Grafica al calcolatore

-

Computer Graphics

4 - Modellazione

Geometria

Ci interessa definire nello spazio le figure geometriche importanti dal punto di vista della modellazione grafica e del rendering

- Rette: sono identificabili da un punto qualsiasi Q che giaccia sulla retta e da una direzione data da un versore u. E facile vedere che sono il luogo dei punti dati da $P = Q + t\mathbf{u}\ t \in IR$

In termini di componenti si vede facilmente che vale la seguente equazione

$$\frac{x - x_Q}{u_x} = \frac{y - y_Q}{u_y} = \frac{z - z_Q}{u_z}$$

- Se si vuole specificare una retta dati due punti R e Q, basta usare le formule date qui sopra tenendo conto che il versore che identica la retta e dato da $\mathbf{u}=(R-Q)/|R-Q|$

- Semiretta: basta aggiungere il vincolo t>= 0
- Segmenti: dati i punti iniziale e finale P e Q possiamo scriverli come $P = Q + t(R-Q)\ t \in [0; 1]$
- Sfere: dato centro O e raggio r, i punti della superficie sferica sono dati dall'equazione $P = O + r\mathbf{u}$ con $\mathbf{u}$ versore generico.
- Si dimostra facilmente che, in termini delle coordinate, la superficie sferica è data dall'equazione

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = r^2$$

- Piani: dati 3 punti non allineati P, Q ed R il luogo dei punti che descrive il piano che li comprende è la combinazione affine

$$S = \alpha P + \beta Q + \gamma R \quad \alpha,\beta,\gamma \in IR \quad \alpha+\beta+\gamma=1$$

- Alternativamente si puo definire un piano a partire da un punto Q che vi appartiene e da un vettore u che ne identica la normale come il luogo dei punti P tali che $(P - Q)\bullet\mathbf{u} = 0$

In termini di coordinate abbiamo

$$(x-x')u_x + (y-y')u_y + (z-z')u_z = 0$$

Per passare dalla prima alla seconda rappresentazione basta prendere come punto Q e come vettore $\mathbf{u} = (P-Q) \times (R-Q)$

- Semispazi: il piano di cui sopra identica due semispazi, uno positivo ed uno negativo:

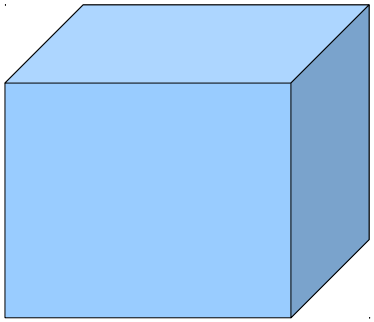
$$(P-Q)\ \mathbf{u} > 0$$
$$(P-Q)\ \mathbf{u} < 0$$

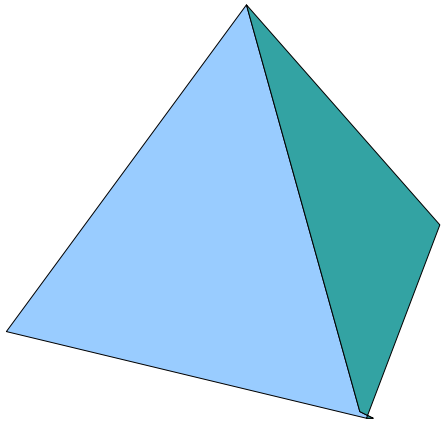
Poligoni

- Un poligono P è un insieme finito di segmenti (spigoli) di IR^2, in cui ogni estremo (vertice) è comune a esattamente due segmenti, che si dicono adiacenti.
- Un poligono è detto semplice se ogni coppia di spigoli non adiacenti ha intersezione vuota.
- Teorema di Jordan: Un poligono semplice P divide il piano in due regioni o facce, una limitata (detta interno di P) ed una illimitata (detta esterno di P).
- Per convenzione, un poligono viene rappresentato dalla sequenza dei suoi vertici $P_1 \ldots P_n$ ordinati in modo che l'interno del poligono giaccia alla sinistra della retta orientata da P_i a P_{i+1}, ovvero i vertici sono ordinati in senso antiorario.

