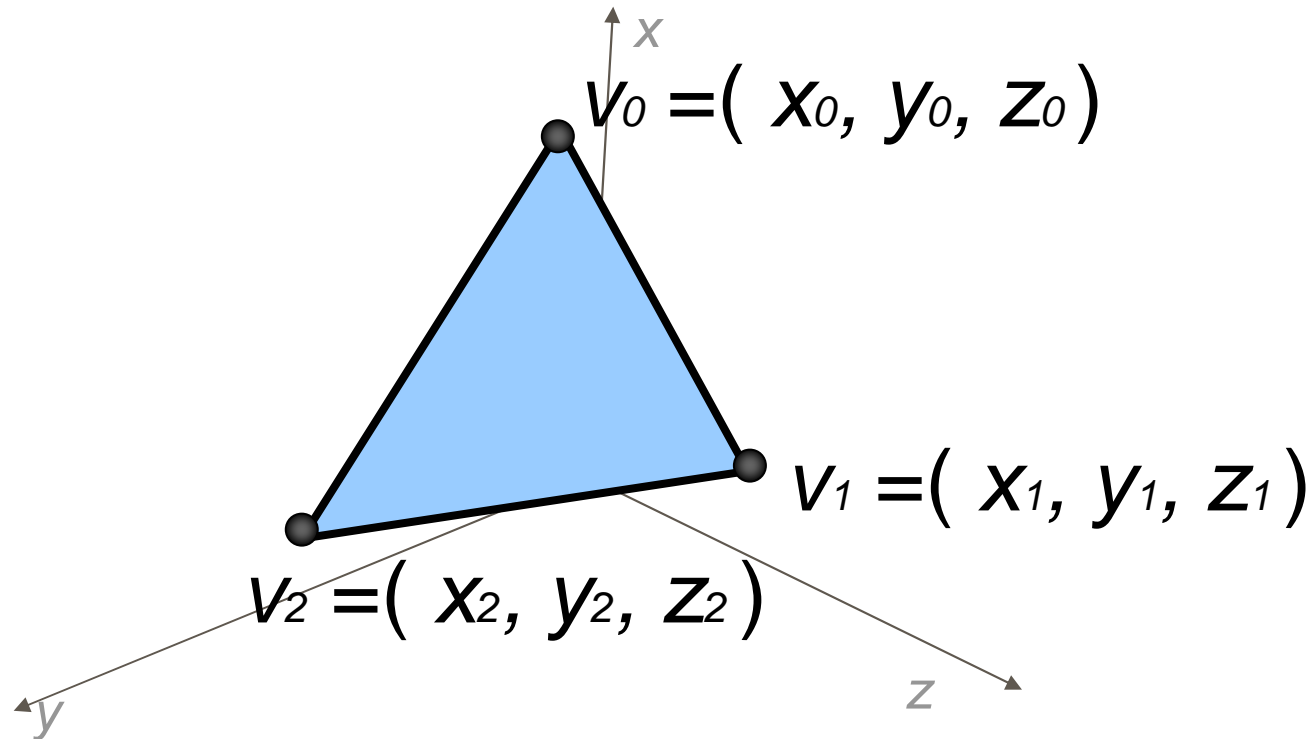
# Un pò di storia...

- ~20 anni:
  - dalla metà degli '80 (e.g.: SGI Iris - 1986)
    - dalla metà dei '90 si comincia a passare dai mainframes ai PC
- progressi enormi
  - nella efficienza
    - piu' che "Moore's Law": **~2.4x / year** invece di **~1.6 / year**
  - nella funzionalità
    - Daremo una sguardo alle architetture più moderne e qualche cenno al futuro (molto prossimo) della pipeline.

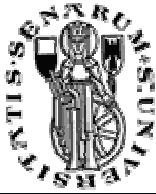


# Rasterizzazione di Triangoli

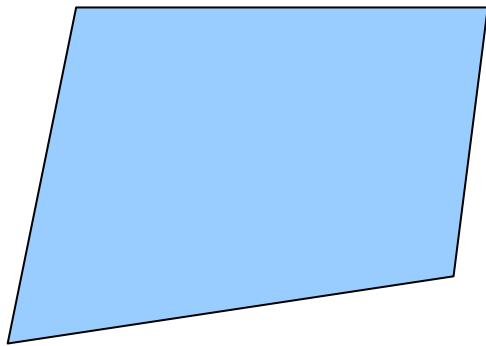
- punto primo: **tutto sia composto da triangoli (3D)**
  - o al limite da punti o segmenti





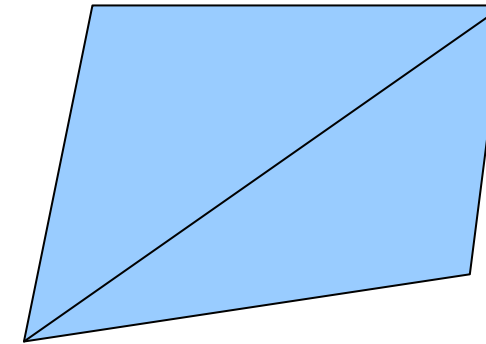


# Quadrilateri → triangoli



un quadrilatero?

