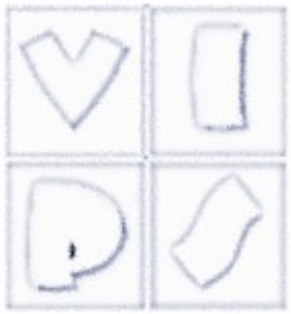


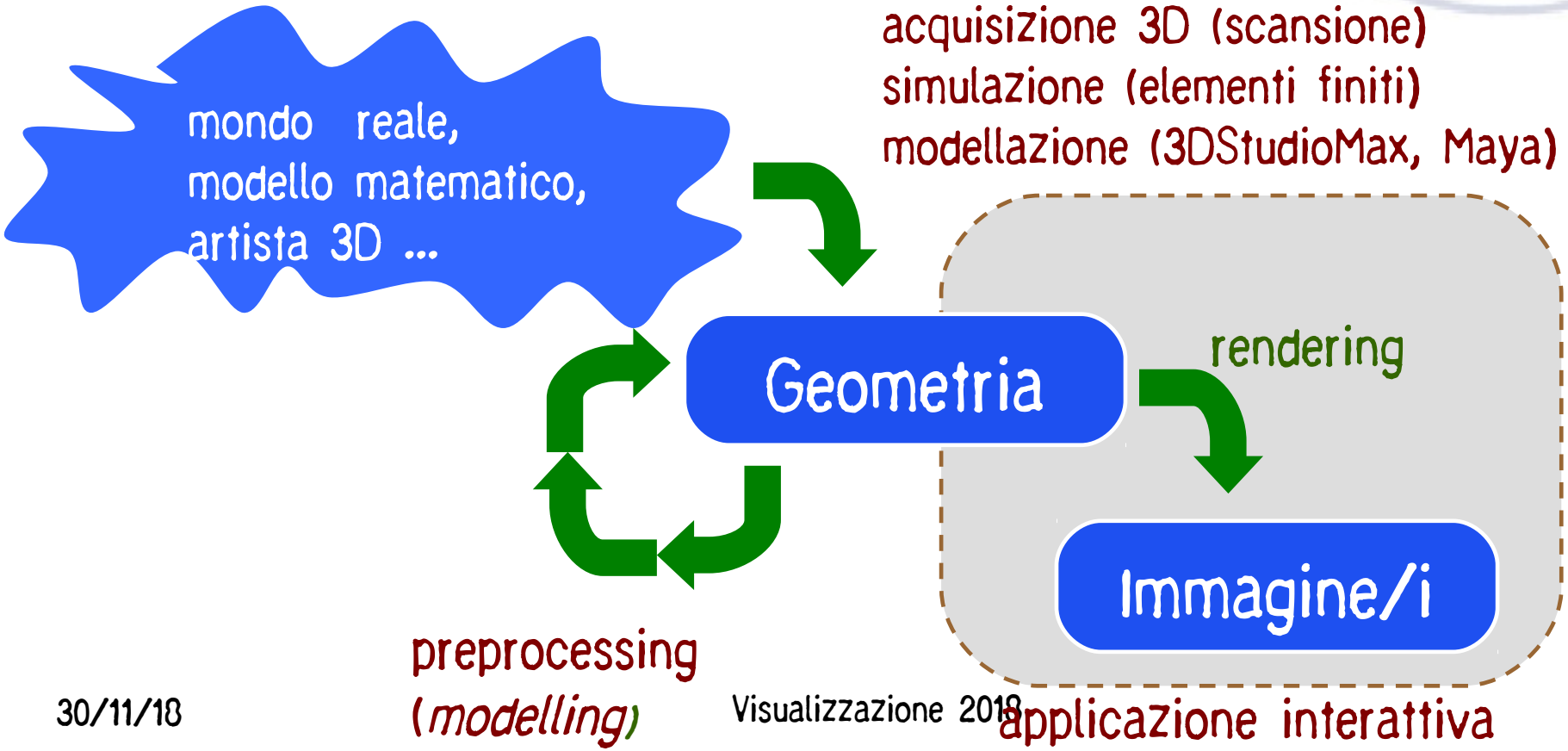
Fondamenti di Grafica al calcolatore

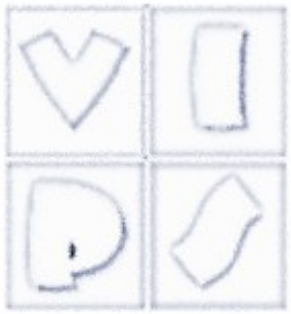
3 - Modellazione di oggetti e scene

andrea.giachetti@univr.it

Modellazione

- Due problemi da affrontare nelle applicazioni grafiche
  - Modellazione degli oggetti
  - Rendering: simulazione della formazione delle immagini



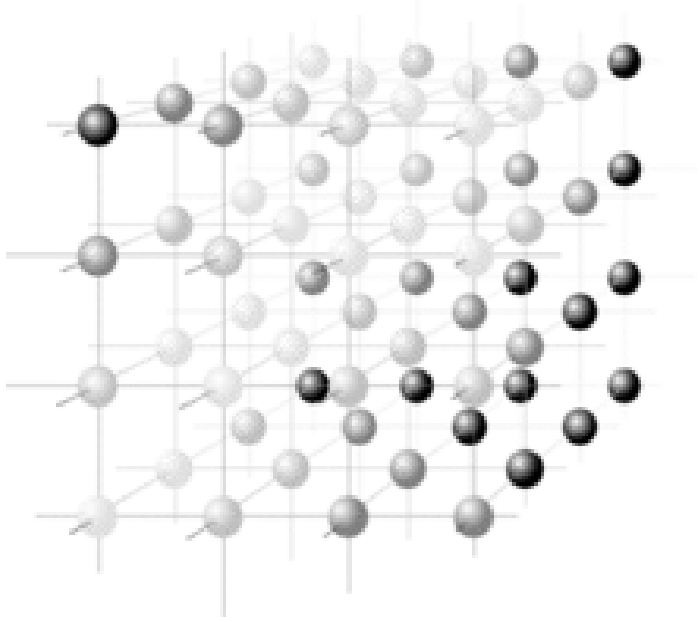


Modellazione

- Nelle applicazioni grafiche si modella una scena con degli oggetti 2D o 3D da trasformare in immagini digitali
- Consideriamo il caso 3D
- Potemmo usare molti modi di rappresentare gli oggetti
  - Primitive geometriche
  - Suddivisione spaziale
  - Campionamento di punti volumetrico o di superficie
  - Funzioni parametriche
- In effetti in diversi ambiti si usano diversi metodi

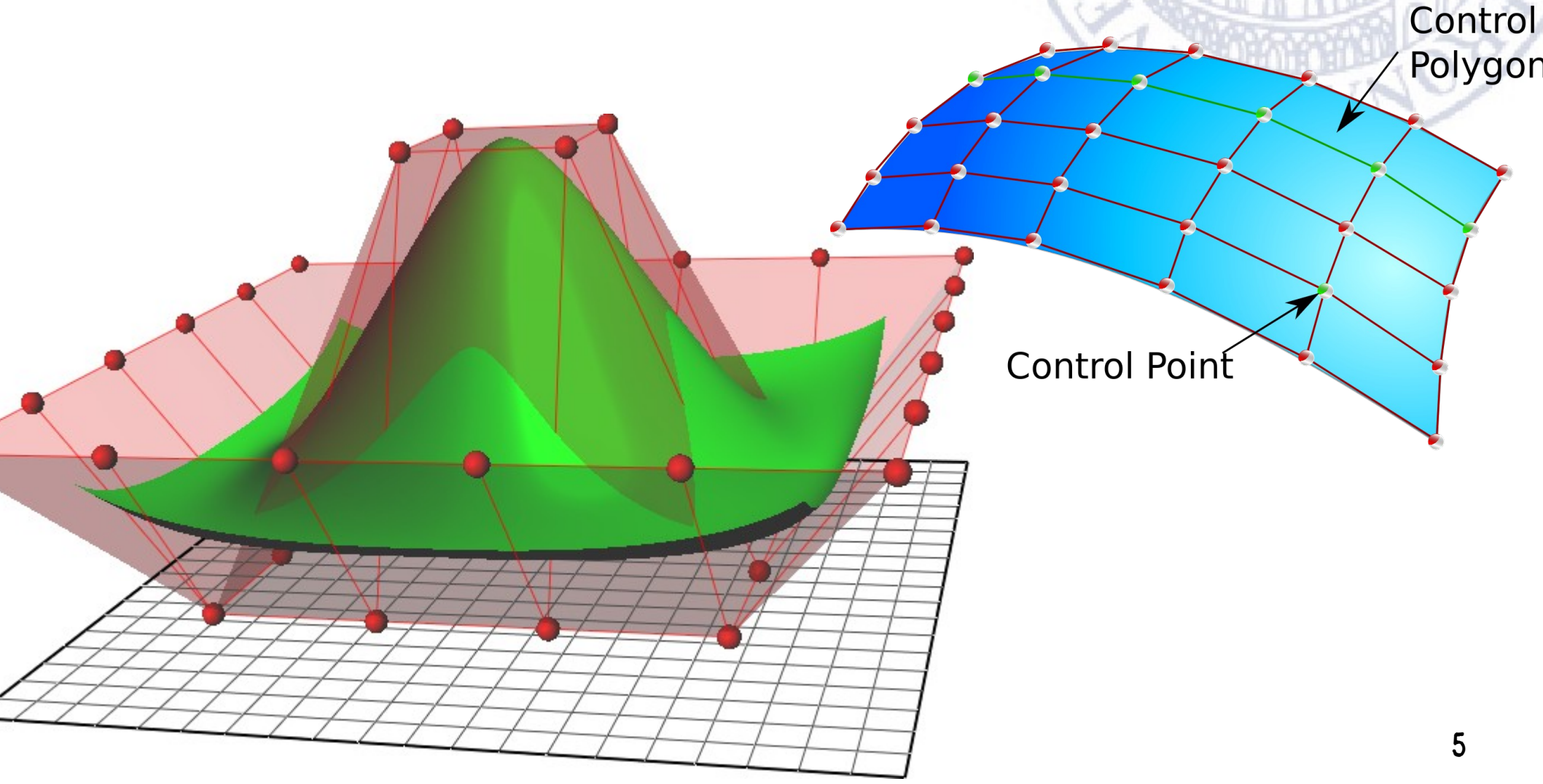
Partizionamento spaziale

- La scena è divisa in cubetti con assegnati attributi (es. densità)
- Tipico in visualizzazione medica



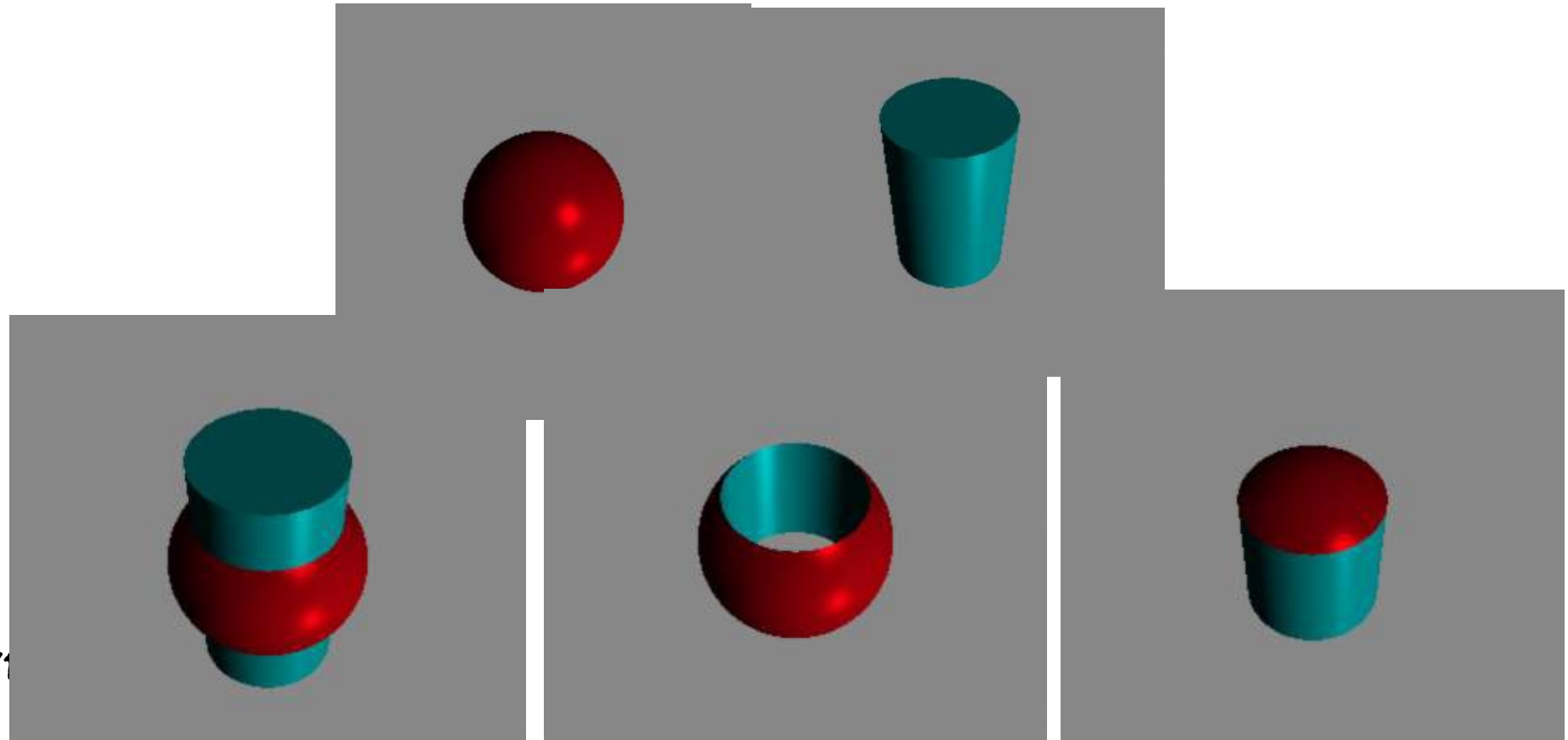
Funzioni parametriche

- Es. superfici parametriche (NURBS, splines)



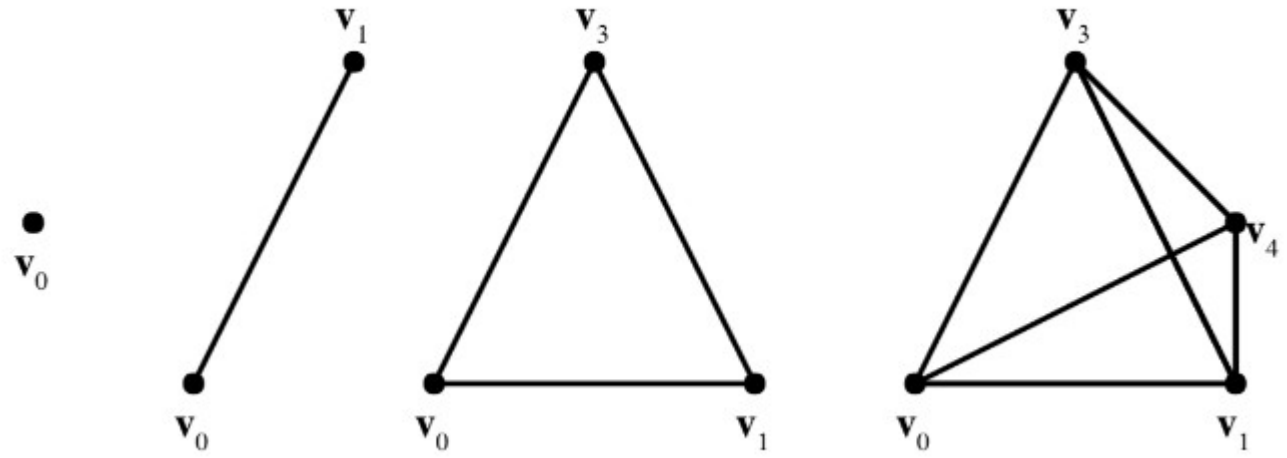
Geometria costruttiva solida (CSG)

- Altra rappresentazione particolarmente adatta per il modeling (diffusa nel settore CAD), ma poco efficiente per il rendering.
- Si tratta, essenzialmente, di costruire degli oggetti geometrici complessi a partire da modelli base con operazioni booleane



Modellazione

- Necessità di usare rappresentazioni condivise e standardizzate
- Ottimali per gli algoritmi di rendering, animazione, ecc
- Quindi i programmi di modellazione/rendering/grafica usano di solito un particolare tipo di modellazione legato alla approssimazione di superfici con modelli poligonali
- Gli oggetti sono rappresentati con punti, spigoli e poligoni, spesso solo triangoli
- Per modellazione volumetrica si usano i poliedri

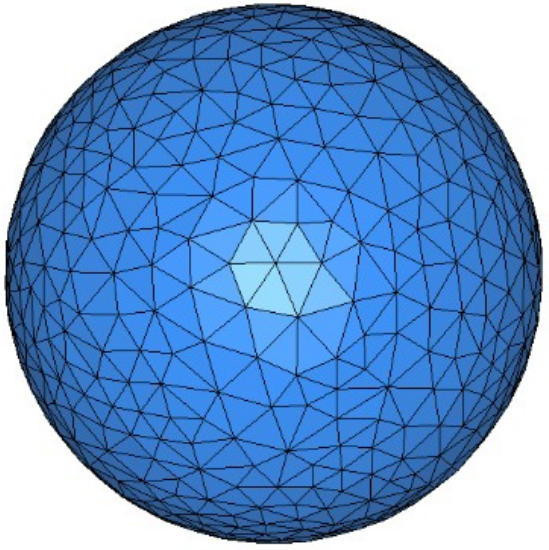


Maglie (mesh) di poligoni

- Nella grafica 3D interattiva si usa quasi sempre la modellazione basata su approssimazione poligonale degli oggetti (del loro contorno: è una *boundary representation*).
- Si tratta di approssimare una superficie 2D con un insieme di poligoni convessi opportunamente connessi gli uni agli altri.
- Nella pipeline di rendering si lavora in genere con i soli triangoli
-