Poliedri

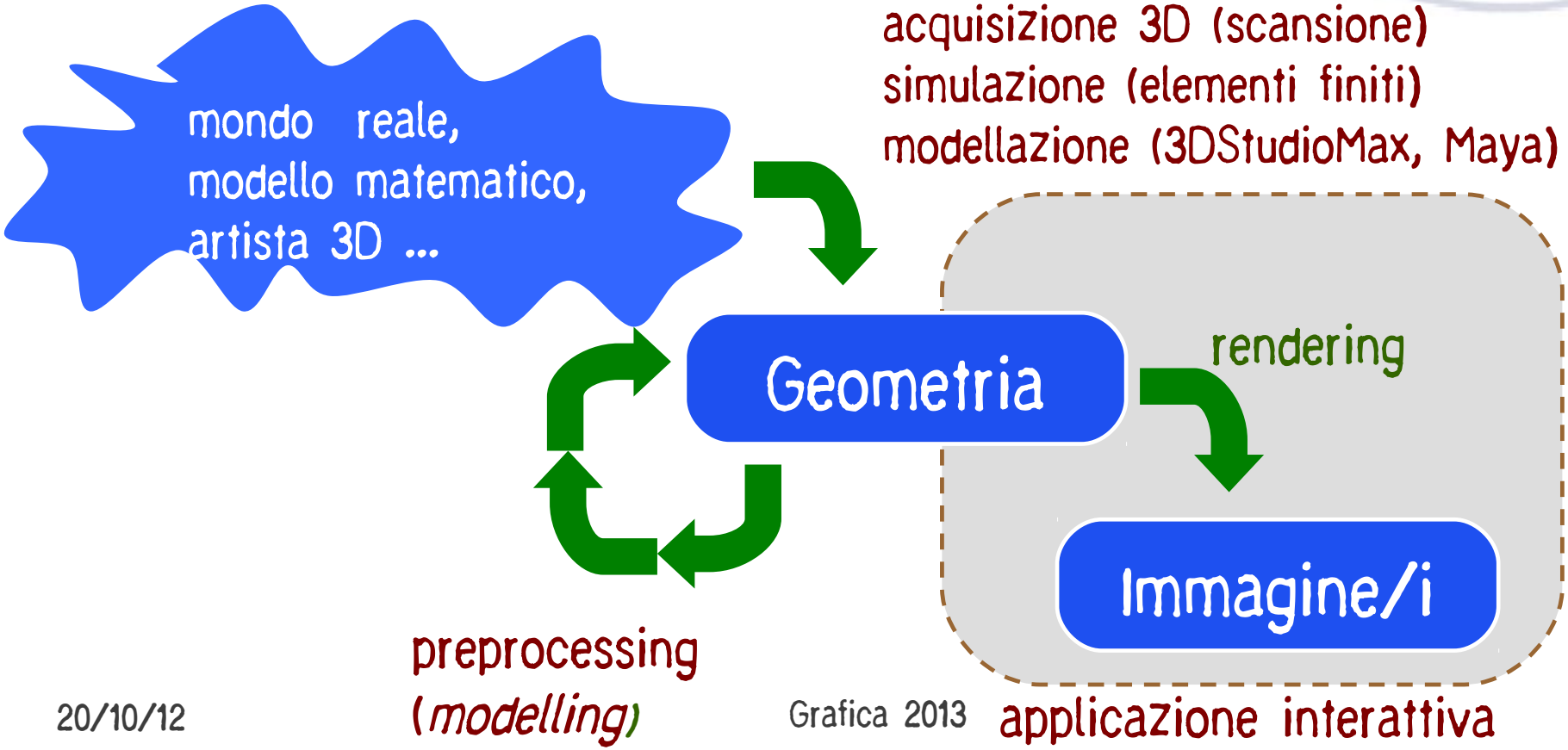
In IR^3 un poliedro semplice e definito da una insieme finito di poligoni (facce) tali che ciascuno spigolo di una faccia e condiviso da esattamente un'altra faccia e le facce non si intersecano che negli spigoli.





Modellazione

- Definiamo ora possibili strutture dati per modellare gli oggetti nello spazio.
- Poi vedremo come modellare anche la formazione delle immagini attraverso il "rendering"



Rappresentazione degli oggetti

- Gli oggetti che si vogliono rappresentare in una applicazione grafica hanno di solito caratteristiche particolari
  - Sono finiti
  - Sono chiusi (non sempre)
  - Sono continui
- Le rappresentazioni di oggetti (regioni dello spazio, in generale) si suddividono in
  - basate sul contorno (boundary): descrivono una regione in termini della superficie che a delimita (boundary representation, o b-rep).
  - basate sullo spazio occupato (o volumetriche).
  - Inizieremo dalle rappresentazioni basate sul contorno, ed in particolare da quella poligonale.

Poligoni e triangoli

- Nella grafica 3D al calcolatore si usa spesso una approssimazione poligonale degli oggetti (del loro contorno).
- Si tratta di approssimare una superficie 2D con un insieme di poligoni convessi opportunamente connessi gli uni agli altri.
- Nel rendering poi attraverso la pipeline di rasterizzazione si lavora a basso livello con i soli triangoli (altri poligoni convertiti in triangoli)
- Possiamo usare la geometria definita finora per definire rigorosamente le proprietà dei modelli triangolati
  - Un insieme di triangoli è matematicamente un 2-complesso simpliciale puro