↑  
"quad"

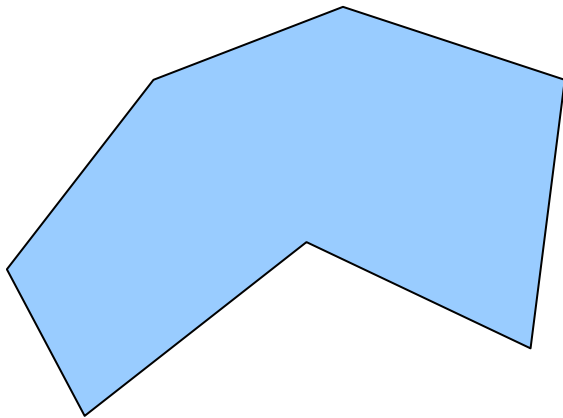


due triangoli!

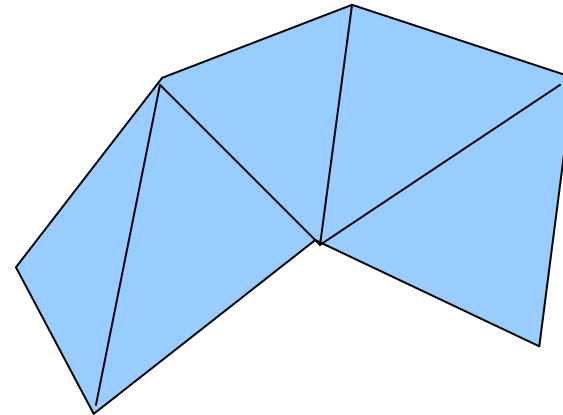
↑  
"diagonal split"



# Poligoni → triangoli



un poligono a  $n$  lati?

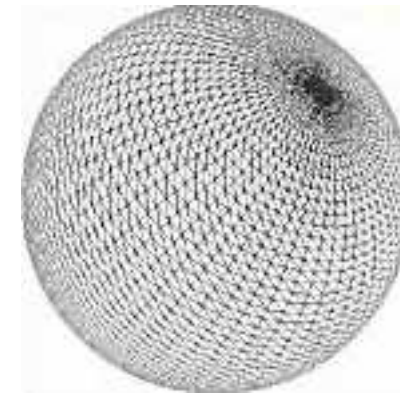
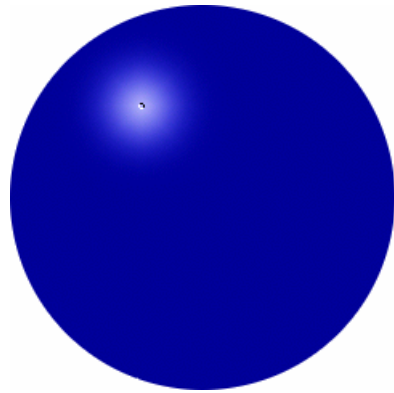


$(n-2)$  triangoli!

↑  
triangolarizzazione di un poligono  
non un problema banale



# CSG → triangoli

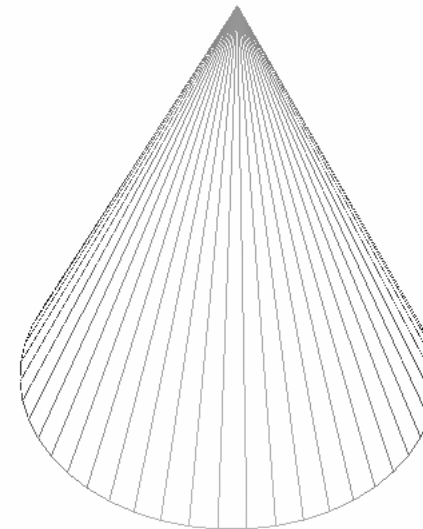
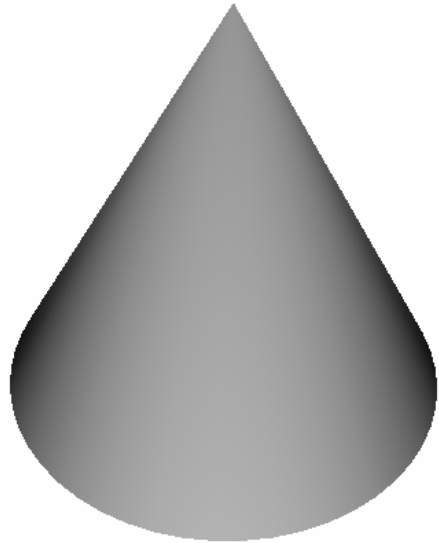


la superficie di  
un solido geometrico,  
per es. una sfera?

triangoli!