[Andrzej Barabasz]

spheres

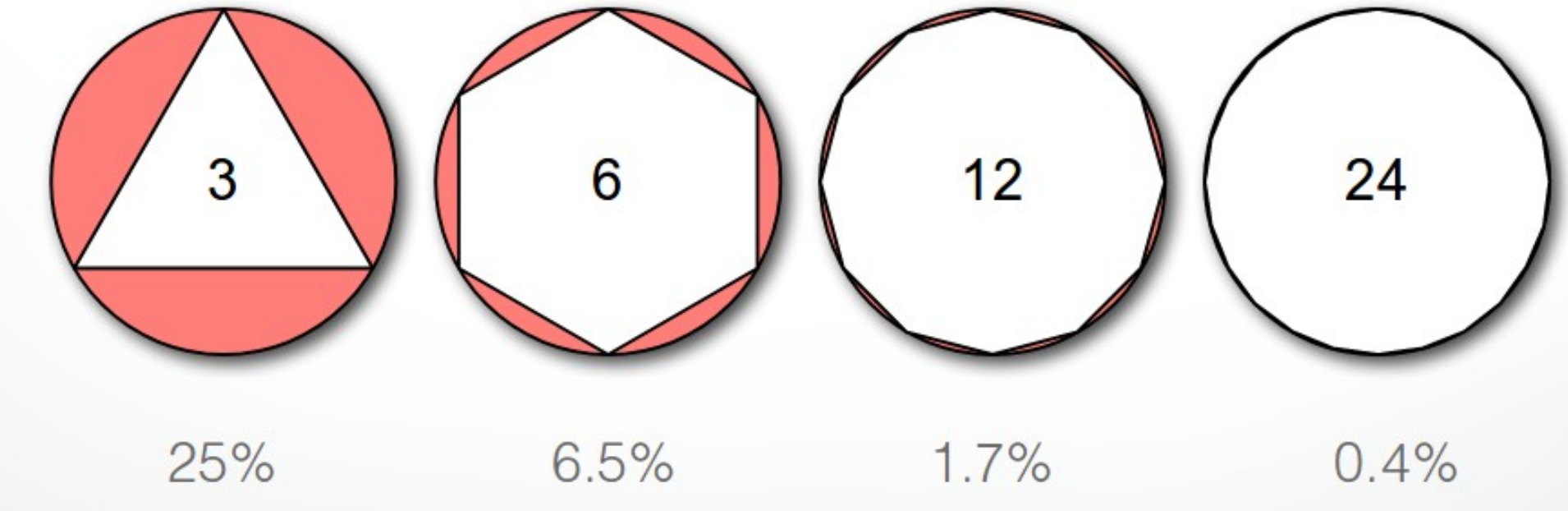


[Rineau & Yvinec CGAL manual]

approximate

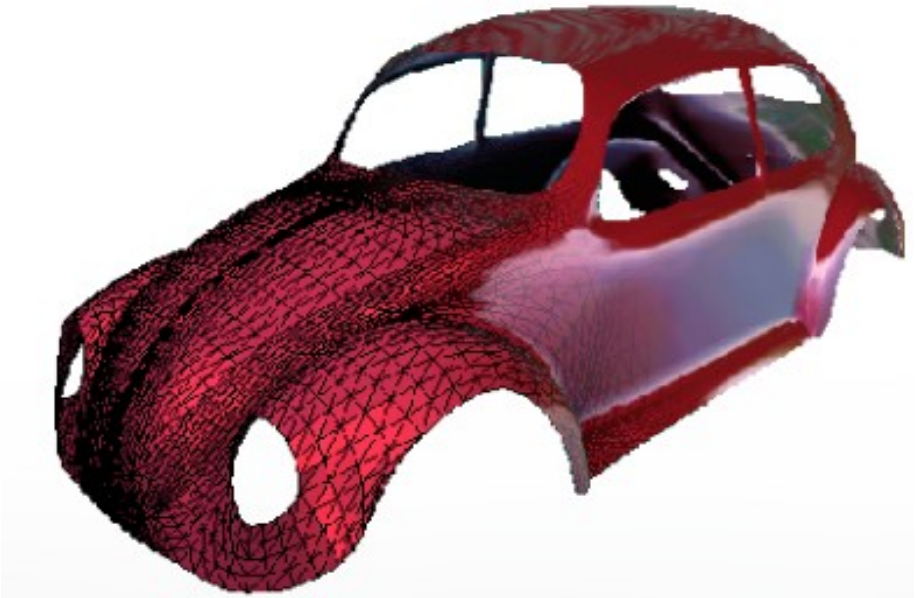
Maglie (mesh) di poligoni

- L'errore di approssimazione è inversamente proporzionale al numero di facce. Esempio 2D



Mesh poligonali

- Rappresentazione lineare a tratti
- Può rappresentare superfici con qualunque topologia (numero buchi, componenti connesse)
- L'accuratezza può essere adattata localmente
- Il rendering è ottimizzato sulle GPU

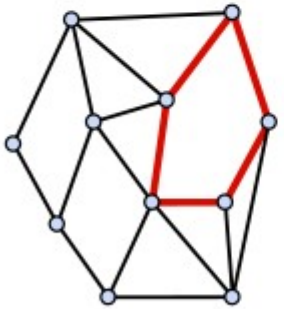


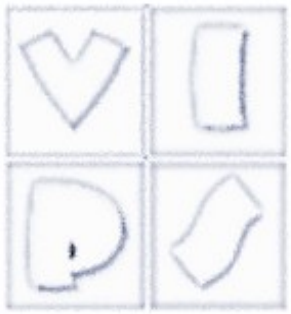
Mesh (maglie) di poligoni

- I poligoni sono definiti da una catena chiusa di punti nello spazio 3D.
- I poligoni si dicono semplici se non si autointersecano
- Se tutti i punti sono su un piano si dice poligono planare
- Una maglia (mesh) poligonale è un insieme di poligoni le cui intersezioni siano esclusivamente vertici e lati (spigoli) dei poligoni
- Due poligoni che condividono un lato si dicono adiacenti
- I poligoni della maglia si chiamano anche facce.

$$v_0, v_1, \ldots, v_{n-1}$$

$$\{(v_0, v_1), \ldots, (v_{n-2}, v_{n-1})\}$$





Mesh (maglie) di poligoni

- Mesh
  - Geometria
    - vertici, che sono punti (x,y,z) che campionano la superficie
  - Connettività (topologia)
    - Come i vertici sono connessi
      - Ogni vertice può avere un numero vario di spigoli incidenti (che definiscono il grado di un vertice)
  - Bordo
    - Il bordo di una mesh corrisponde agli spigoli che appartengono a un solo poligono
      - Una mesh senza bordo si dice chiusa
  - Attributi
    - Sui vertici o le facce della mesh posso associare attributi (eventualmente da visualizzare)
      - Misure, ma anche colore, materiale, normali, coordinate texture

Mesh di triangoli

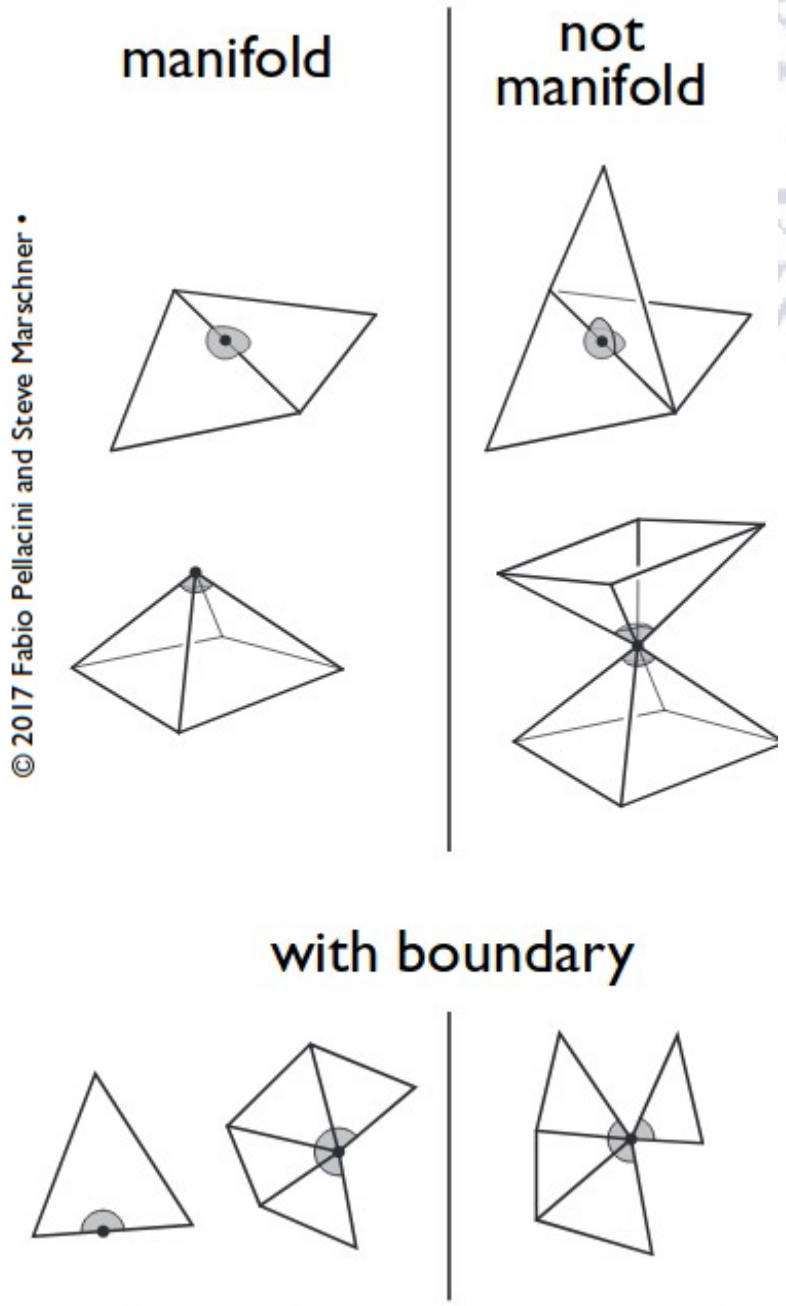
- Una maglia (mesh) triangolare è un'insieme di triangoli le cui intersezioni siano esclusivamente vertici e lati (spigoli) dei triangoli e che sia anche una varietà bidimensionale con bordo.
- Due triangoli che condividono un lato si dicono adiacenti
- I triangoli della maglia si chiamano anche facce.
- La condizione di essere varietà o 2-manifold i traduce nei seguenti vincoli sulla struttura:
  - uno spigolo appartiene al massimo a due triangoli (quelli eventuali che appartengono ad uno solo formano il bordo della maglia)
  - se due triangoli incidono sullo stesso vertice allora devono essere raggiungibili l'uno dall'altro attraverso un percorso tra triangoli adiacenti ovvero devono formare un ventaglio o un ombrello.

Manifoldness (Varietà)

- Per rappresentare la superficie di un oggetto, la superficie dovrebbe soddisfare la condizione di essere 2-manifold (essere una varietà bidimensionale)
- Significa che l'intorno di ogni punto deve essere isomorfo a un disco

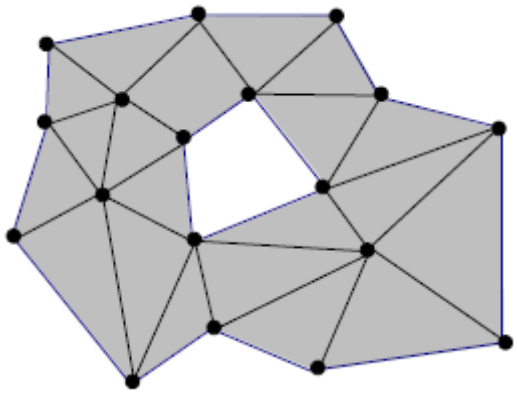
Mesh di triangoli

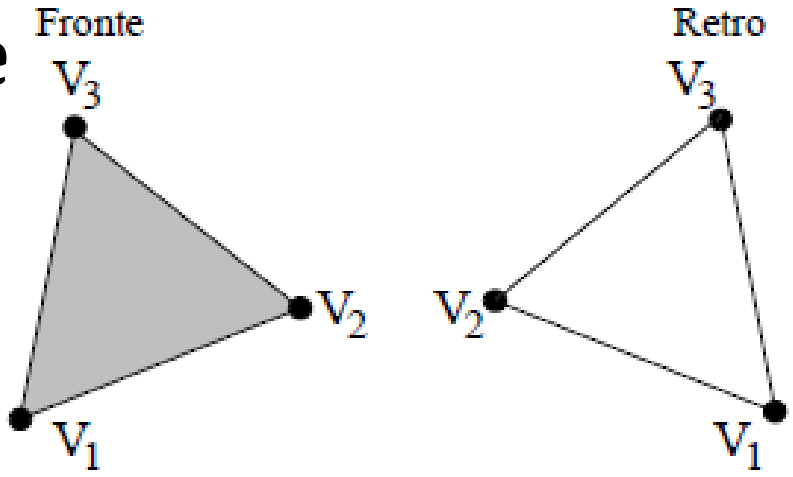
- La condizione di essere varietà o 2-manifold i traduce nei seguenti vinco sulla struttura:
  - uno spigolo appartiene al massimo a due triangoli (quelli eventuali che appartengono ad uno solo formano il bordo della maglia)
  - se due triangoli incidono sullo stesso vertice allora devono essere raggiungibili l'uno dall'altro attraverso un percorso tra triangoli adiacenti ovvero devono formare un ventaglio o un ombrello.
- Le mesh possono essere con bordo o senza bordo



Orientazione

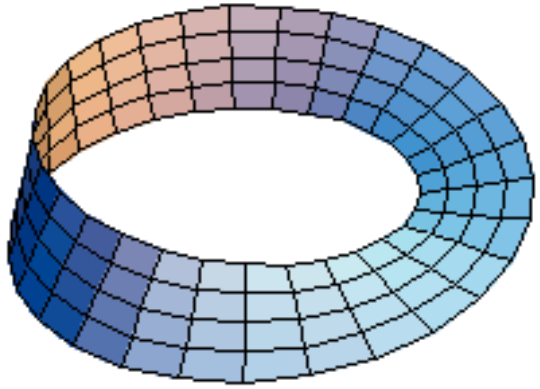
- Il bordo della maglia consiste di uno o più anelli (sequenza chiusa di spigoli) o loop.
- Se non esistono spigoli di bordo la maglia è chiusa (come quelle che rappresentano la superficie di una sfera).
- L'orientazione di una faccia è data dall'ordine ciclico (orario o antiorario) dei suoi vertici incidenti. L'orientazione determina il fronte ed il retro della faccia.





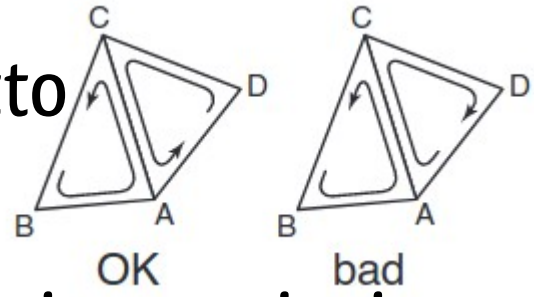
Mesh orientabili

- L'orientazione di due facce adiacenti è compatibile se i due vertici del loro spigolo in comune sono in ordine inverso. Vuol dire che l'orientazione non cambia attraversando lo spigolo in comune.
- La maglia si dice orientabile se esiste una scelta dell'orientazione delle facce che rende compatibili tutte le coppie di facce adiacenti.
  - Non tutte le mesh 2-manifold sono orientabili (es. anello di Moebius)

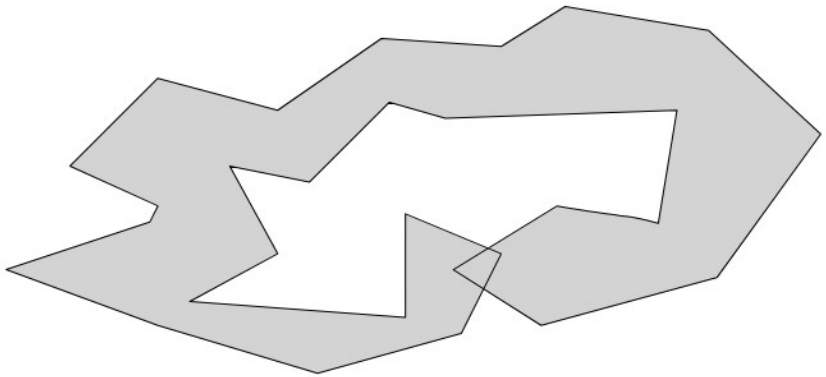


Correttezza delle mesh

- Una mesh poligonale manifold chiusa va bene per rappresentare gli oggetti 3D
- Divide l'interno dall'esterno
- Deve essere anche orientata in modo corretto

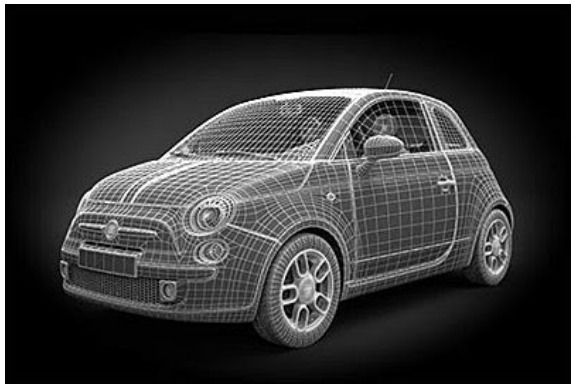


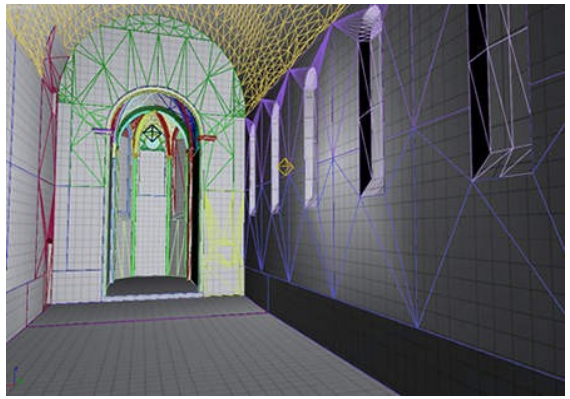
- Correttezza geometrica: non deve avere autointersezioni

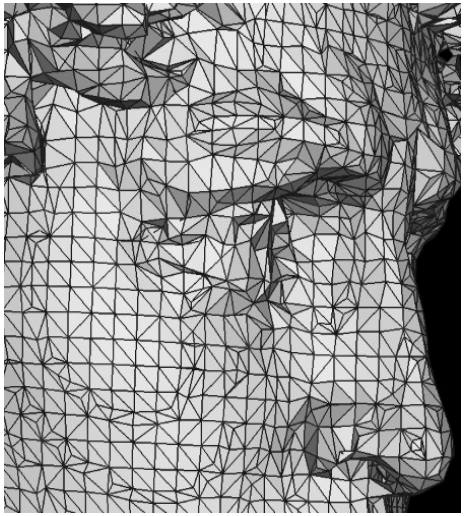


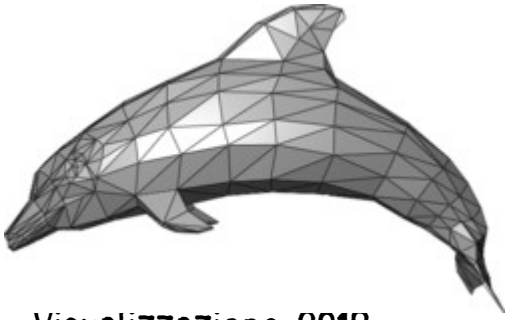
Mesh di triangoli

- Nella pratica sono il tipo di modello dominante
- Usato nella gran parte delle applicazioni interattive
  - Dato che il rendering è ottimizzato in hardware