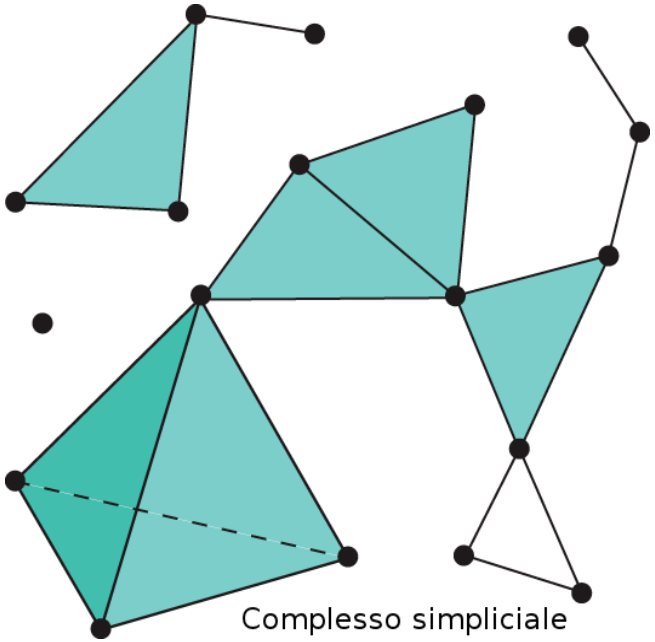
Simplessi

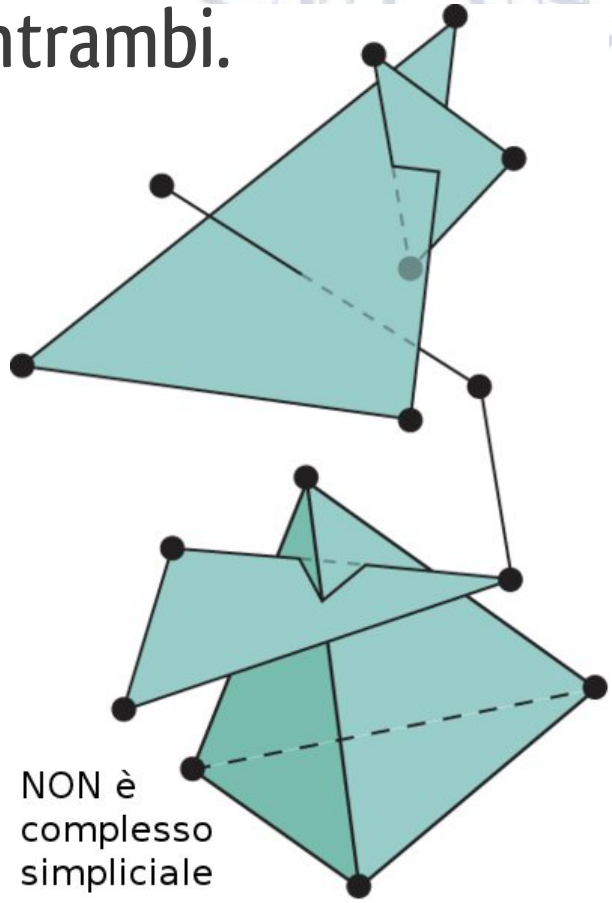
- Un simplesso è l'analogo n-dimensionale del triangolo.
- Specificamente, un simplesso di ordine n (o n-simplesso) è il guscio convesso di n+1 punti affinemente indipendenti in IR^d
- Uno 0-simplesso è un punto, un 1-simplesso è un segmento, un 2-simplesso è un triangolo, un 3-simplesso è un tetraedro.
- Il guscio convesso di un qualunque sottoinsieme degli n+1 punti punti che definiscono il n-simplesso si chiama **faccia** del simplesso. Le facce sono a loro volta simplessi (di ordine n).
- Se il sottoinsieme è proprio, anche la faccia si dice propria.
- Le facce di ordine 0 sono i punti stessi, chiamati vertici. Le facce di ordine 1 si chiamano spigoli. La faccia di ordine n è l' n-simplesso stesso.

Complessi simpliciali

- Un complesso simpliciale K è un insieme di simplessi che soddisfano le seguenti condizioni:
  - Ogni faccia di un simplesso in K appartiene a sua volta a K.
  - L'intersezione di due simplessi 1 e 2 è una faccia comune a 1 e 2 oppure è vuota.
- Se l'ordine massimo dei simplessi è k, K prende il nome di k-complesso simpliciale
  - Per esempio, un 2-complesso simpliciale deve contenere almeno un triangolo e nessun tetraedro.
  - Un k-complesso simpliciale è puro se ogni simplesso di ordine < k è la faccia di un k-simplesso.

- Per esempio, un 2-complesso simpliciale puro è fatto solo di triangoli (non ci sono vertici o spigoli "orfani").
- Due simplessi 1 e 2 sono incidenti se 1 è una faccia propria di 2 o vale il viceversa.
- Due k-simplessi sono (k-1)-adiacenti se esiste un (k-1) simplesso che è una faccia propria di entrambi.


Complesso simpliciale


NON è complesso simpliciale

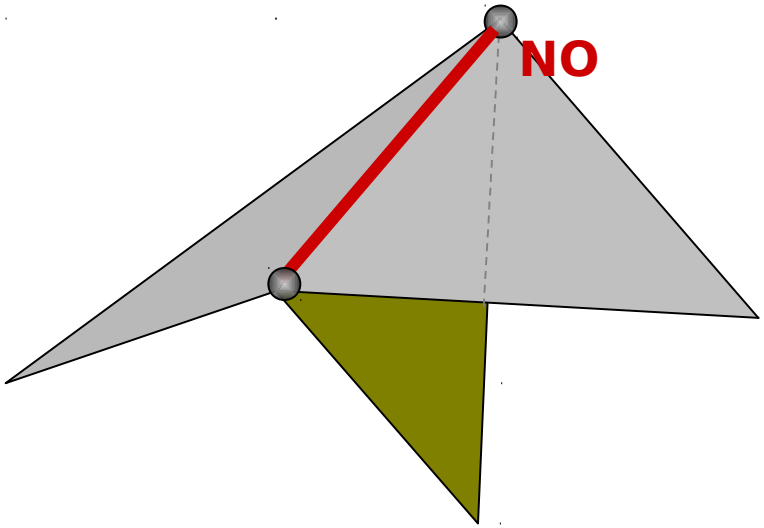
Varietà

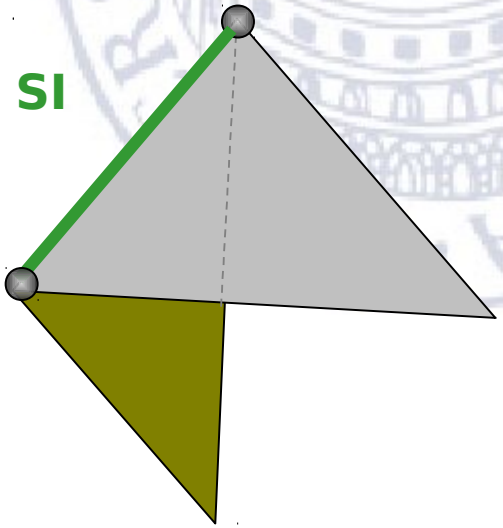
- Una varietà k-dimensionale X è un sottoinsieme di IR^d in cui ogni punto ha un intorno omeomorfo alla sfera aperta di IR^k.
- In generale le superifici degli oggetti solidi (sfere, poliedri, ecc.) sono varietà bidimensionali.
- Omeomorfismo: applicazione biiettiva, continua, con inversa continua. Intuizione: trasformazione senza "strappi".
  - In una varietà k-dimensionale **con bordo** ogni punto ha un intorno omeomorfo alla sfera aperta o alla semisfera aperta di IR^k.
  - Il bordo di X è l'insieme dei punti che hanno un intorno omeomorfo alla semisfera aperta.
  - Una varietà è sempre una varietà con bordo, eventualmente vuoto.
  - Il bordo, se non è vuoto, è a sua volta una varietà k-1 dimensionale senza bordo.