# CSG → triangoli

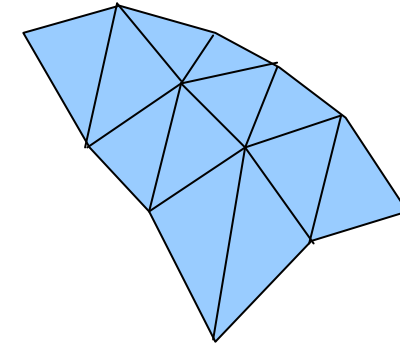
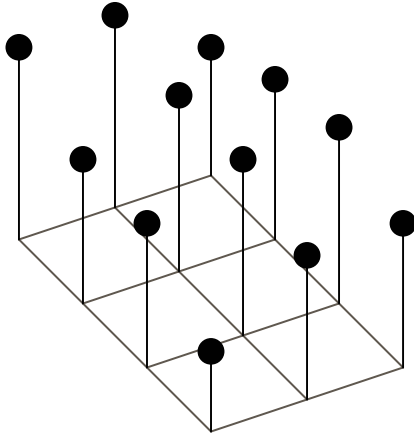


la superficie di  
un solido geometrico,  
per es. una cono?

triangoli!

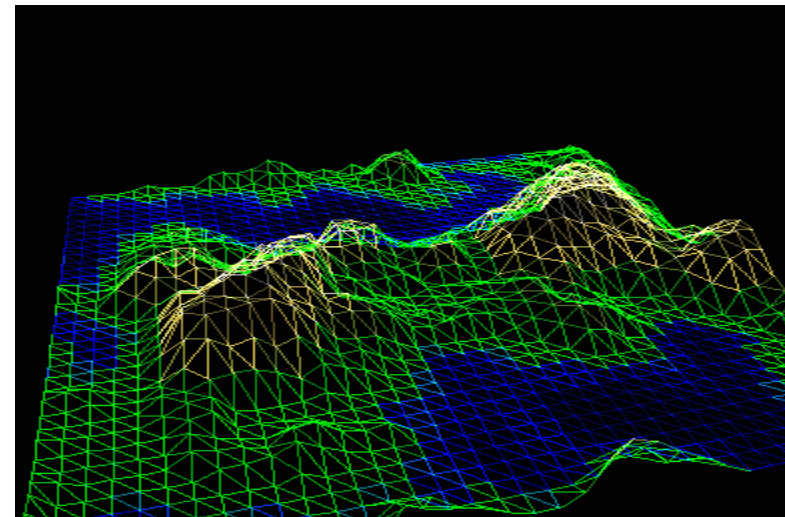


# Height field $\rightarrow$ triangoli

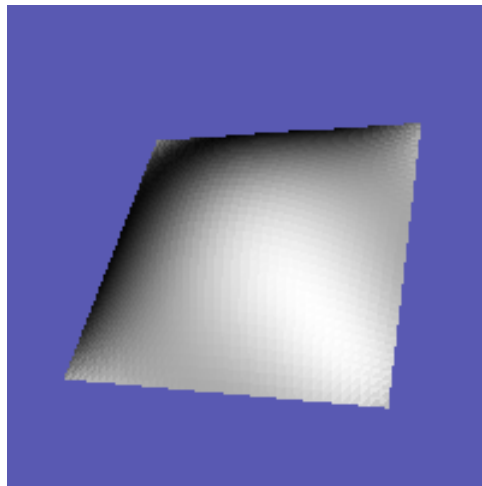


triangoli!

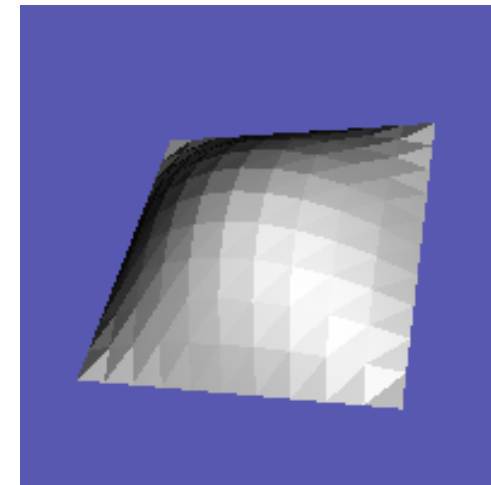
un campo d'altezza?  
(array 2D di altezze,  
e.g. per modellare un terreno?)