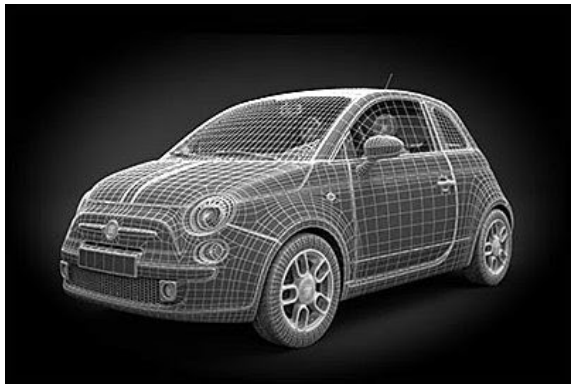


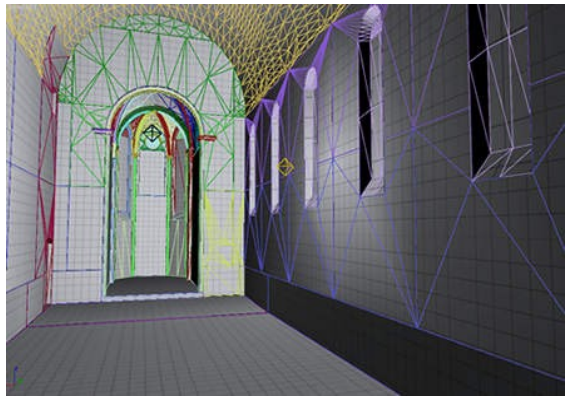


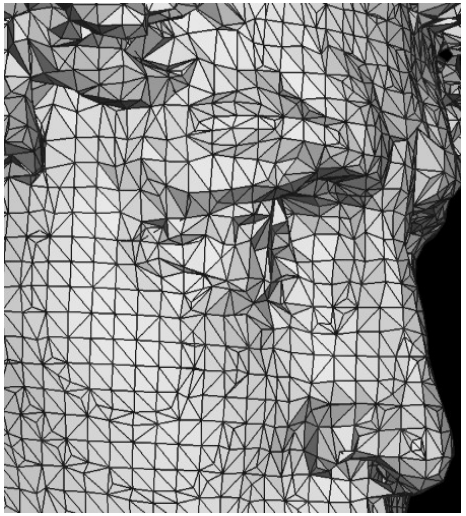


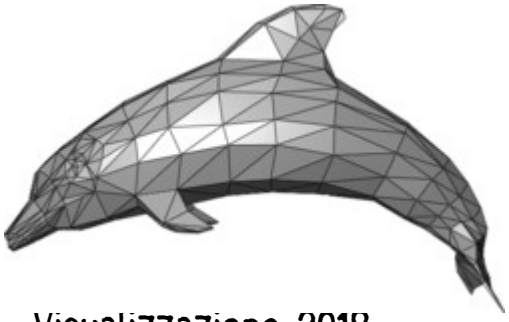
Mesh di triangoli

- Generate da modellazione CAD, acquisizione con scanner, ricostruzione da immagini (Computer Vision)
- Il numero di poligoni determina il dettaglio, ma il costo in memoria può essere notevole









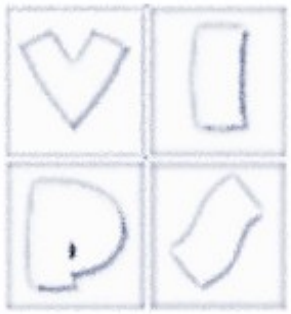
Mesh triangolare - Limiti

- Non è sempre semplice modellare le entità da rappresentare con triangoli...
  - Esempi:
    - Nuvole
    - Fiamme
    - Capelli, pelliccia

by Niniane Wang (non real time)

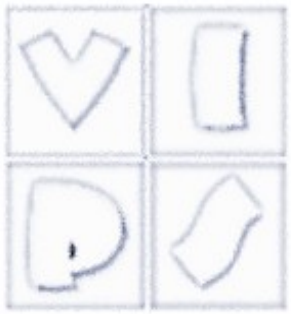
by N. Adabala Florida Uni (non real time)

by M. Turitzin and J. Jacobs Stanford Uni (real time!)



Mesh e rendering

- Per determinare l'effetto di una qualsiasi trasformazione affine su un oggetto (traslazione, rotazione, scalatura, composizioni varie di queste), basta applicare la trasformazione ai vertici (che sono punti); le informazioni connettive date dagli spigoli non cambiano in questo tipo di trasformazioni
  - Questo rende piuttosto semplice il rendering di oggetti descritti in termini di maglie poligonali. qualsiasi trasformazione viene eseguita sui vertici, cioè si tratta di applicare trasformazioni affini su punti
  - L'affermazione precedente è vera anche per la proiezione; per vedere come si proietta la forma di una maglia su un piano (l'immagine), basta seguire la proiezione dei vertici.



Mesh e rendering

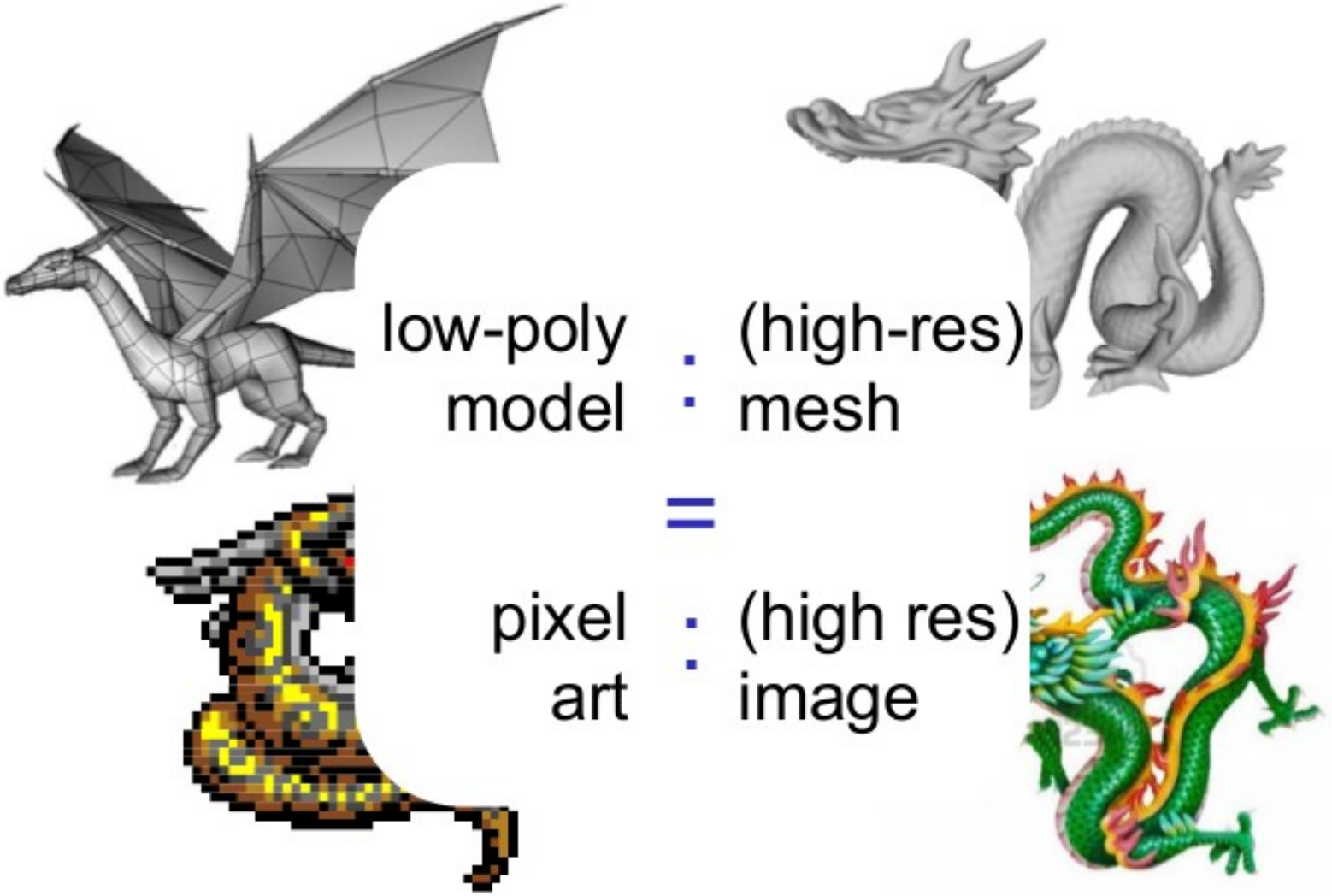
- La pipeline di rasterizzazione (che abbiamo visto essere il paradigma dominante e che ci interessa nella grafica interattiva) è stata creata e ottimizzata per le mesh di triangoli
- In essa la complessità computazionale del rendering è proporzionale al numero di triangoli!
- La risoluzione è critica

Caratteristiche delle mesh

- Manifoldness
- Risoluzione. Numero di triangoli/poligoni
  - Accuratezza nell'approssimazione
- Attenzione: se la geometria sottostante è semplice, i triangoli possono essere inutili
  - Algoritmi di semplificazione
- Curvature e normali
  - La loro qualità influirà poi sugli algoritmi di illuminazione

Immagini e mesh

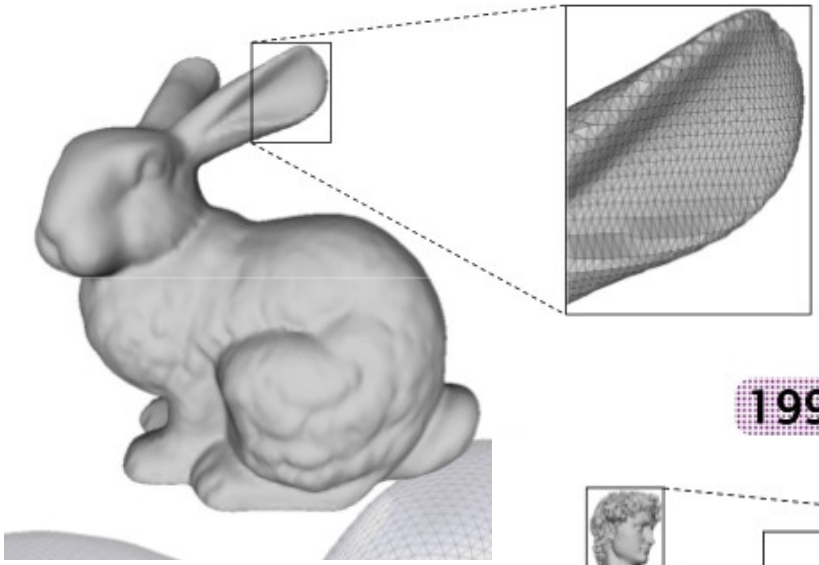
- Simile concetto di risoluzione



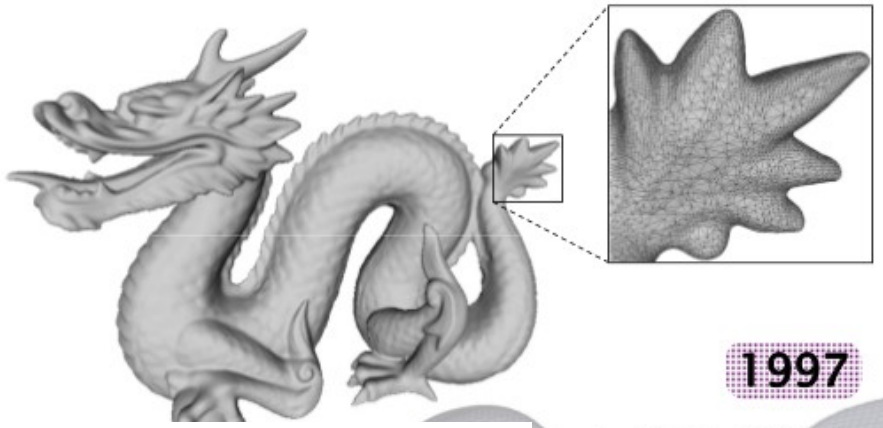
[M Fratarcangeli]

Risoluzione

- Incremento risoluzione nel tempo
  - Da. M.Fratarcangeli

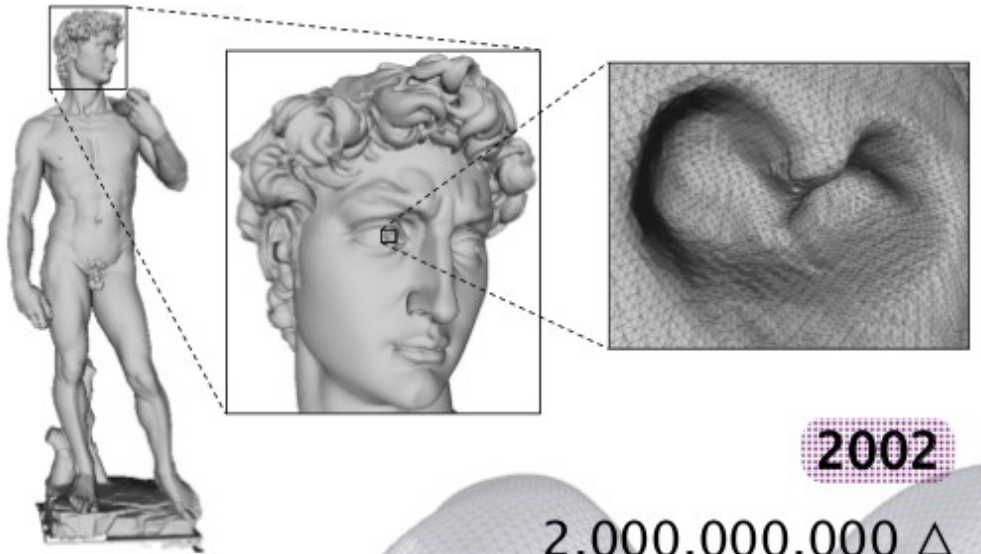


1994



1997

1.200.000 △

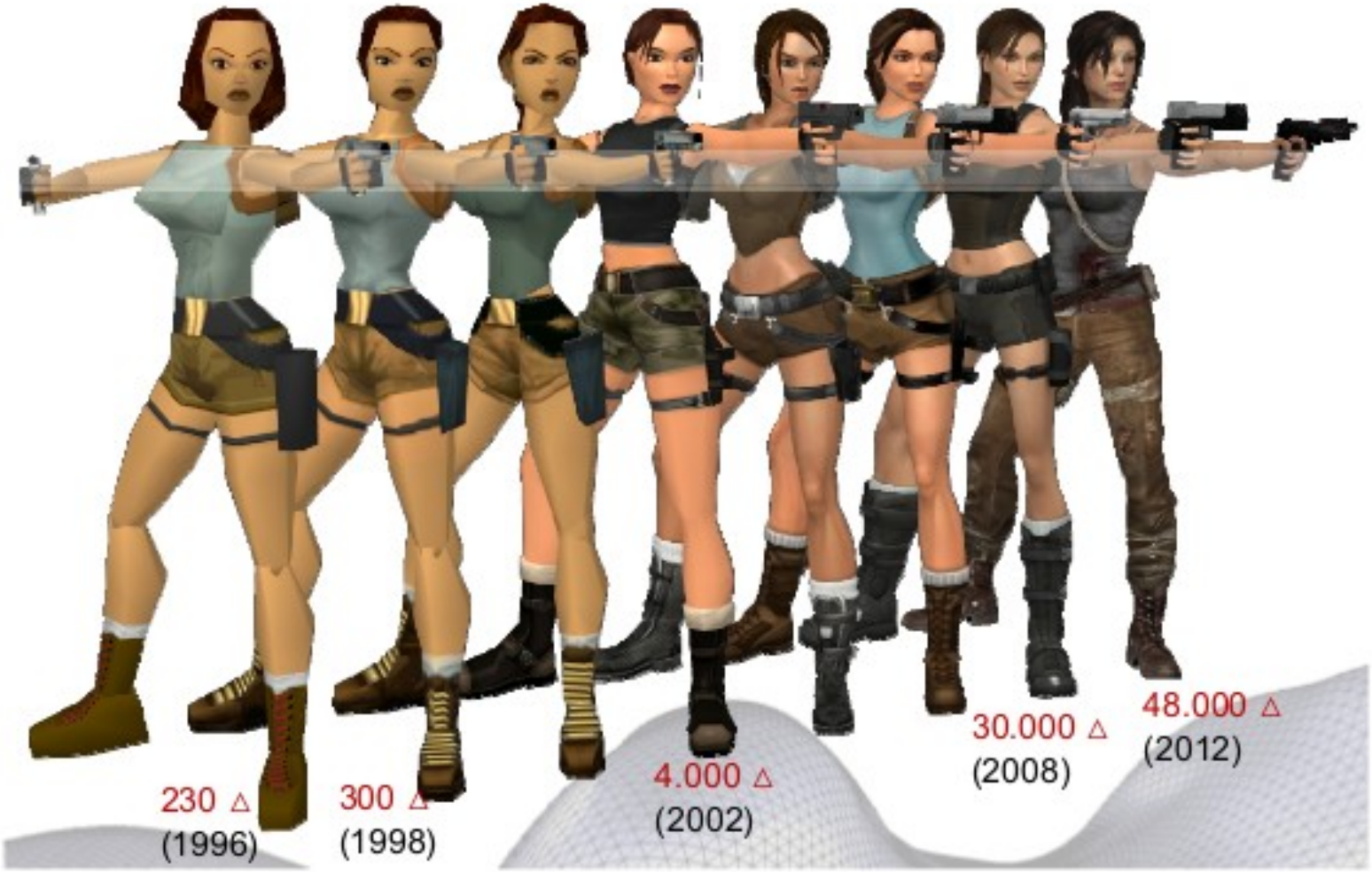


2002

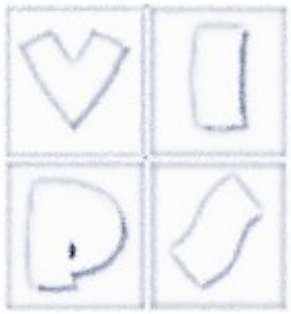
2.000.000.000 △

Evoluzione

- Modelli più dettagliati col progredire dell'hardware grafico nei videogames



[M Fratarcangeli]



Memorizzazione

- Alla creazione dei modelli vengono in genere generati nodi, connettività e attributi. Gli algoritmi di processing e rendering devono accedere in vario modo a tale informazione
- Diversi modi di rappresentare questa informazione
- Nei programmi applicativi sono quindi necessarie delle procedure per convertire una rappresentazione in un'altra
- Progettare ed implementare tali procedure è un ottimo modo per capire nel dettaglio le varie rappresentazioni utilizzate per descrivere maglie poligonali
- Nei disegni e negli esempi ci concentreremo sul caso di maglia triangolare, ma il discorso è valido in generale per tutti i poligoni convessi