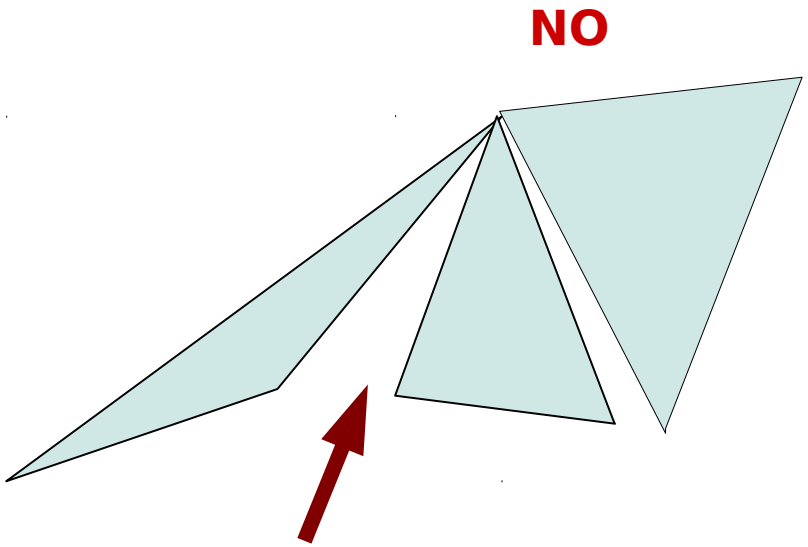
Mesh poligonali

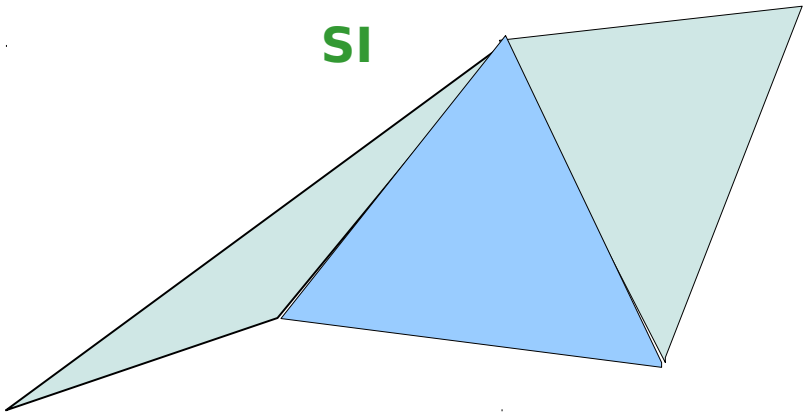
- Una maglia (mesh) triangolare è 2-complesso simpliciale puro che è anche una varietà bidimensionale con bordo.
- I triangoli della maglia si chiamano anche facce.
- La condizione di essere varietà si traduce nei seguenti vincoli ulteriori sulla struttura del complesso simpliciale:
  - uno spigolo appartiene 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 appartenere alla chiusura transitiva della relazione di 1-adiacenza, ovvero devono formare un ventaglio o un ombrello.
- Si usa il termine condizione 2-manifold (varietà)

Condizione 2-manifold



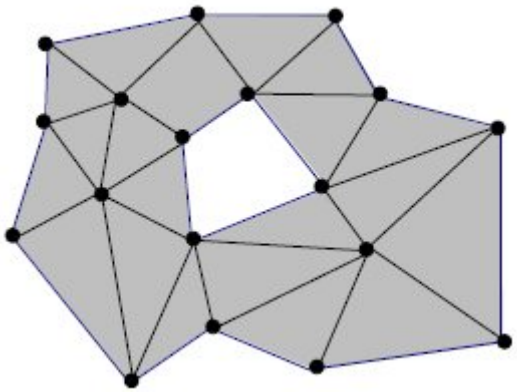


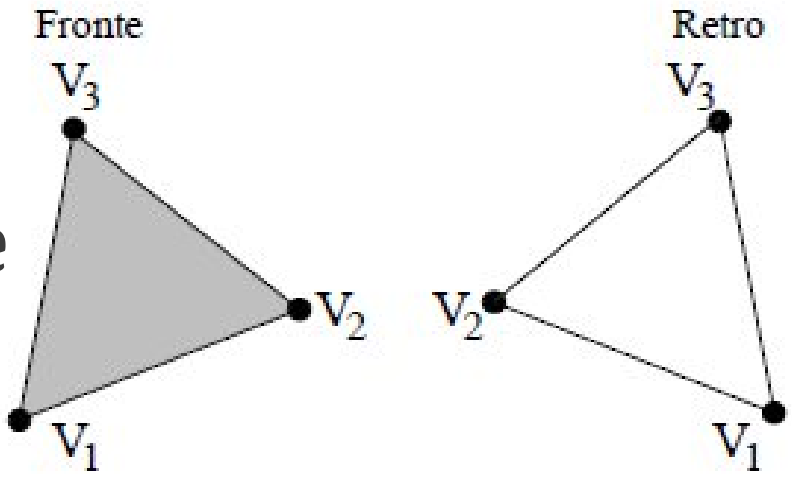




Orientazione

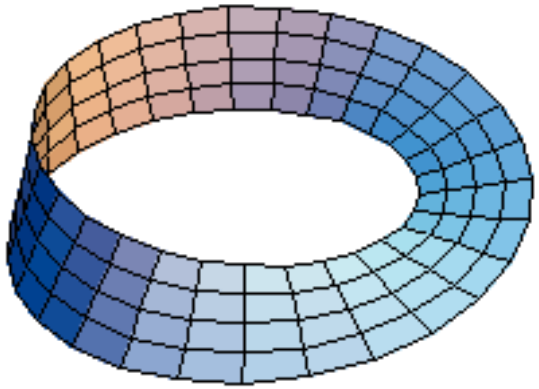
- Il bordo della maglia consiste di uno o più anelli (sequenza chiusa di spigoli) o loop.
- Se non esistono spigoli di bordo la maglia è chiusa (come quelle che rappresentano la superficie di una sfera).
- L'orientazione di una faccia è data dall'ordine ciclico (orario o antiorario) dei suoi vertici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





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)



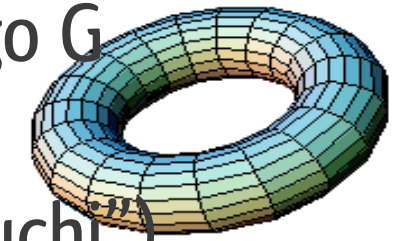
Maglie/Mesh generiche

- Abbiamo definito la maglia triangolare. In maniera analoga si può estendere la definizione a maglie poligonali gneriche
  - Maglie poligonali generiche: i poligoni possono avere qualsiasi numero di spigoli e non è detto che ci sia un solo tipo di poligono. Sono raramente utilizzate in grafica al calcolatore
  - Quadrangolari (quad meshes): gli elementi poligonali sono tutti quadrilateri. Sono alle volte usate, per esempio se si vuole fare il rendering di un terreno descritto da un array di altezze. In una maglia quadrangolare bisogna imporre un vincolo aggiuntivo di planarità per ogni quadrilatero che la compone.
- OpenGL consente di descrivere maglie poligonali generiche, ma per disegnarle li suddivide in triangoli.