## Superfici parametriche → triangoli



una superficie curva  
parametrica?  
per es. NURBS \*, b-splines \*...

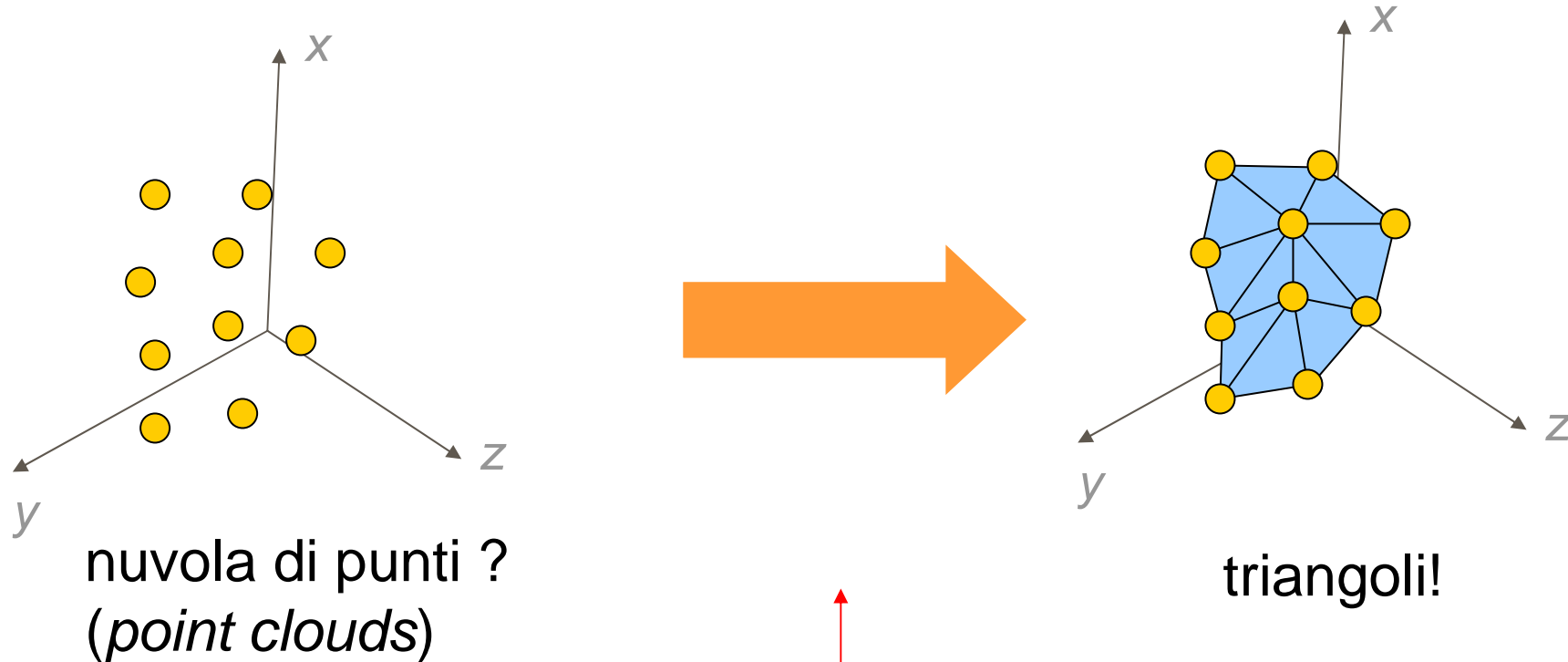


triangoli!

questo è facile. Il contrario, che qualche  
volta è utile, MOLTO meno (fitting)



# Point Clouds → triangoli



↑  
problema molto studiato,  
e (nel caso generale) difficile



# Point Clouds → triangoli

Facoltà di  
Ingegneria







# Voxels $\rightarrow$ triangoli



volume?



triangoli  
che definiscono  
una  
iso-superficie

triangoli!



algoritmi di segmentation  
(e.g. "marching cubes" \*)



# Superfici implicite $\rightarrow$ triangoli



superfici implicite?



triangoli  
che definiscono  
la superficie  
esplicitamente

triangoli!

nb: non c'e' un modo solo per farlo.  
Modo + semplice (non ottimo): campionare volume e  
estrarre isosuperficie a valore 0



# Non solo triangoli...

- Non sempre e' semplice modellare le entità da rappresentare con triangoli, servono anche punti e linee
  - Esempi di applicazione:
    - Punti (particle system): nuvole, fuoco
    - Linee: pelliccia di animali, barba, baffi, capelli



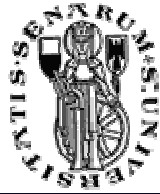
by Niniane Wang



by N. Adabala Univ. Florida



by M. Turitzin and J. Jacobs  
Stanford University



# Domande?