Elementi base

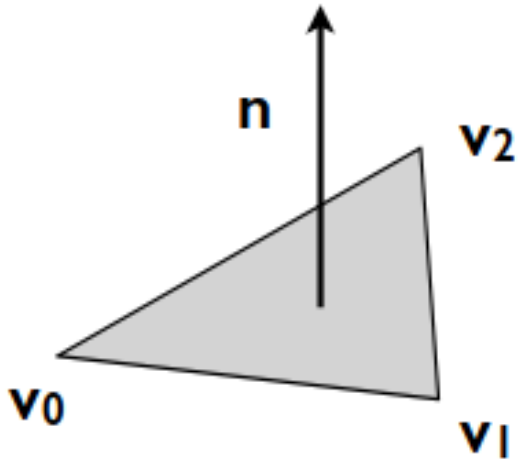
- **Vertici:** sono gli elementi 0 dimensionali e sono identificabili con punti nello spazio 3D (essenzialmente tre coordinate); alle volte può essere utile associare ai vertici altre caratteristiche oltre alla posizione (tipo il colore)
- **Spigoli:** sono elementi 1 dimensionali e rappresentano un segmento che unisce due vertici. Di solito non contengono altre informazioni.
- **Facce:** sono i poligoni bidimensionali, formati da un certo numero di spigoli e di vertici (dimostrare che sono in numero uguale). I vertici o gli spigoli si usano per identificare la faccia; possono contenere altre informazioni (tipo il colore)

Mesh di triangoli: triangoli

- Normali: si ricavano semplicemente con il prodotto vettore di vettori differenza di vertici
- Così si ottiene anche l'area
- Orientazione frontale se lista vertici antioraria

$$\mathbf{n} = \frac{(\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)}{|(\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)|}$$

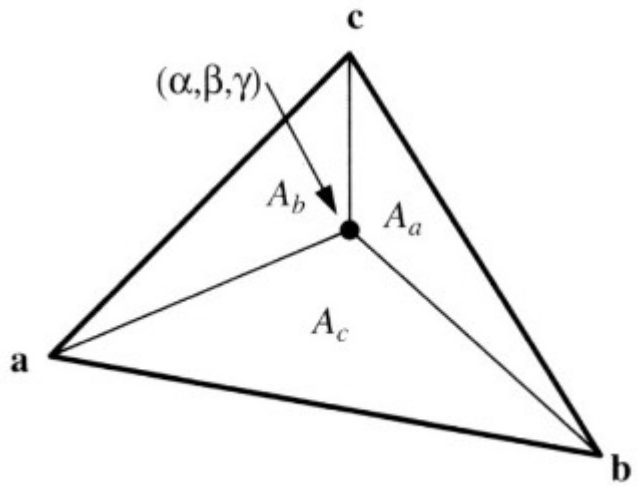
$$A = \frac{|(\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)|}{2}$$



Coordinate baricentriche

- I punti dei triangoli possono essere espressi come combinazione lineare (affine) delle coordinate dei vertici
- I coefficienti sono detti coordinate baricentriche
- Si usano per interpolare gli attributi

$$\mathbf{p} = w_0\mathbf{v}_0 + w_1\mathbf{v}_1 + w_2\mathbf{v}_2$$
$$w_0 + w_1 + w_2 = 1$$
$$w_0 \geq 0 \quad w_1 \geq 0 \quad w_2 \geq 0$$



Coordinate baricentriche

- I punti dei triangoli possono essere espressi come combinazione lineare (affine) delle coordinate dei vertici
- I coefficienti sono detti coordinate baricentriche
- Si usano per interpolare gli attributi

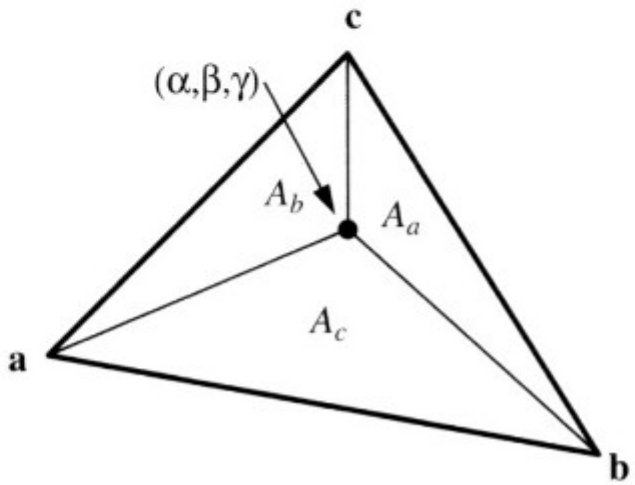
Algebraic view: linear combination

$$w_0 = 1 - w_1 - w_2 \Rightarrow$$

$$\mathbf{p}(w_1, w_2) = \mathbf{v}_0 + w_1(\mathbf{v}_1 - \mathbf{v}_0) + w_2(\mathbf{v}_2 - \mathbf{v}_0)$$

Geometric view: relative areas

$$w_0 = A_0/A \quad w_1 = A_1/A \quad w_2 = A_2/A$$

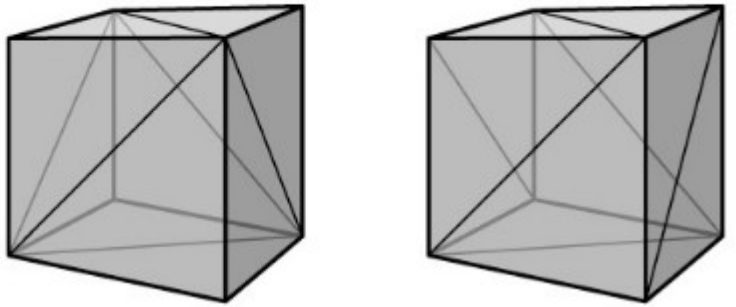


[Shirley 2000]

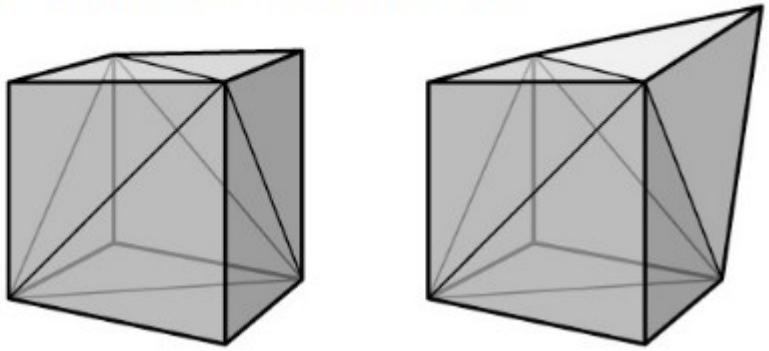
Geometria e topologia

- Geometria di una mesh: dove sono le superfici nello spazio
- Topologia: come sono connessi

same geometry, different mesh topology:

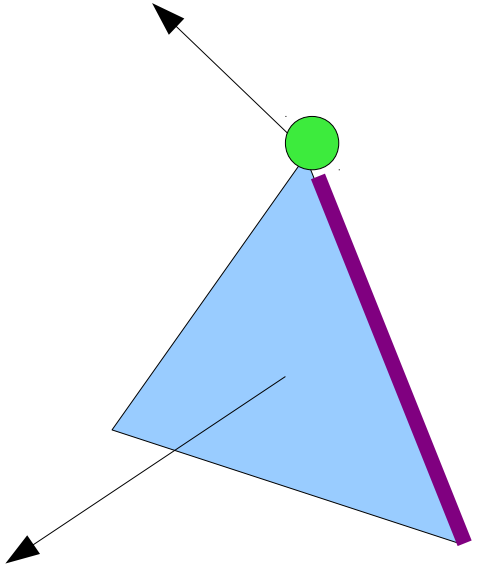


same mesh topology, different geometry:



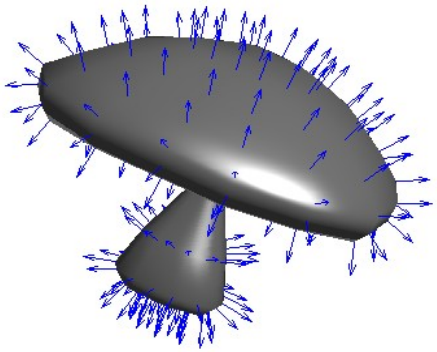
Mesh triangolare - Attributi

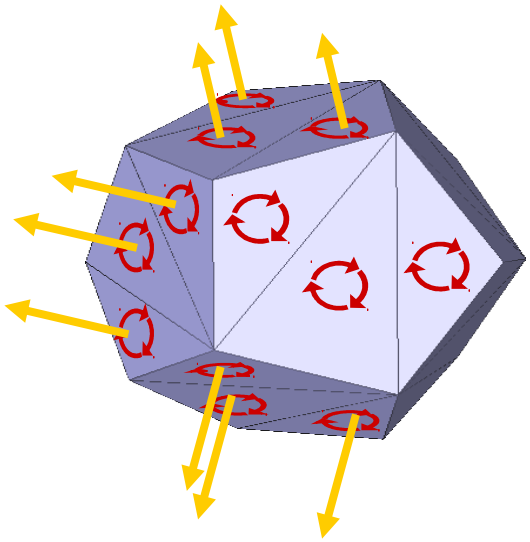
- Per rappresentare gli oggetti con le proprietà fisiche relativa a colore e riflessione della luce, devo abbinare alla geometria dei valori di proprietà (attributi)
- Posso definirli:
  - per vertice
    - esplicito un attributo per ogni vertice
  - per faccia
    - esplicito un attributo per ogni faccia
  - per wedge (vertice di faccia)
    - esplicito tre attributi per ogni faccia
- Attributi più comuni:
  - Colore
  - Normali (versori perpendicolari)
  - coordinate texture (vedremo)



Attributi

- **Normali:** è fondamentale sapere quale è l'esterno della superficie e quale l'interno, e qual è l'orientazione locale della superficie; a tal scopo si associa spesso ad una maglia poligonale anche l'informazione sulla normale uscente.
- La normale **n** ad una faccia è data dal prodotto vettore di due suoi spigoli consecutivi non collineari
  - attenti al verso: la normale è uscente dal fronte della faccia
  - Per un triangolo (V_1,V_2,V_3) si ha: $\mathbf{n} = (V_3-V_2)\mathbf{x}(V_1-V_2)$.

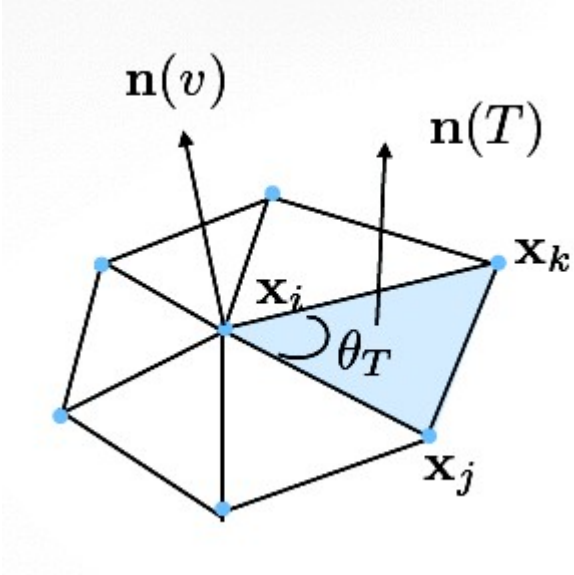




Normali sui vertici

- Posso fare una media (pesata) delle normali delle facce. Pesi?
  - Uniformi? (1)
  - Area? |T|
  - Angolo?

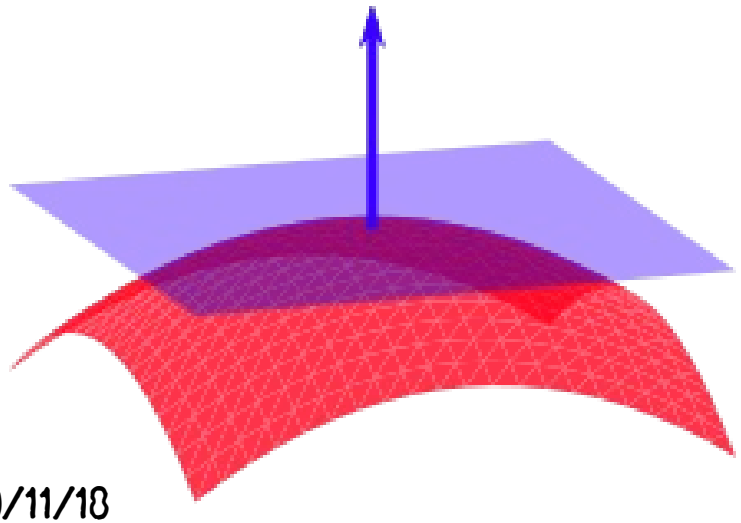
$$\mathbf{n}(T) = \frac{(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

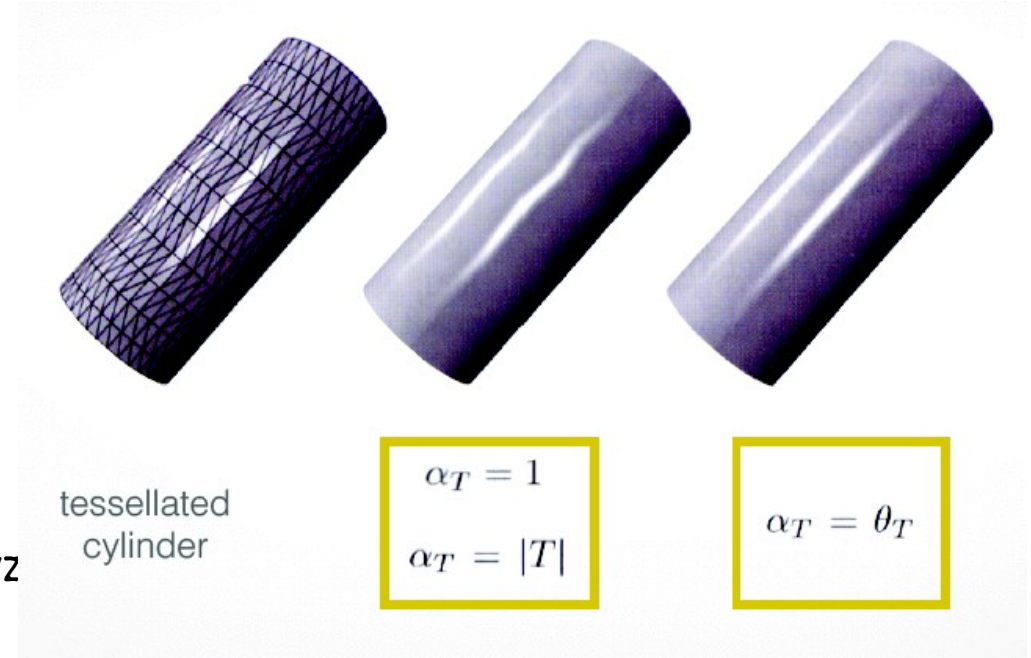


$$\mathbf{n}(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T) \right\|}.$$

Mesh e approssimazione

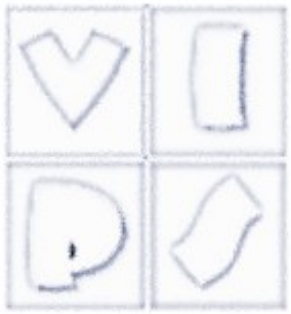
- Ricordiamo che con le mesh vogliamo in genere approssimare superfici lisce
- Quindi sui vertici (o sulle facce) vorremmo approssimare la geometria differenziale (es. calcolo normali, piani tangenti, curvature)
- Il calcolo delle quantità differenziali è in genere influenzato dalla triangolazione
  - Triangoli





Coordinate texture

- Un altra cosa utile nella pipeline di rasterizzazione sarà memorizzare coordinate che mappino i vertici di un modello sulle coordinate di un'immagine, per poter "appiccicare" questa immagine e decorare l'oggetto senza usare troppi triangoli ("texture mapping")

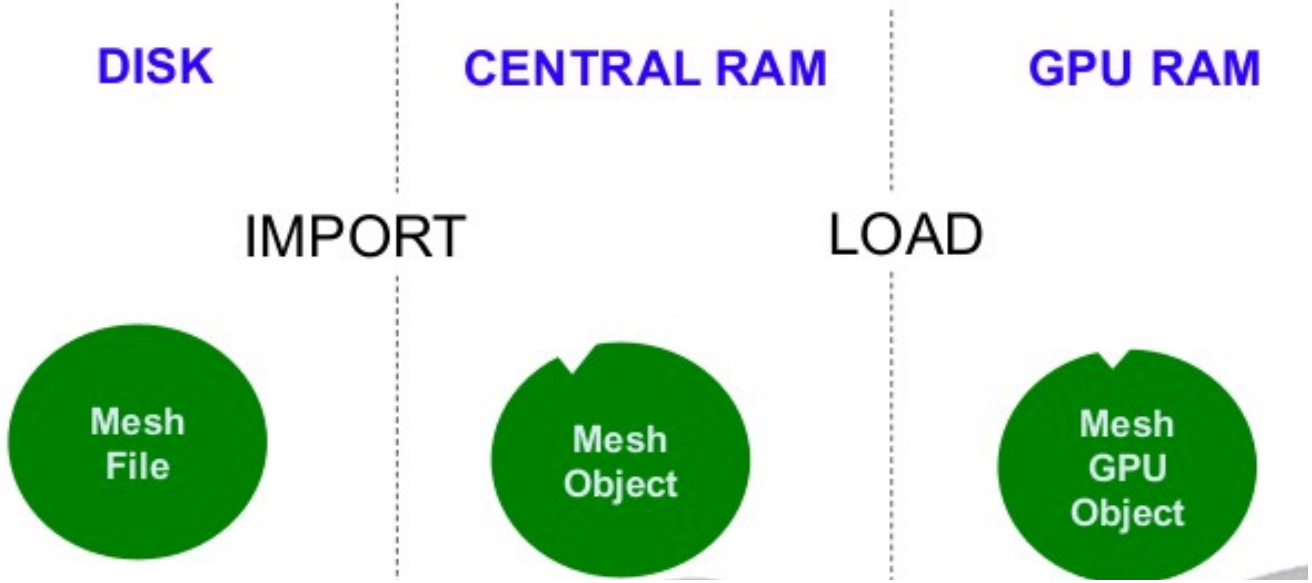


Struttura della mesh

- I vertici danno informazioni di tipo posizionale, gli spigoli informazioni di tipo connettivo (non c'è informazione spaziale)
- Gli spigoli connettono i vertici, permettendo di introdurre un concetto di “vicinanza” tra vertici e dando le informazioni di tipo topologico (ovvero definiscono un grafo)
- Le facce sono determinate una volta dati i vertici e gli spigoli, quindi non introducono nulla
  - Al più possono avere associati attributi, anche se è raro

Strutture dati per le mesh

- Ci sono vari modi di conservare le informazioni sui modelli e utilizzarle nei programmi
- Nelle applicazioni interattive le mesh dovranno essere caricate da file in memoria e poi caricate dalla RAM alla GPU
- In GPU devono essere array di vertici con connettività
- Ma in memoria le strutture dati possono essere diverse