Equazione di Eulero

Se V è il numero di vertici, L il numero di spigoli ed F il numero di facce della maglia poligonale orientabile chiusa di genere G, allora vale la Formula di Eulero $V - L + F = 2 - 2G$

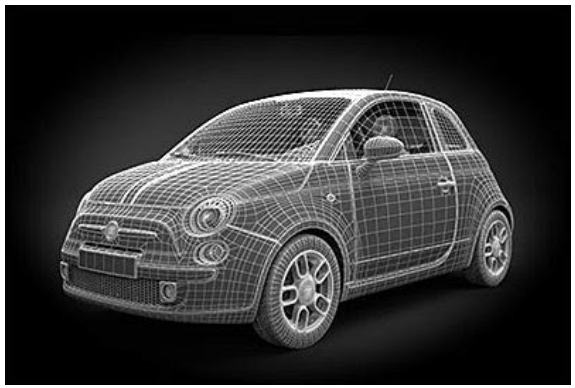
Una superficie ha genere G se può essere tagliata lungo G linee semplici chiuse senza disconnetterla (intuitivamente, ma non rigorosamente "numero di buchi")

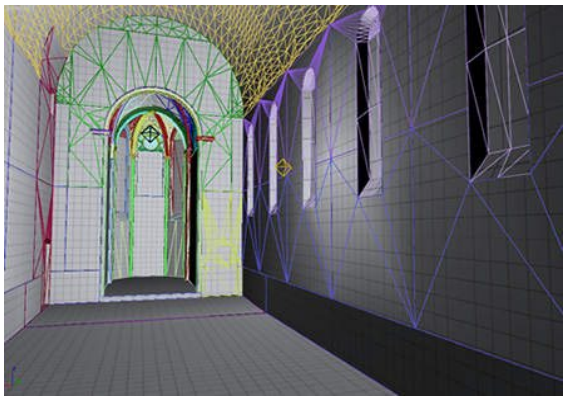


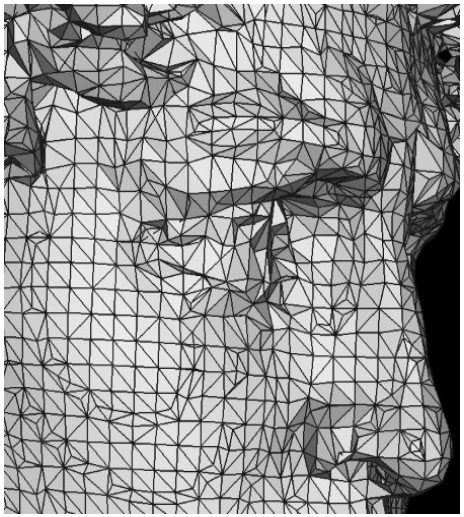
- Il genere di una superficie dipende (e determina) la sua topologia; per una sfera, per esempio, G = 0, mentre per un toro (una ciambella) G = 1.
- Più in generale, per una maglia poligonale orientabile (e varietà bidimensionale) vale la formula $V - L + F = 2(S - G) - B$
  - S numero di componenti connesse, B è il numero di anelli di bordo

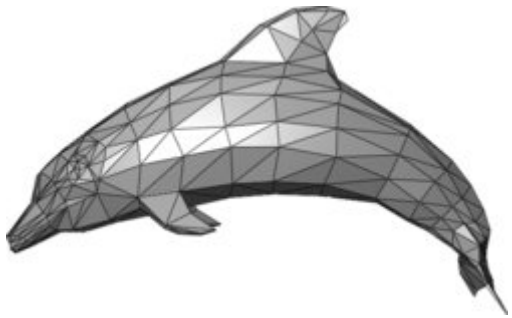
Mesh di triangoli

- Generate da modellazione CAD, acquisizione con scanner, ricostruzione da immagini (Computer Vision)
- Anche molto voluminose: vedremo come **semplificarle**