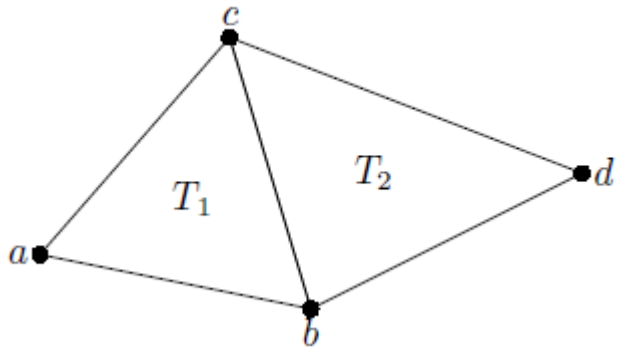


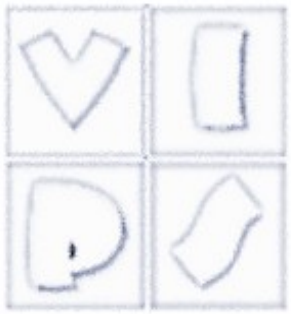
Strutture semplici

- **Lista di triangoli** Potrei specificare tutte le facce della maglia come terne di triplette di coordinate cartesiane
  - Spreco di memoria: duplico le coordinate. Come trovo i vicini?
- Meglio usare una struttura **indicizzata**, con la lista dei vertici e la lista delle facce con i puntatori ai vertici (indici)
  - Struttura normalmente usate in openGL per il rendering
  - Separazione geometria/topologia

```
typedef struct {
float v1[3];
float v2[3];
float v3[3];}
faccia;
```

```
typedef struct {
float x,y,z;
} vertice;
typedef struct {
vertice* v1,v2,v3;
} faccia;
```

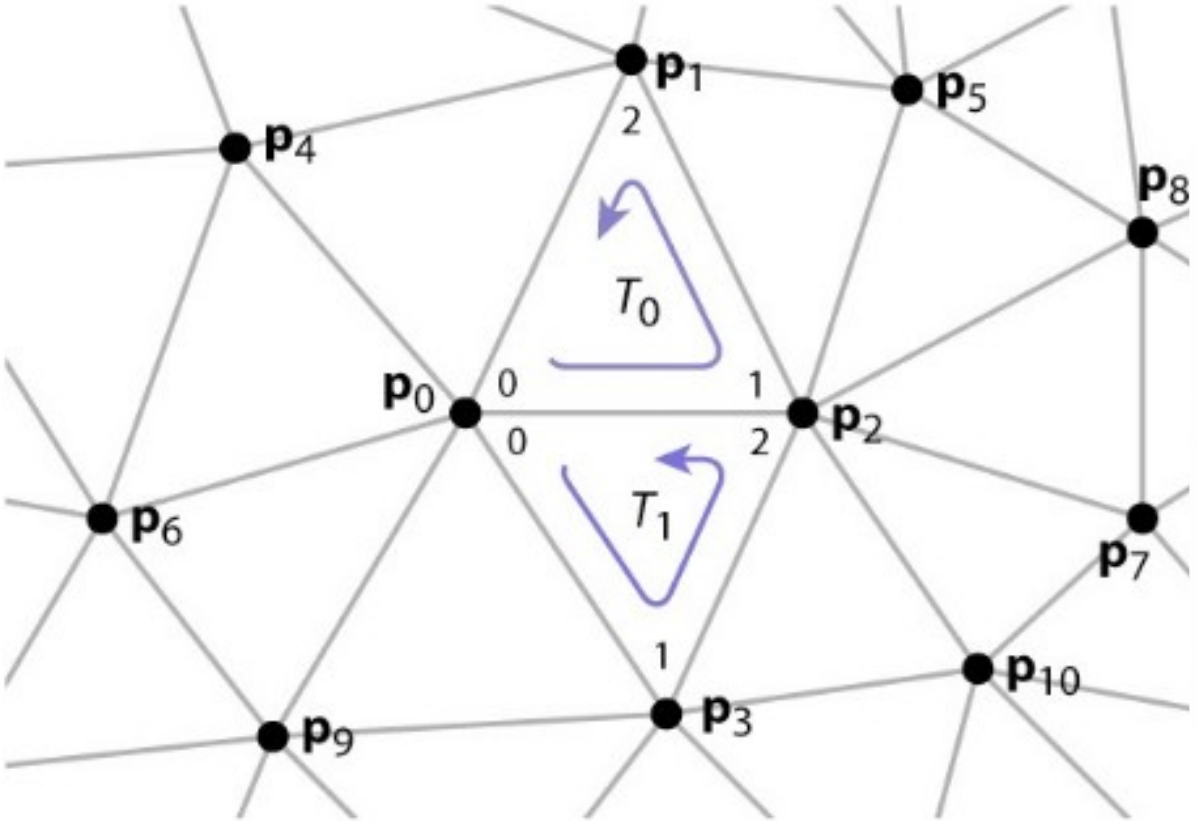


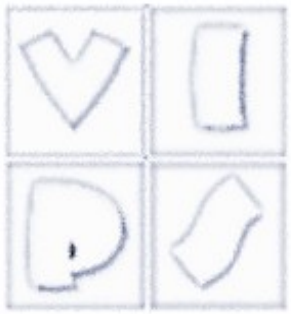


Mesh indicizzata

verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
	x_2, y_2, z_2
	x_3, y_3, z_3
	⋮

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
	⋮





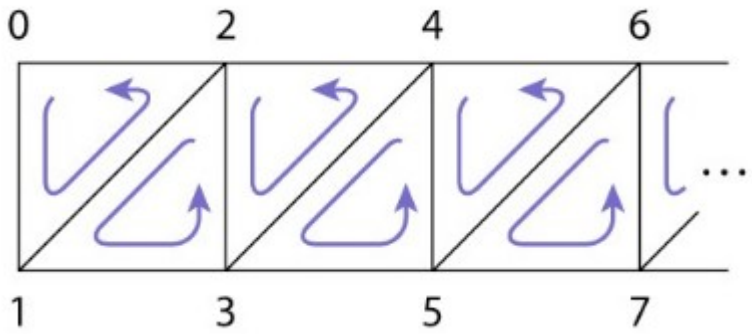
Mesh indicizzata

- Esempio (memorizzazione indicizzata nei file OFF)

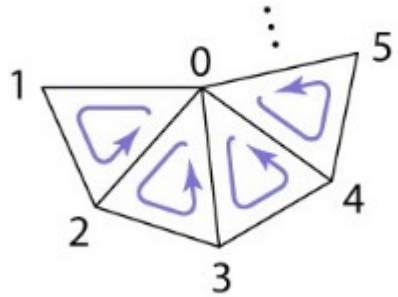
```
OFF
4  4  0
-1  -1  -1
1  1  -1
1  -1  1
-1  1  1
3  1  2  3
3  1  0  2
3  3  2  0
3  0  1  3
```

Strip e fans

- Compressione di parti di mesh eliminando ripetizioni di indici
- Triangle strip

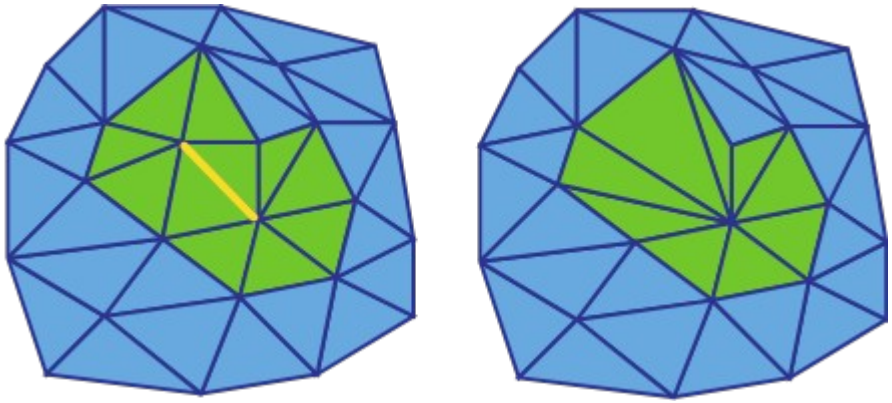


- Triangle fan

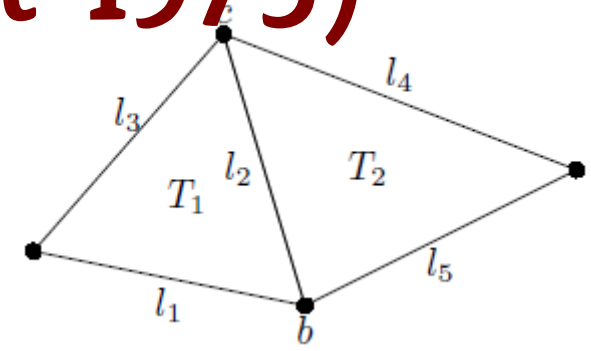


Strutture ottimizzate per ricerca di incidenza

- La struttura dati vista non è ottimale se devo cercare di trovare i "vicini" di punti, spigoli e triangoli
  - Cioè fare ricerche di incidenza (quali facce incidono su un vertice)… o cercare adiacenza di triangoli
- Soluzione: creare strutture dati ad hoc per rendere veloci le ricerche
  - Soluzione tipica: aggiungere nelle strutture dati degli elementi puntatori ai vicini
  - Costo: un po' di ridondanza



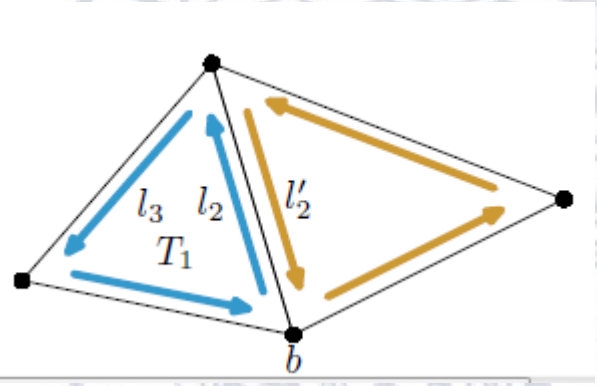
Winged edge (Baugmart 1975)



- Si aggiungono dei puntatori allo spigolo per rendere più semplice l'analisi delle incidenze.
- L'elemento base è lo spigolo (edge) con le sue due facce incidenti (wings)
  - Lo spigolo l2 contiene un puntatore ai due vertici su cui incide (b; c), alle due facce su cui incide (T_1, T_2) ed ai due spigoli uscenti da ciascun vertice
  - Un vertice contiene un puntatore ad uno degli spigoli che incide su di esso, più le coordinate (ed altro)
  - La faccia contiene un puntatore ad uno degli spigoli che vi incide (ed altro).
- La struttura assume che ogni spigolo non di bordo abbia due facce incidenti (manifold)

```
typedef struct {
we_vertice* v_ini,
v_fin;
we_spigolo* vi_sin,
vi_dstr;
we_spigolo* vf_sin,
vf_dstr;
we_faccia* f_sin,
f_dstr;
} we_spigolo;
typedef struct {
float x, y, z;
we_spigolo* spigolo;
} we_vertice;
typedef struct {
we_spigolo* spigolo;
} we_faccia;
```

Half edge



- Ogni spigolo viene diviso in due spigoli orientati in modo opposto
  - Ciascun mezzo spigolo contiene un puntatore al vertice iniziale, alla faccia a cui "appartiene", al mezzo spigolo successivo (seguendo l'ordinamento) ed al mezzo spigolo gemello
  - Ogni vertice, oltre alle coordinate (e attributi) contiene un puntatore ad uno qualsiasi dei mezzi spigoli che esce da tale vertice
  - Ogni faccia contiene uno dei suoi mezzi spigoli (oltre ad altre caratteristiche quali, ad esempio, la normale)
- Efficiente per le ricerche ed elegante

```
typedef struct {
he_vertice*
origine;
he_spigolo*
gemello;
he_faccia* faccia;
he_spigolo*
successivo;
} he_spigolo;
typedef struct {
float x, y, z;
he_spigolo*
spigolo;
} he_vertice;
typedef struct {
he_spigolo*
spigolo;
} he_faccia;
```

Note

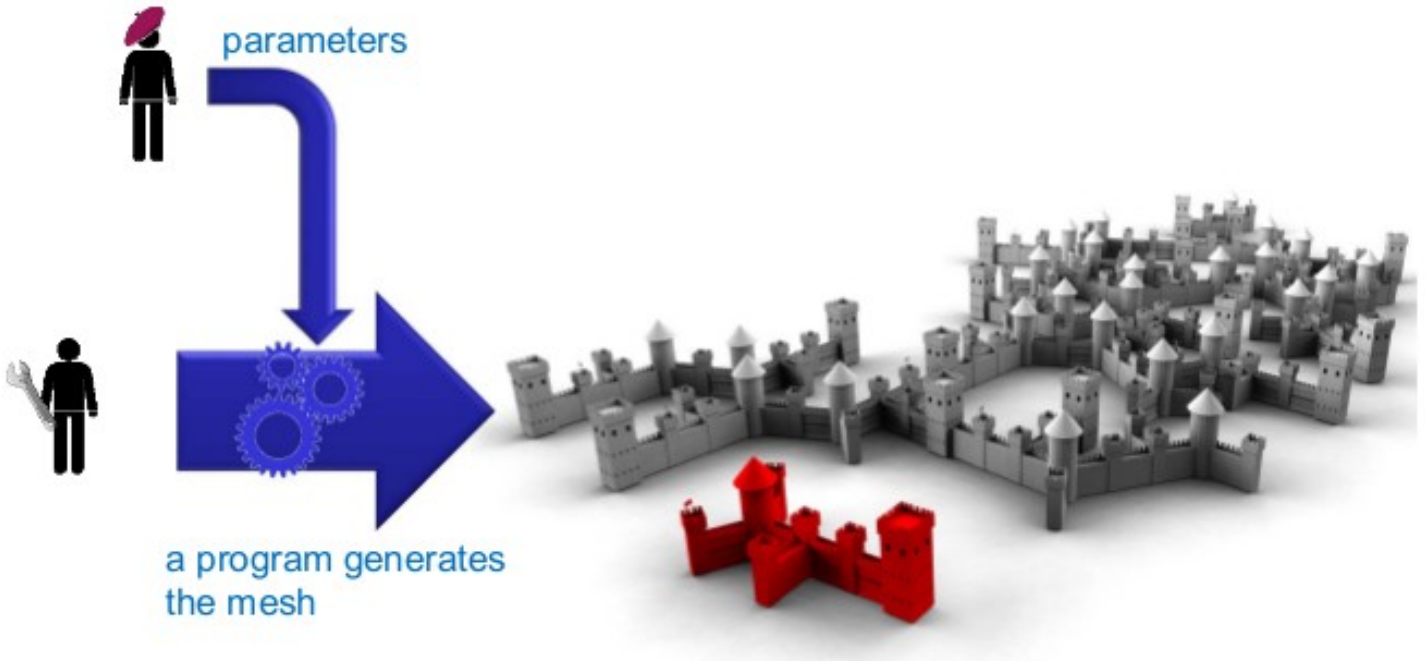
- La stessa applicazione grafica può far uso di più di una struttura dati
- La rappresentazione con la lista di vertici essendo semplice e leggera è tipicamente usata come formato per i file contenenti la geometria di oggetti. E si mappa direttamente con la struttura richiesta dall'hardware grafico
- Le applicazioni grafiche che devono fare processing di mesh (cioè modificare interattivamente i modelli) in genere convertono i modelli anche in struttura dati più utile ai fini algoritmici (per esempio la half-edge)

Formati file

- Tantissimi in diversi contesti
  - 3DS - 3D Studio Max file format
  - OBJ - Another file format for 3D objects
  - 3DX - Rinoceros file format
  - BLEND - Blender file format
  - DAE - COLLADA
  - FBX - Autodesk interchange file format
  - X - Direct X object
  - SMD - good for animations (by Valve)
  - MD3 - quake 3 vertex animations
  - SKP - Google sketch up
  - KMZ - Google Earth model
  - OFF - A general 3D mesh Object File Format
  - OOGL - Object Oriented Graphics Library
  - PLG - Used by REND386/AVRIL
  - POV - "persistence of vision" ray-tracer
  - QD3D - Apple's QuickDraw 3D Metafile format
  - DXF - (exchange format, Autodesk's AutoCAD)
  - VIZ - Used by Division's dVS/dVISE
  - FLT - MulitGen Inc.'s OpenFlight format
  - STL
  - IGES Initial Graphics Exchange Specification
  - X3D – tentative next VRML
  - PLY – introduced by Cyberware – range scan data
  - IV Open Inventor File Format Info Renderman
  - LWO, LWB & LWS Lightwave 3D file formats
  - MGF Materials and Geometry Format
  - MSDL Manchester Scene Description Language
  - 3DML by Flatland inc.
  - C4D – Cinema 4D file format
  - ZModeler File format
  - etc, etc, etc...

Modellazione procedurale

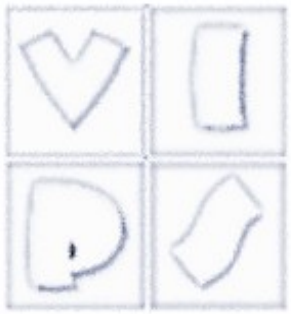
- Si possono creare modelli in modo procedurale
  - Da una descrizione parametrica



[M Fratarcangeli]

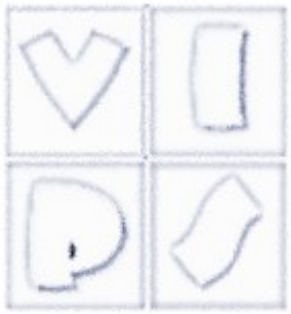
3D scanning

- Generalmente catturano nuvole di punti
- Trasformate in mesh 3D da software
- Anche altissima risoluzione



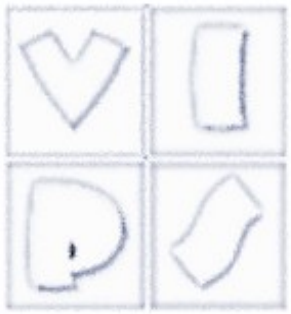
Modellazione da scanner

- Quindi servono algoritmi
  - Meshing: creazione delle triangolazioni dalle nuvole di punti
  - Semplificazione: riduzione del numero di punti dei modelli per rendere possibile l'uso in grafica
  - Controllo sulla qualità
  - Creazione di attributi sulle mesh



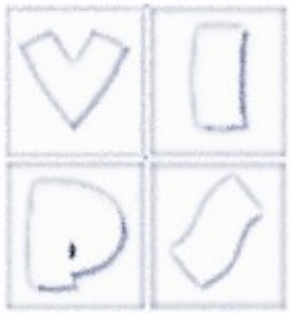
Mesh processing

- Importante ramo della ricerca in grafica al calcolatore
- Molti algoritmi utilizzabili interattivamente o offline
- Librerie e strumenti molto diffusi
- Editor
  - 3DStudio Max, Blender, Maya
- Tool per processing (anche di 3D scan)
  - Meshlab
- Librerie
  - CGAL, VCG, ...



Algoritmi

- Generazione mesh da nuvole di punti
  - Ball pivoting, Poisson, ecc.
- Generazione mesh da descrizioni volumetriche
  - Marching cubes, ecc.
- Pulizia
- Rimozione buchi e parti non manifold
- Schemi di suddivisione
- Semplificazione

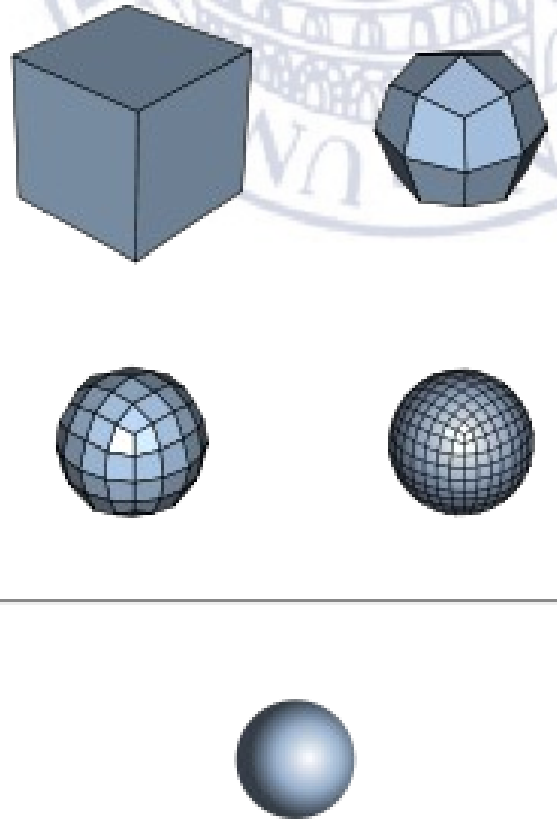


Pulizia, correttezza

- Le strutture dati delle mesh non garantiscono correttezza geometrica e topologica
- Se ci serve occorre correggere le mesh per ottenerla
- Ricerca e rimozione vertici ed edges non-manifold
- Ricerca e rimozione vertici duplicati, facce duplicate
- Ricerca buchi e eliminazione
- Ricerca autointersezioni ed eliminazione
- Non sempre banale/risolvibile

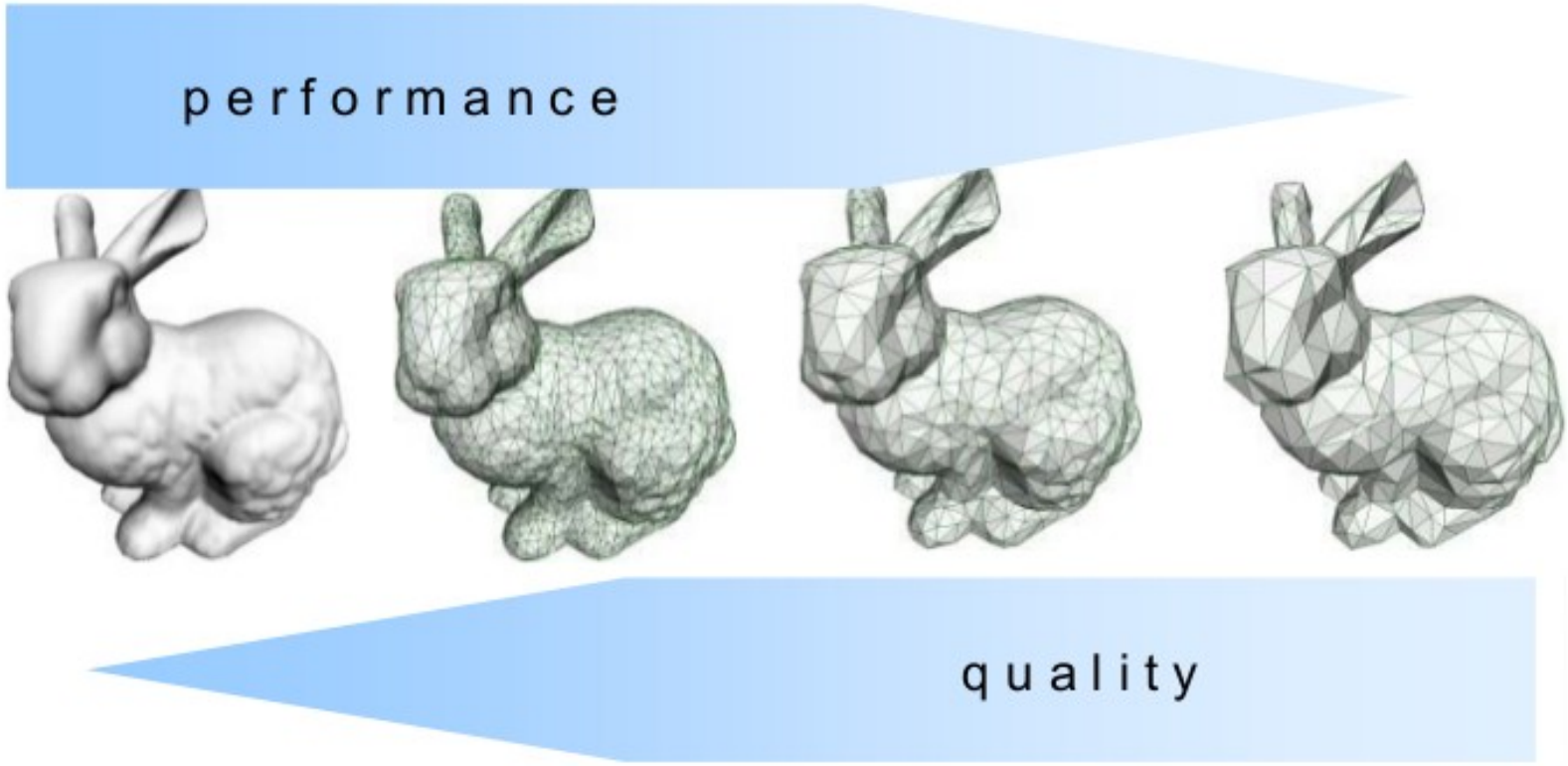
Schemi di suddivisione

- Una tecnica di modellazione che permette di generare mesh con molti poligoni, anche smooth, da modelli di base molto schematici
- Ricorsivamente si suddividono i poligoni
- Si può poi procedere a un raffinamento delle posizioni dei vertici con una regola (es. una funzione interpolante)
- La suddivisione tende a una superficie limite
- Può codificare in modo compatto superficie liscia (alternativa a nurbs, ecc)



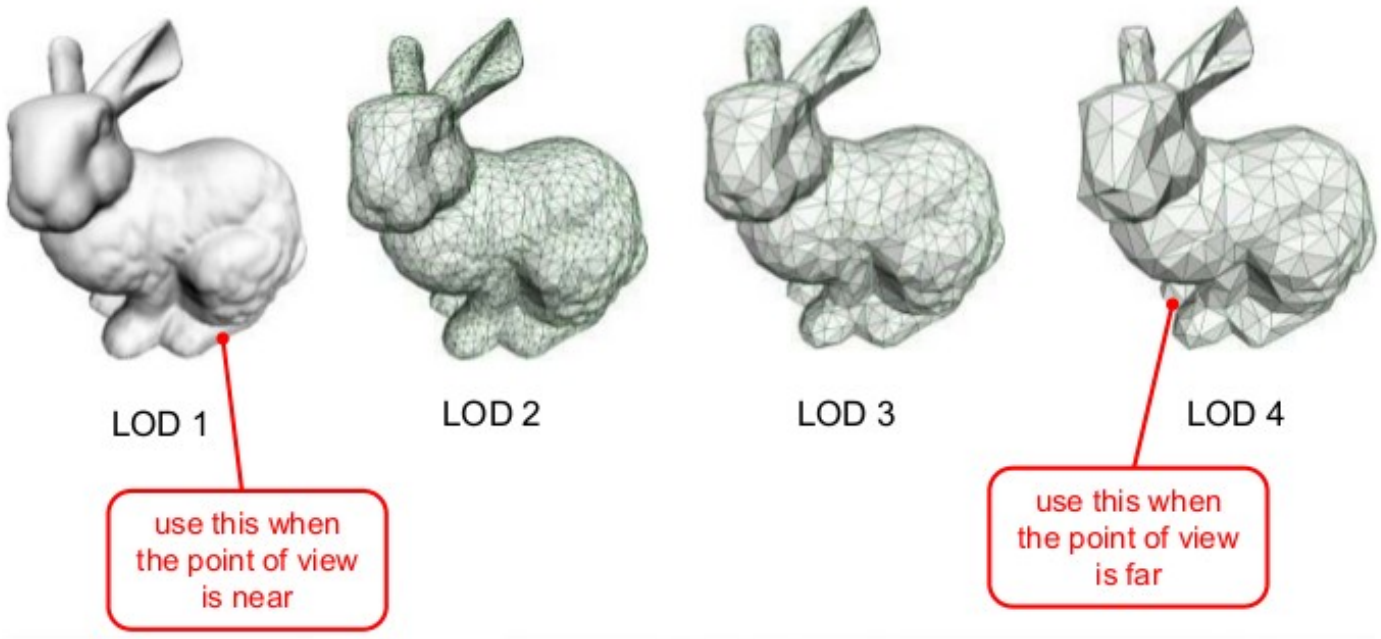
Semplificazione

- Algoritmi per semplificare:
  - Es quadric edge collapse
  - Idea: elimino iterativamente edge che cambiano meno la geometria
    - Ottengo modelli con meno triangoli che approssimano l'originale



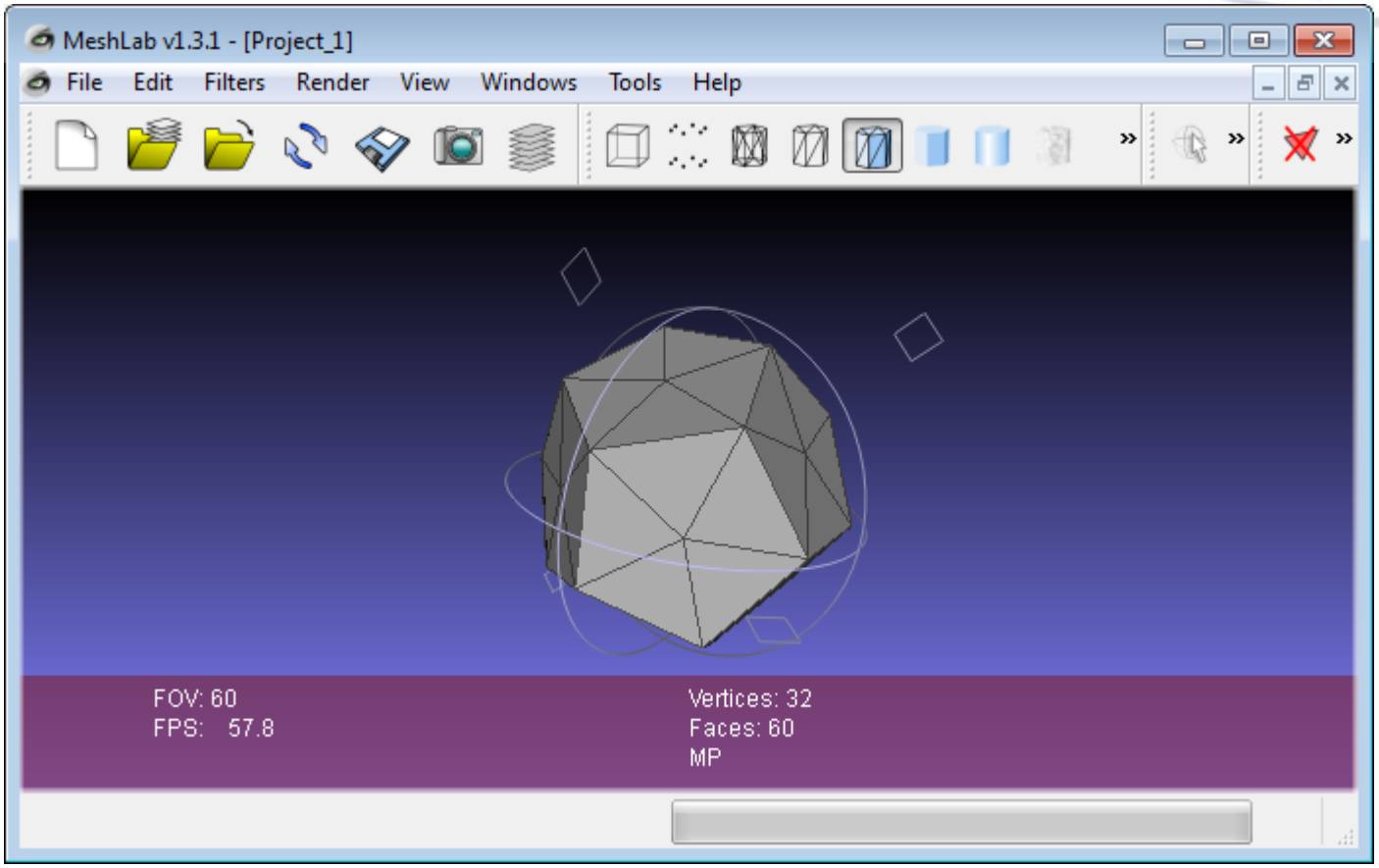
Semplificazione e multirisoluzione

- A volte può essere utile avere modelli a vario livello di dettaglio da utilizzare selettivamente nel renedring



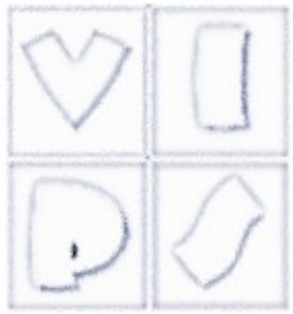
Meshlab

- Strumento open source ottimo per il processing di mesh con algoritmi di analisi e processing moderni
  - http://meshlab.sourceforge.net/

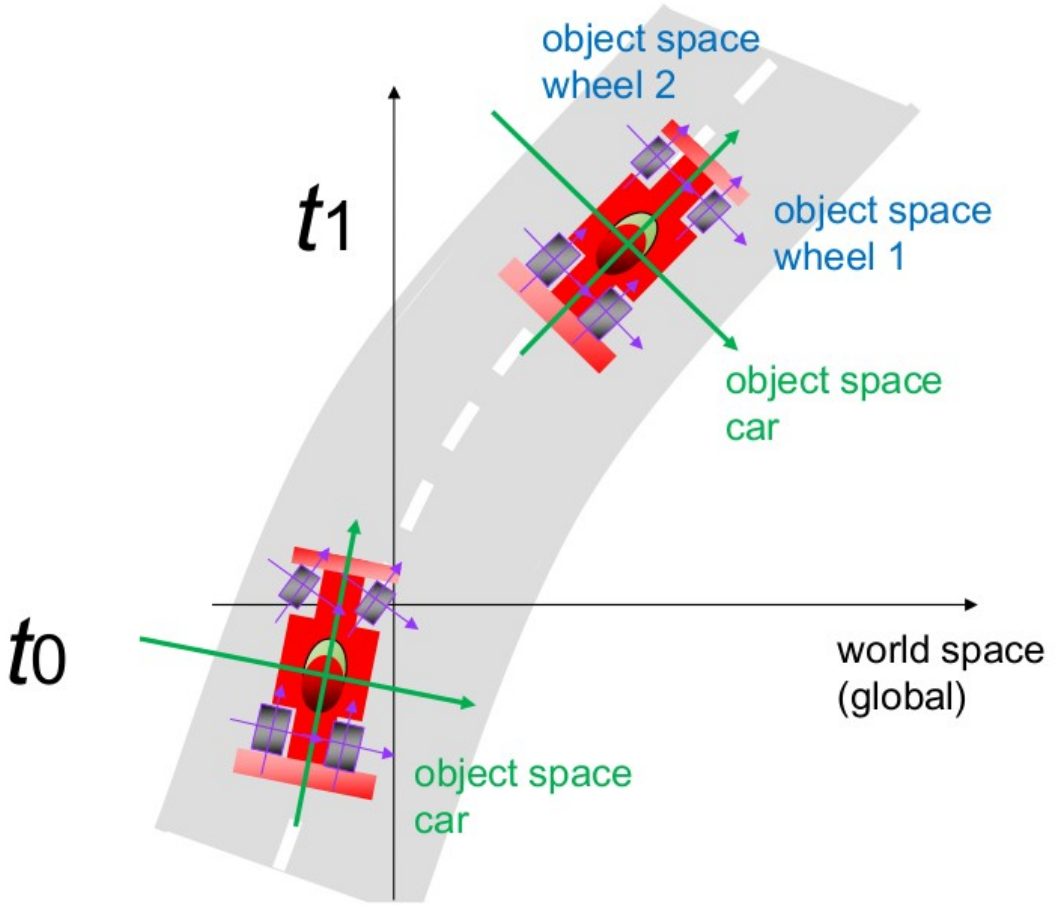


Modelli composti/scene

- Nelle applicazioni grafiche interattive i modelli interagiscono e quindi si muovono, in maniera programmata oppure in maniera interattiva
- Il moto rigido di singoli oggetti sappiamo modellarlo e quindi programmarlo
- Per oggetti composti di parti rigide la soluzione è quella di creare modelli gerarchici (es. automobile, robot)
  - Le ruote sono figlie del telaio: compongono il moto rigido del telaio più il proprio moto individuale
  - Il braccio del robot compone il moto dell'avambraccio con il suo individuale

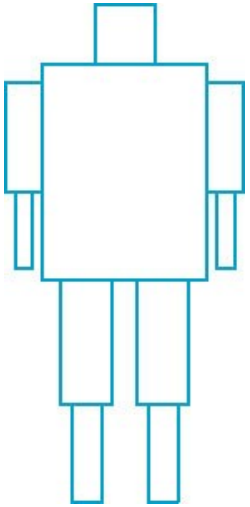


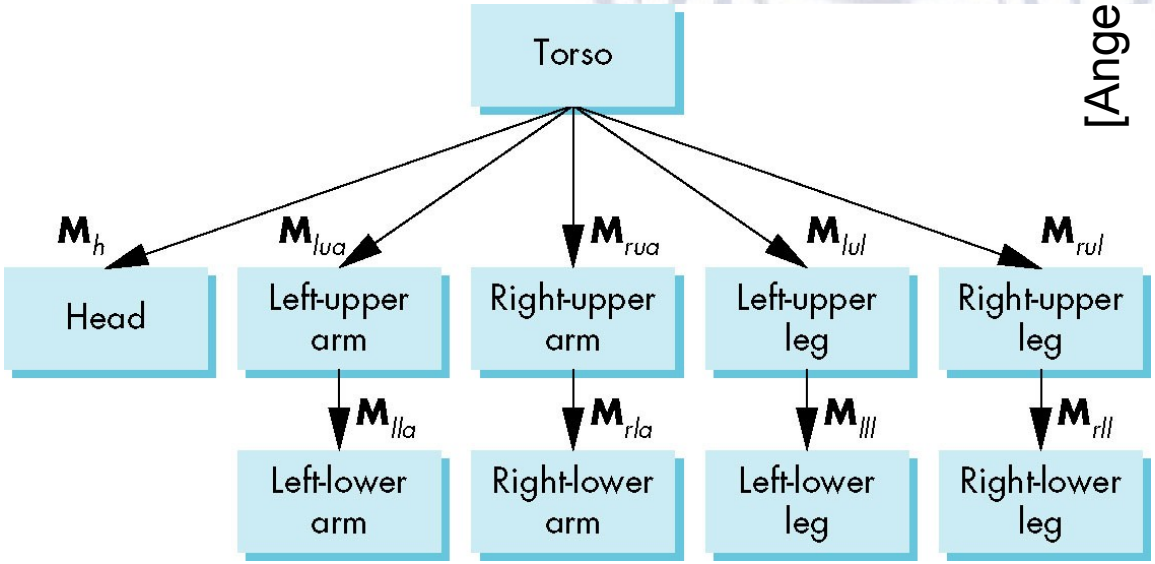
Esempio


object space
wheel 2
object space
wheel 1
object space
car
t1
t0
world space
(global)
object space
car