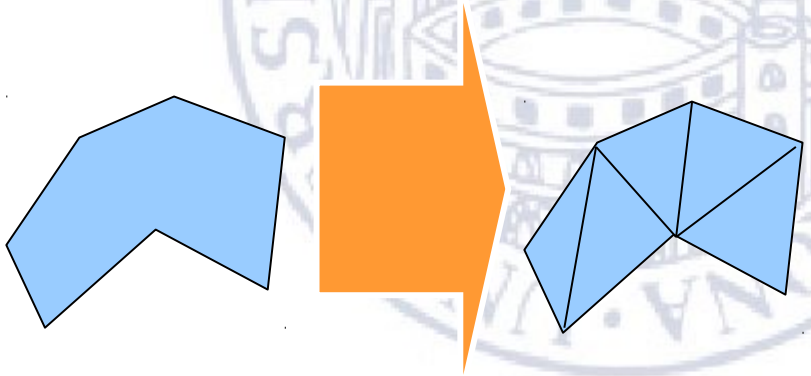




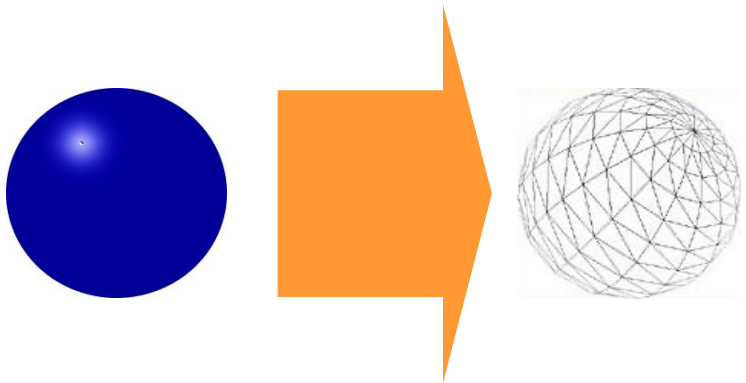


Costruzione della mesh triangolare

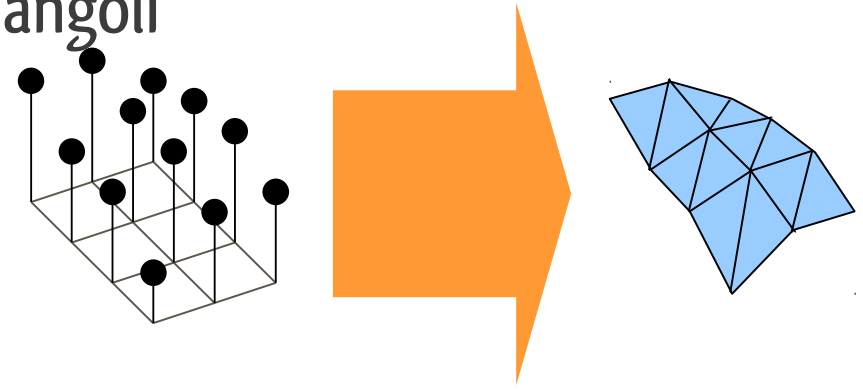
- Conversione da altri formati:
  - Poligoni → Triangoli



  - Superf. Quadriche → Triangoli

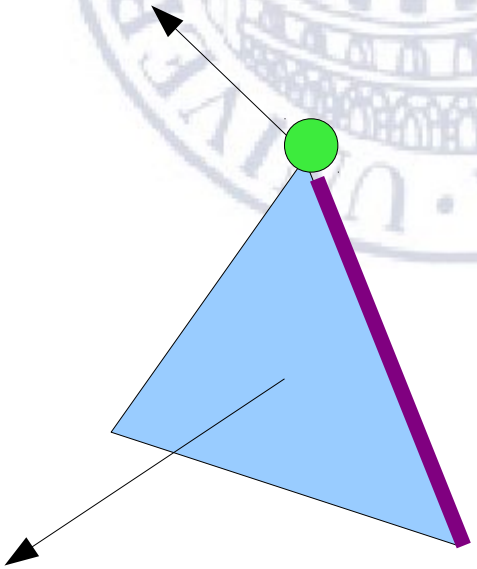


  - Campi di altezze o Punti → Triangoli



Mesh triangolare - 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
  - coordinate texture

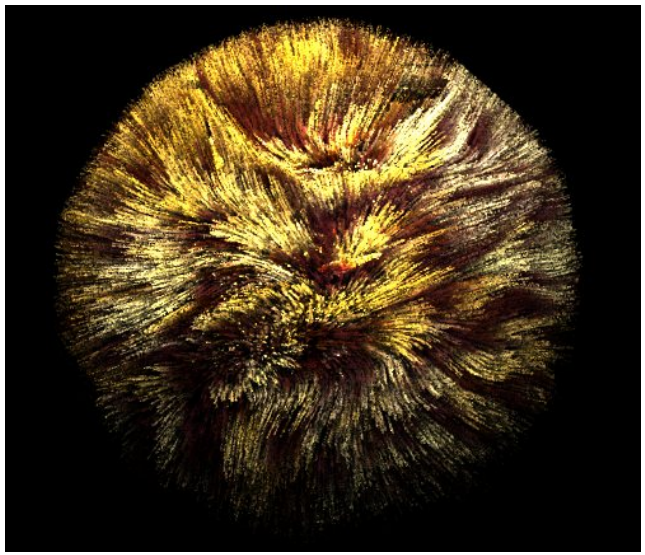


Mesh triangolare - Limiti

- 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)



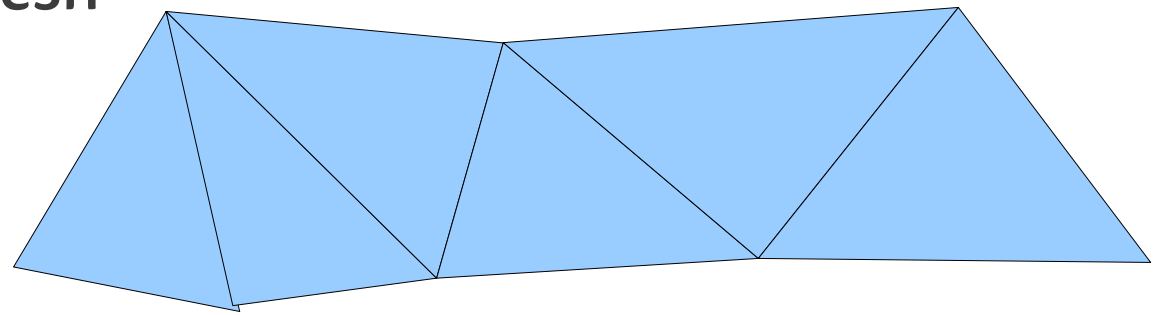
by M. Turitzin and J. Jacobs
Stanford Uni (real time!)