Modello gerarchico

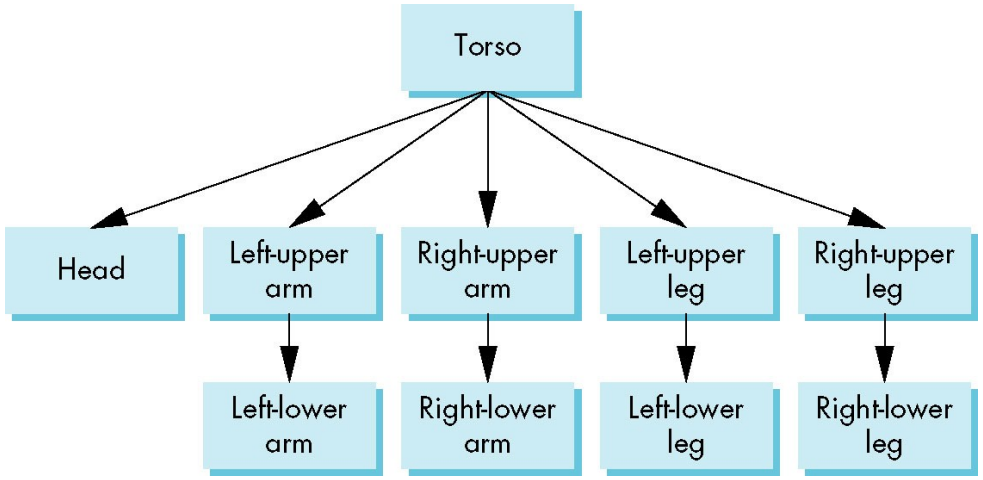
- Il moto delle "foglie" è la composizione delle trasformazioni geometriche date dalle matrici che si incontrano nella discesa del grafo

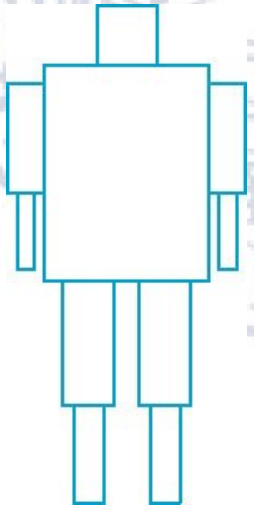




[Angel]

Costruire un modello

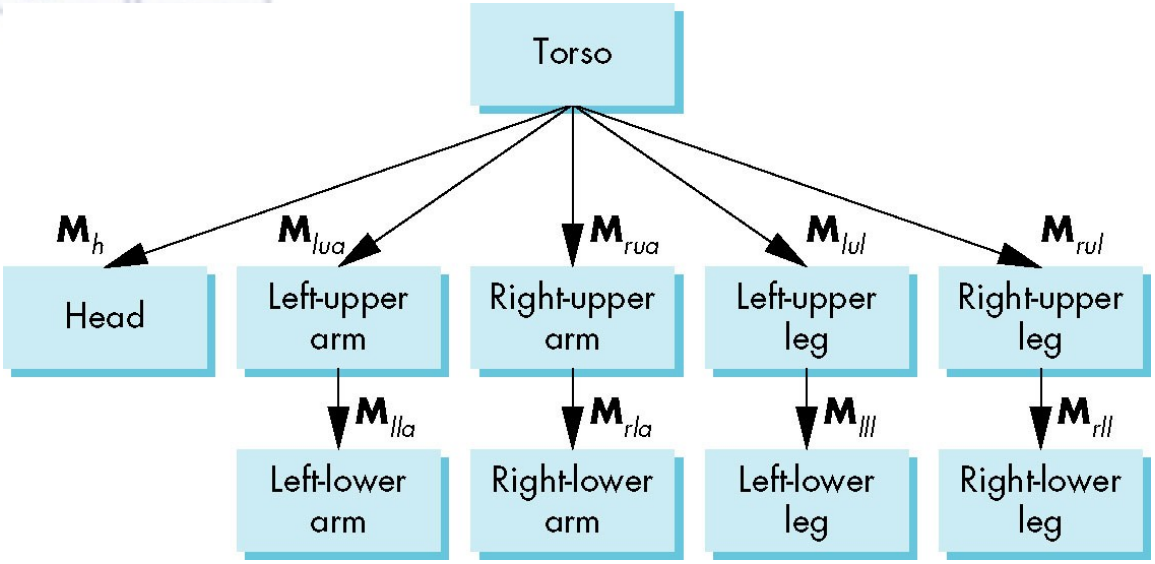


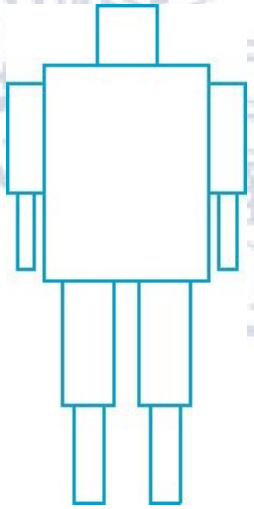


[Angel]

- Supponiamo di creare facilmente la geometria delle parti (es. cilindri)
- Vogliamo creare strutture per accedere alle parti
  - torso(), left_upper_arm()
- Dentro matrici che descrivono posizione del nodo rispetto al genitore

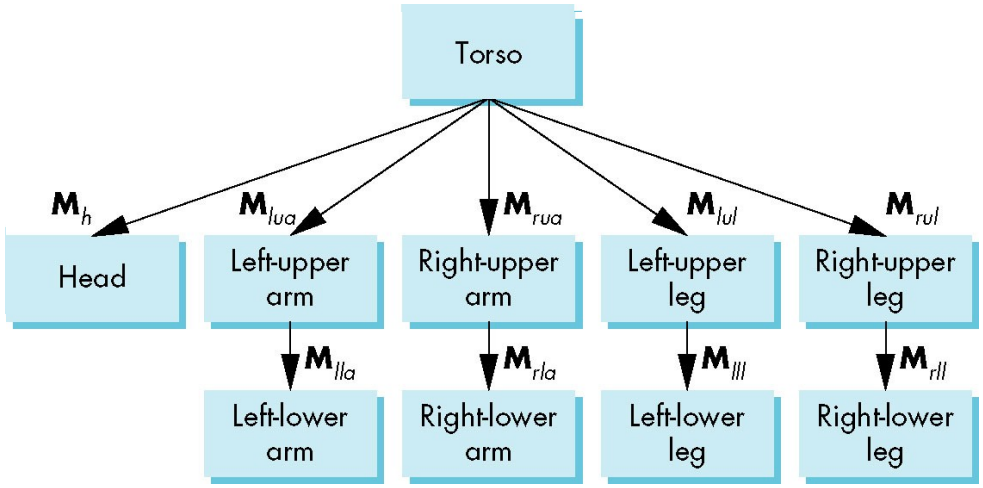
Costruire un modello

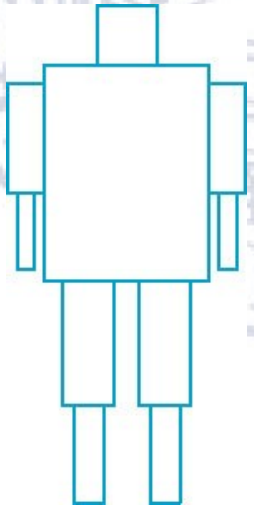




[Angel]

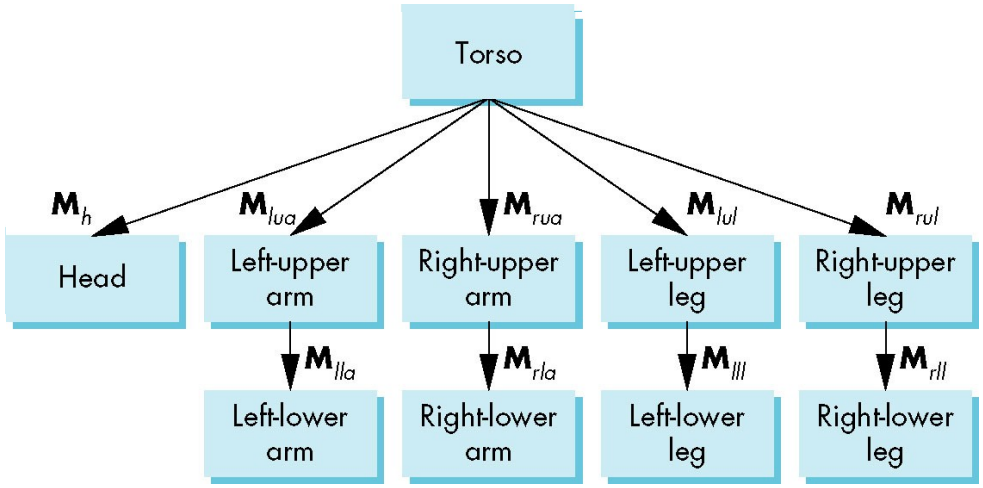
- Supponiamo di creare facilmente la geometria delle parti (es. cilindri)
- Vogliamo creare strutture per accedere alle parti
  - torso(), left_upper_arm()
- Dentro matrici che descrivono posizione del nodo rispetto al genitore

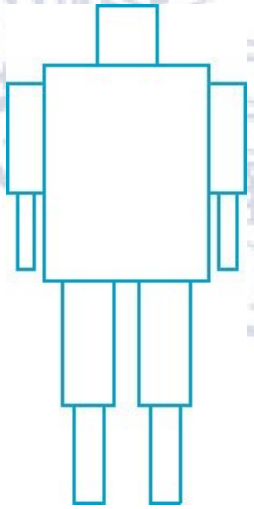




[Angel]

- La posa è determinata da 11 angoli alle giunture (due per la testa e uno per le altre connessioni)
- Fare il rendering si riduce all'attraversamento di un grafo
  - Visito ogni nodo
  - Eseguo la Display function a ognuno di essi: questa descrive la parte del corpo associata, applica la corretta trasformazione per posizione e orientazione

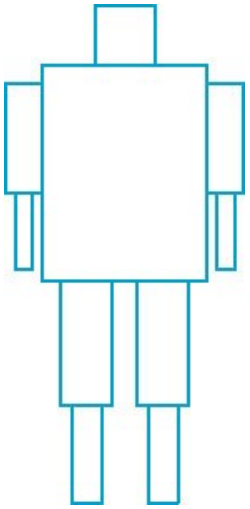


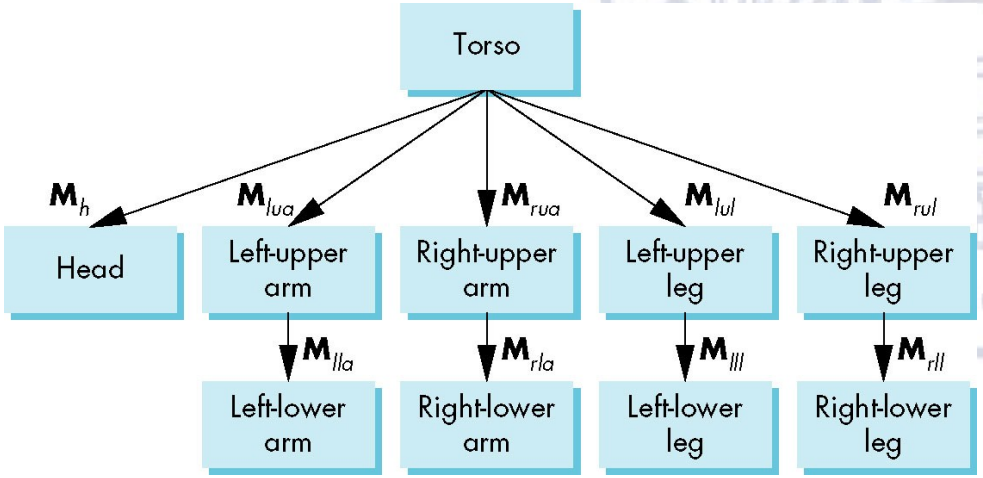


[Angel]

- 10 matrici rilevanti
  - **M** posiziona e orienta il orso
  - $\mathbf{M}_h$ posiziona la testa rispetto al torso
  - $\mathbf{M}_{lua}$, $\mathbf{M}_{rua}$, $\mathbf{M}_{lul}$, $\mathbf{M}_{rul}$ posizionano braccia e gambe
  - $\mathbf{M}_{lla}$, $\mathbf{M}_{rla}$, $\mathbf{M}_{lll}$, $\mathbf{M}_{rll}$ posizionano avambracci e parte inferiore delle gambe

Visualizzazione



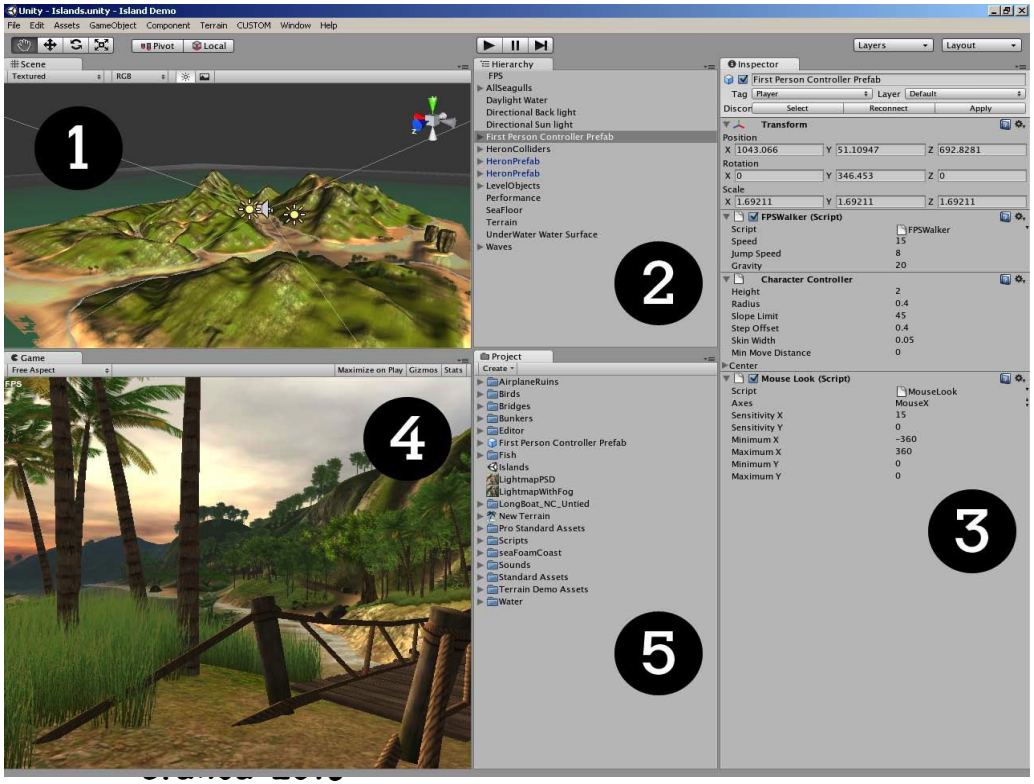


[Angel]

- Metto la model-view matrix a $\mathbf{M}$ e disegno torso
- Set model-view matrix to $\mathbf{MM}_h$ and draw head
- Metto la model-view matrix a $\mathbf{Mm}_{luae}$ e disegno l'arto superiore...
- Invece di ricalcolare $\mathbf{MM}_{lua}$ da zero o usare l'inversione di matrice, si usa memorizzare le matrici parziali in uno stack per poterle recuperare durante l'attraversamento

Modelli gerarchici e scene

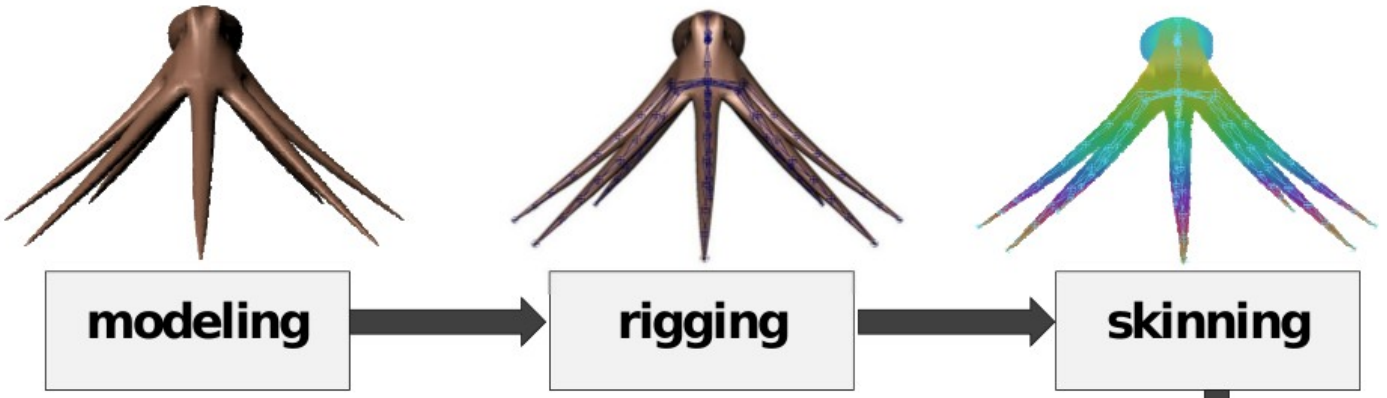
- Nelle applicazioni grafiche di alto livello le scene vengono composte gerarchicamente e le trasformazioni geometriche sono composte
- Es. Unity 3D



Modelli deformabili e animazione

- Tecnica molto usata
  - Rigging (inserimento di uno scheletro)
    - Lo scheletro viene mosso in base alla cinematica es. dello scheletro umano o animale (che può essere in qualche modo modellata da dati reali con tecniche di motion capture)
  - Skinning (movimento della "pelle" in funzione del movimento dello scheletro)
    - In pratica il moto dei nodi delle mesh viene determinato dai punti vicini dello scheletro con una somma pesata degli spostamenti
    - I pesi sono predeterminati
    - Può essere usato anche per modificare interattivamente la posa della mesh

Esempio



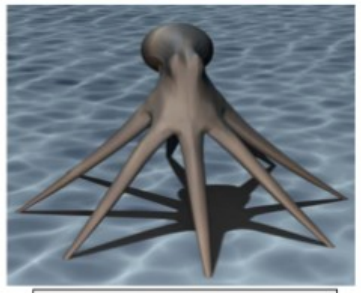
[M Fratarcangeli]

Skeleton

A pose-able framework of joints arranged in a tree structure.