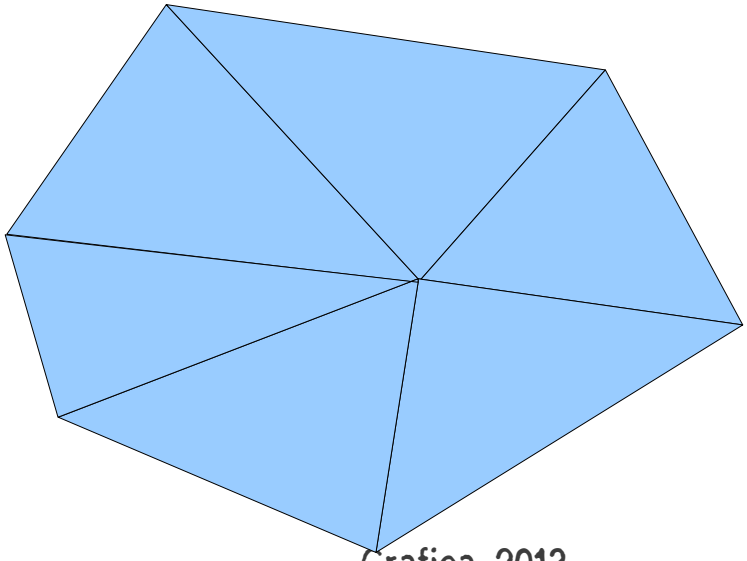
Evitare ridondanze

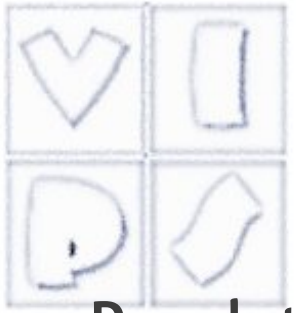
- Quando si devono disegnare due triangoli con uno spigolo in comune, questo viene disegnato due volte. Questo introduce un certo grado di ridondanza che può incidere sulle prestazioni
- Si preferisce quindi raggruppare i triangoli di una maglia in opportuni gruppi che possono essere elaborati in maniere più efficiente. Per le OpenGL si hanno due possibili gruppi
  - Fan di triangoli: è un gruppo di triangoli che hanno in comune un vertice. Il primo viene specificato completamente, per i successivi basta dare il nuovo vertice. Efficiente, ma i triangoli che incidono su un vertice sono in genere pochi
  - Strip di triangoli: gruppo di triangoli che posseggono a due a due uno spigolo in comune. Di nuovo il primo triangolo viene specificato normalmente, per i successivi basta specificare il nuovo vertice. Meno efficiente, ma le strip in genere contengono più triangoli delle fan

Strip e fan

- Esistono algoritmi per creare queste rappresentazioni dalle mesh







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.

Memorizzazione

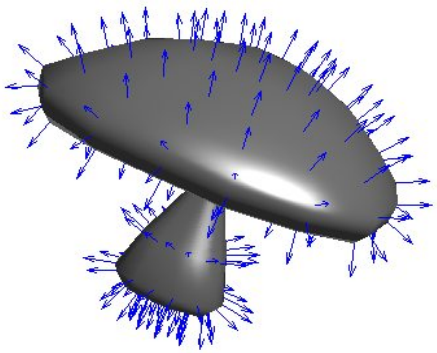
- Alla creazione dei modelli vengono in genere generati nodi, connettività e attributi. Gli algoritmi di processing e rendering devono accedere in vario modo a tale informazione
- Diversi modi di rappresentare questa informazione
- Nei programmi applicativi sono quindi necessarie delle procedure per 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

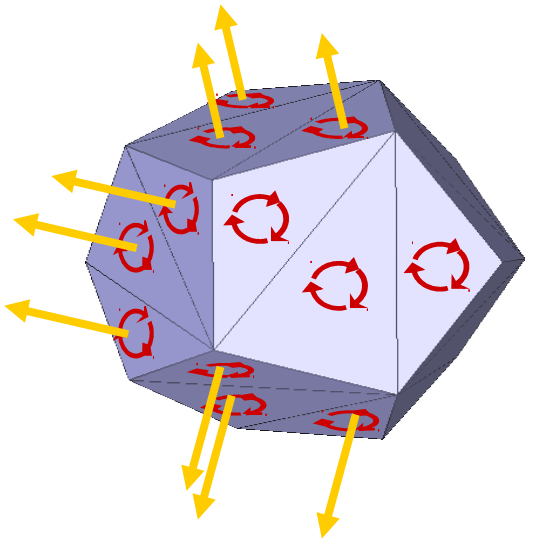
Elementi base

- 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)

Elementi base

- Normali: è fondamentale sapere quale è l'esterno della superficie e quale l'interno; atal scopo si associa spesso ad una maglia poligonale anche l'informazione sulla normale uscente. Come vedremo questa informazione può essere associata alle facce, come sarebbe naturale, oppure ai ai vertici (per ragioni che saranno chiare nel seguito).





Struttura della mesh

- I vertici danno informazioni di tipo posizionale, gli spigoli informazioni di tipo connettivo (non contengono 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
- 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: $\mathbf{n} = (V_3 - V_2) \times (V_1 - V_2)$

Ricerca di incidenza

- Una ricerca di incidenza è una procedura che determina tutti gli elementi di un dato tipo che incidono su un certo elemento
- Ad esempio può essere interessante e utile sapere, data una faccia, quali siano i vertici della maglia che incidono su tale di faccia.
- Gli algoritmi di processing dei modelli si basano su calcoli sul vicinato (calcolo curvature, smoothing, ecc.)
  - Dobbiamo renderli efficienti
  - A volte si applicano anche a runtime
- Le strutture dati possono semplificare tali ricerche

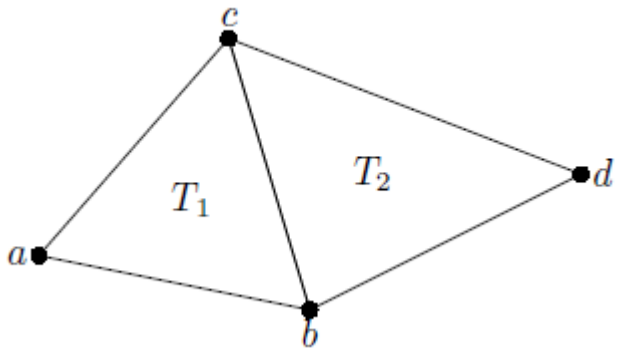
Strutture dati

- Immediata: specificare tutte le facce della maglia come terne di triplette di coordinate cartesiane. Es.