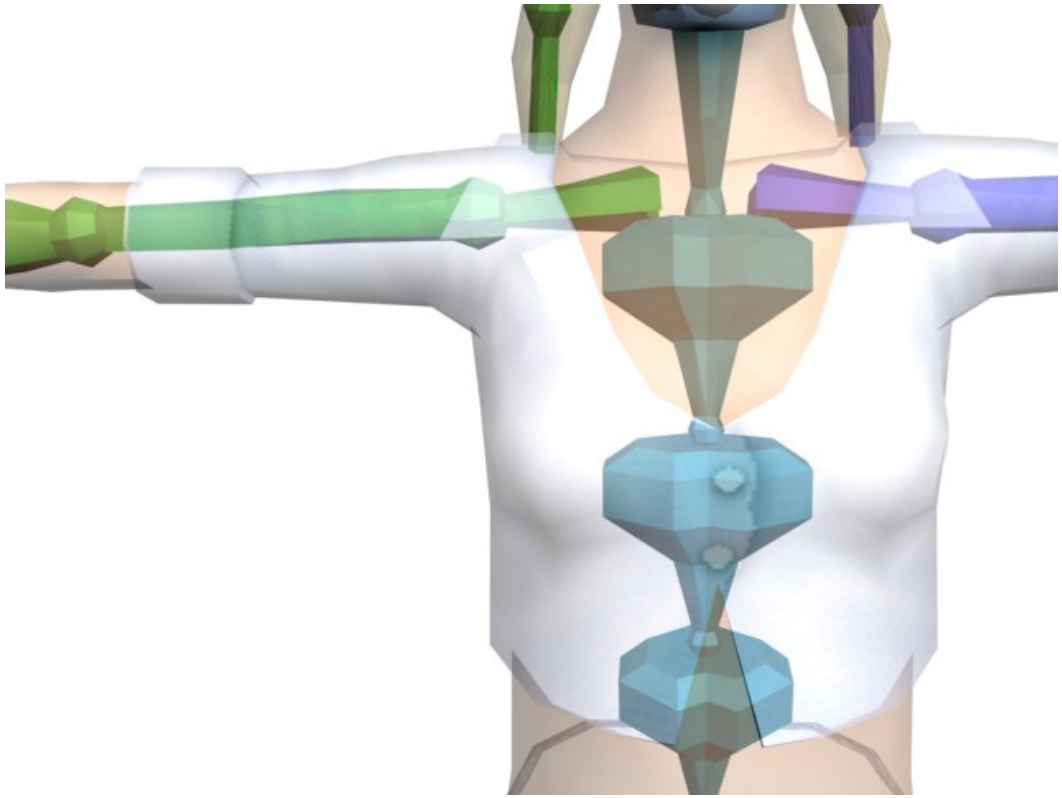
An invisible armature to manipulate the skin and other geometric data of the character.

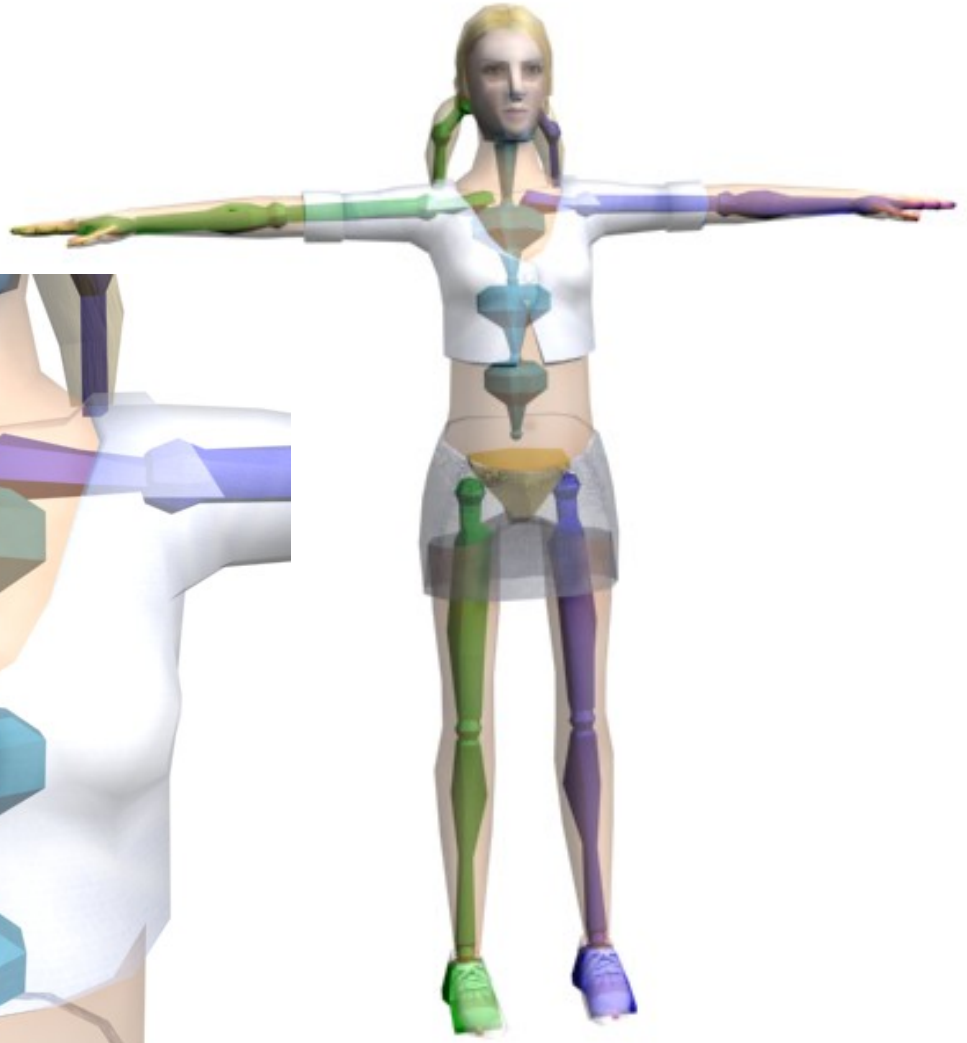
Does not actually render.



Skeleton Rigging

- Occorre definire
  - Bones
  - Joints
  - DOF's
  - Limits





[C.B. Owen]

Posa

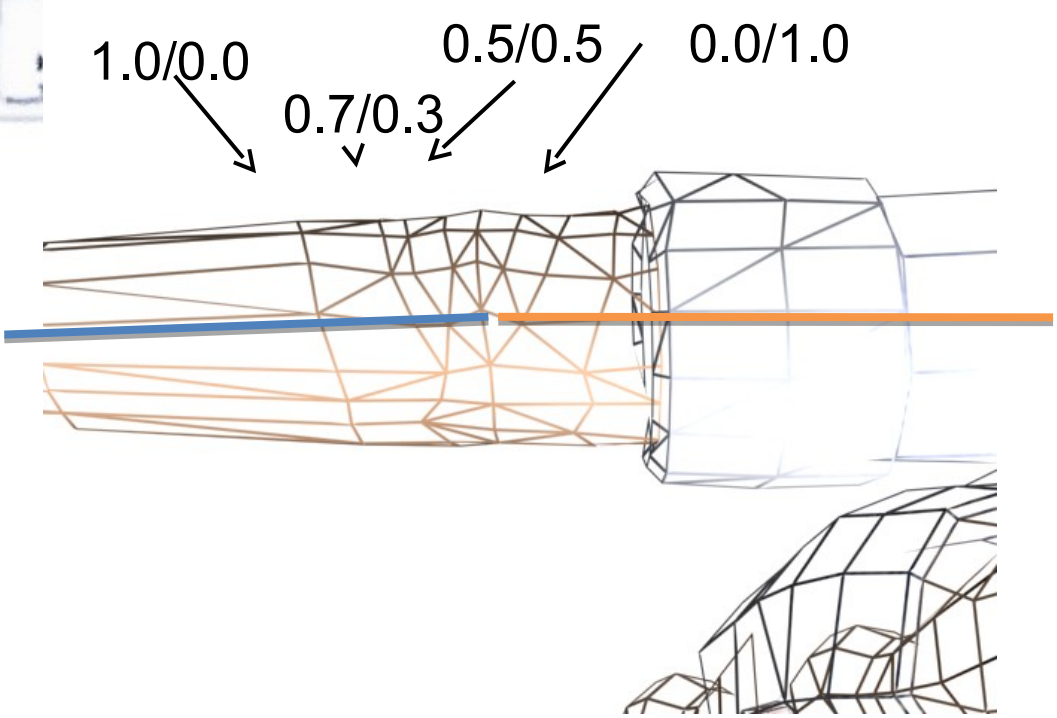
- Adjust DOFs to specify the pose of the skeleton
- We can define a pose Φ more formally as a vector of N numbers that maps to a set of DOFs in the skeleton

$\Phi = [\varphi_1 \varphi_2 \ldots \varphi_N]$

–

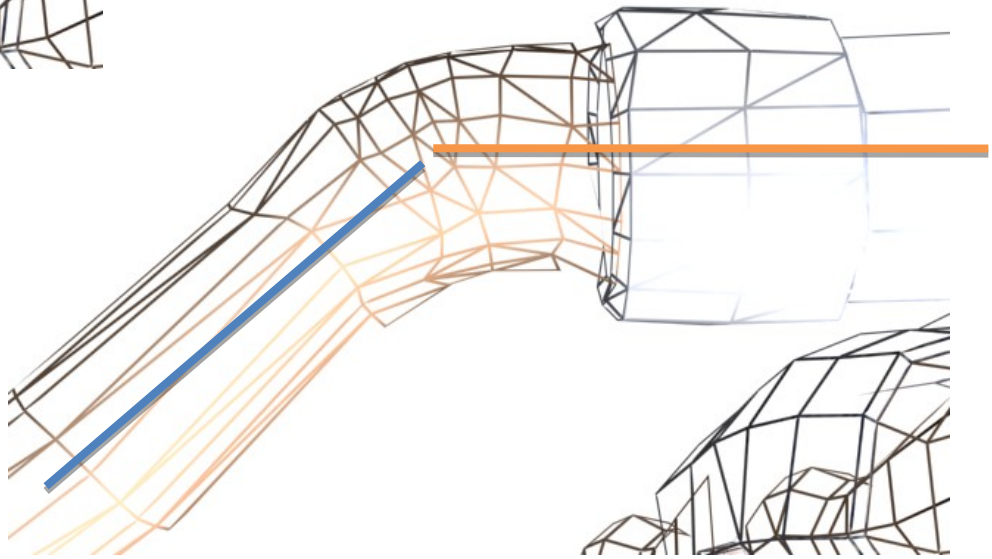
[C.B. Owen]

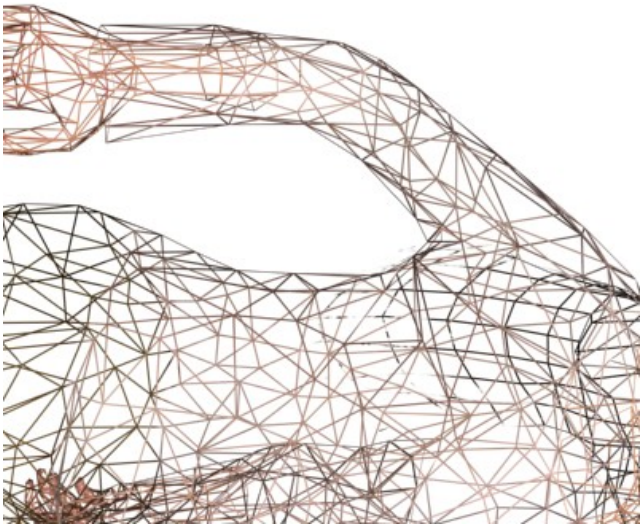
Soft skinning

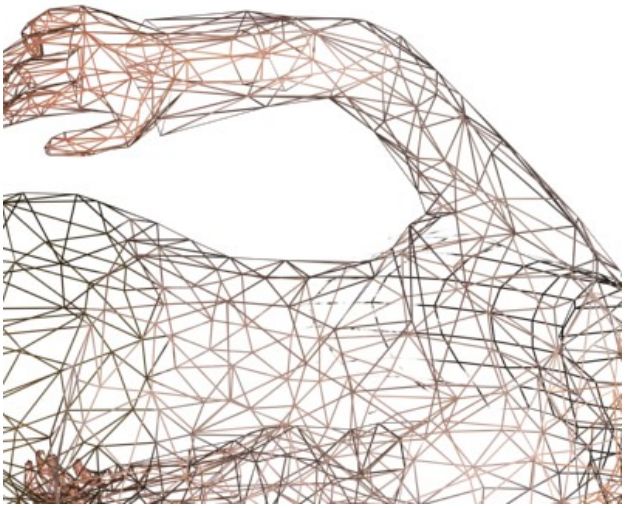


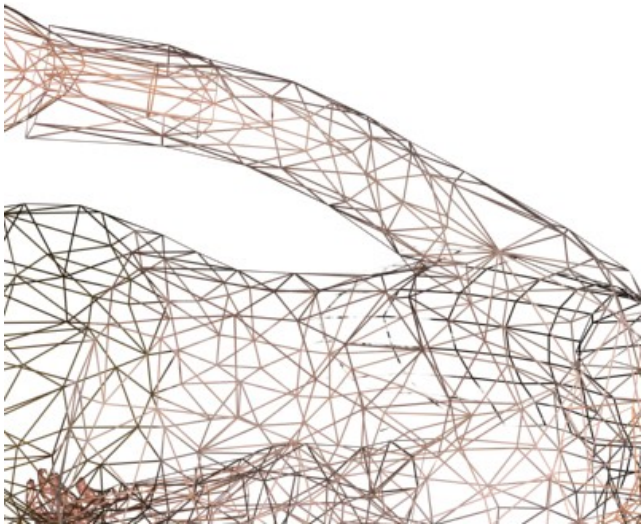
[C.B. Owen]

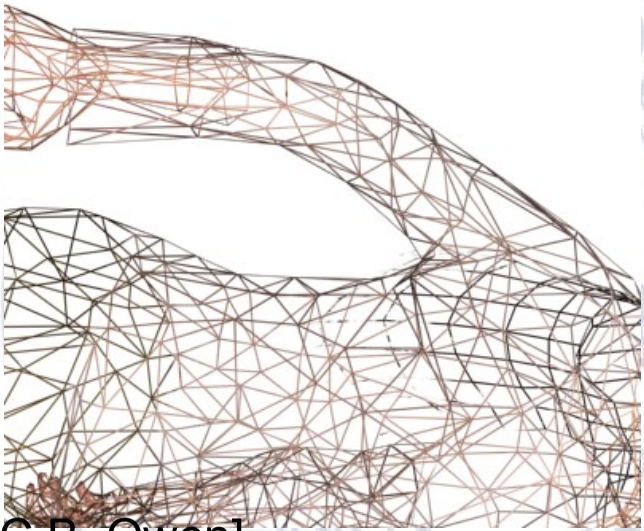
Each vertex can be moved by 1-4 bones, with each bone having a *weight*.







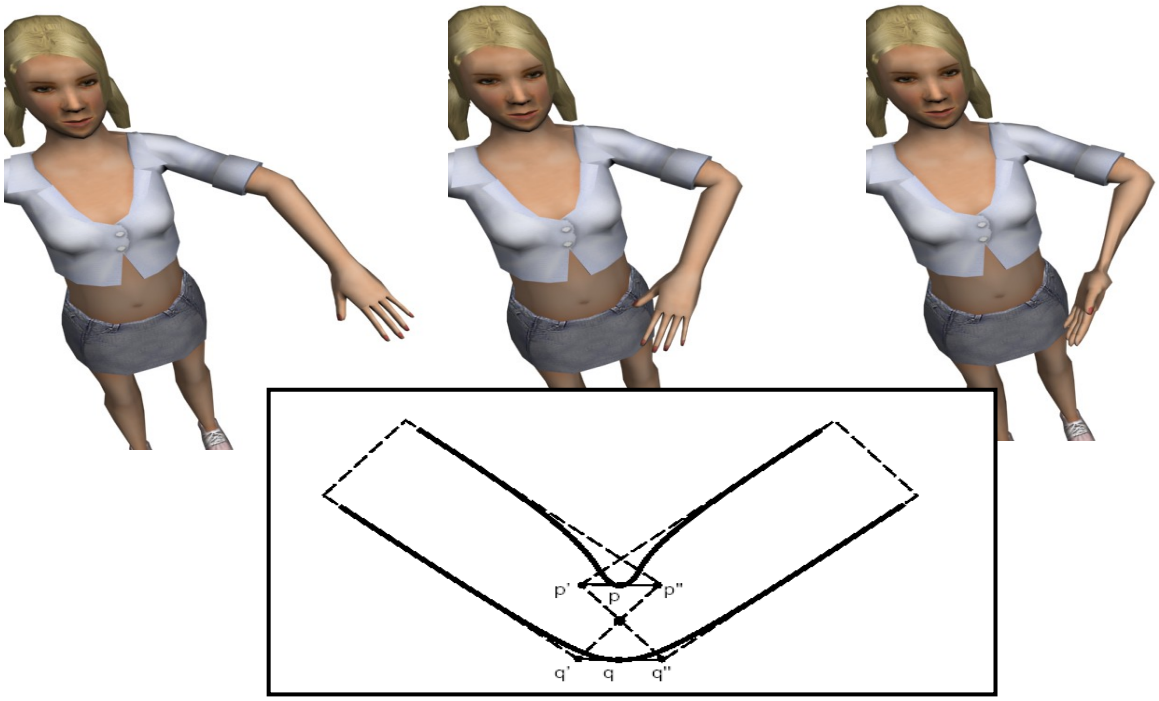




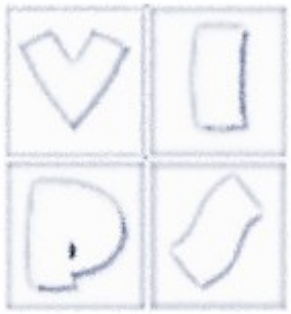
[C.B. Owen]

Limiti

- Non rappresenta i reali movimenti causati dai muscoli
- Non banale da controllare/editare



[C.B. Owen]



Riferimenti

- Wikipedia voce polygon mesh
- http://meshlab.sourceforge.net/
- Angel cap 2.3

Domande di verifica

- Cosa si intende per Constructive Solid Geometry?
- Cos'è una mesh di poligoni?
- Cosa si intende per mesh orientabile?
- Cosa si intende per varietà (2-manifold)
- Come si può ridurre la complessità di una mesh?