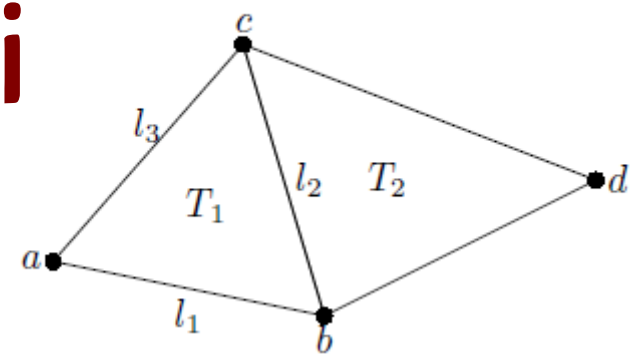
$T_1 = (a_x, a_y, a_z); (b_x, b_y, b_z)(c_x, c_y, c_z)$, …

- Inefficiente; ad esempio i vertici vengono ripetuti nella lista dei poligoni.
- Ricerche di incidenza sono inoltre particolarmente onerose (e a volte non definibili)

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



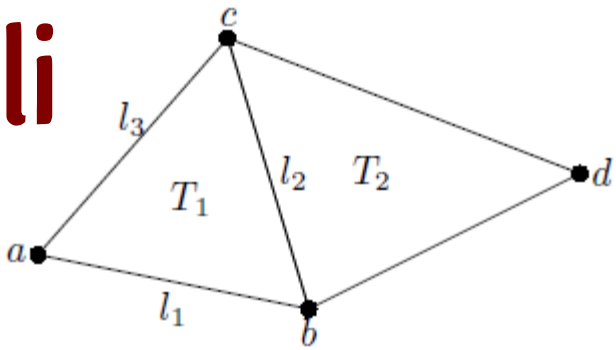
Lista dei vertici



- Si costruisce una lista dei vertici senza ripetizioni
- Le facce sono descritte dai puntatori ai vertici che la compongono (in genere in senso antiorario)
  - In questo modo si elimina la duplicazione dei vertici, ma non quello sugli spigoli; uno spigolo appartenente a due triangoli viene immagazzinato due volte
- Le ricerche di incidenza continuano ad essere complesse

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

Lista degli spigoli

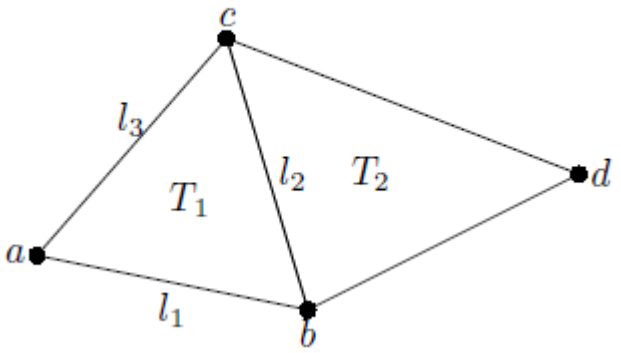


- Lista dei vertici (senza ripetizioni) ed una lista degli spigoli, ciascuno composto dai due puntatori ai vertici incidenti sullo spigolo;
- ciascuna faccia viene descritta infine dai puntatori degli spigoli che la compongono (in genere in senso antiorario)
- In questo modo si elimina la ripetizione sui vertici e sugli spigoli
- Le ricerche di incidenza sono più semplici, ma non tutte

```
typedef struct
{
float x,y,z;
} vertice;
typedef struct
{
vertice* in,
fin;
} spigolo;
typedef struct
{
spigolo*
l_1,l_2,l_3;
} faccia;
```

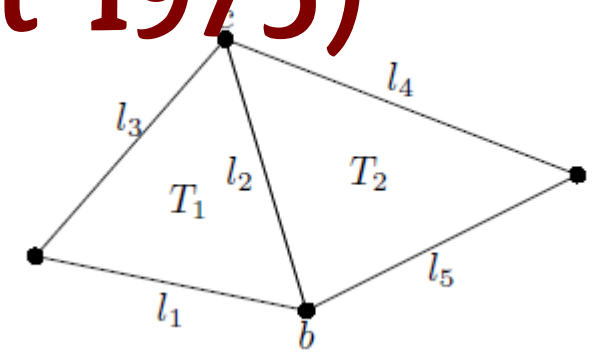
Lista degli spigoli estesa

- Un terzo passo può essere ottenuto aggiungendo alla struttura dati degli spigoli anche i due puntatori alle facce incidenti sullo spigolo
- Ad esempio lo spigolo l_2 punta a T_1 e T_2, oltre che a b e c come prima.
- In questo modo si semplificano di molto le ricerche di incidenza spigolo-faccia




```
typedef struct {
float x,y,z;
} vertice;
typedef struct {
vertice* in, fin;
faccia* sin,
dest;
} spigolo;
typedef struct {
spigolo* a,b,c;
} faccia;
```

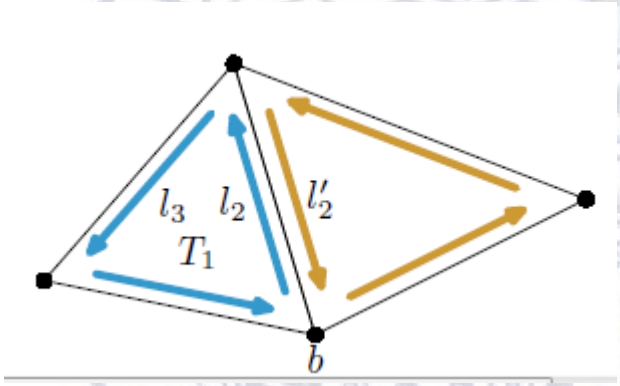
Winged edge (Baugmart 1975)



- 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).

```
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)

```
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
- Le applicazioni grafiche in genere caricano tali file ed usano l'informazione contenuta in essi per riempire una struttura dati più utile ai fini algoritmici (per esempio la half-edge)

Esercizi

- descrivere una procedura che, data una rappresentazione winged-edge di una maglia triangolare, “stampi” tutti le facce incidenti su un dato vertice v, assumendo data una procedura stampaFaccia(we faccia *f)
- descrivere una procedura che, data una rappresentazione half-edge di una maglia triangolare, “stampi” tutti i mezzi-spigoli uscenti dal vertice v con una procedura stampaSpigolo(he spigolo *l)

Formati per mesh

- Esempio .off, obj, .ply, .wrl, x3d, .mesh
- Possono supportare anche diversi tipi di primitive, scene graph
- Esempio .off

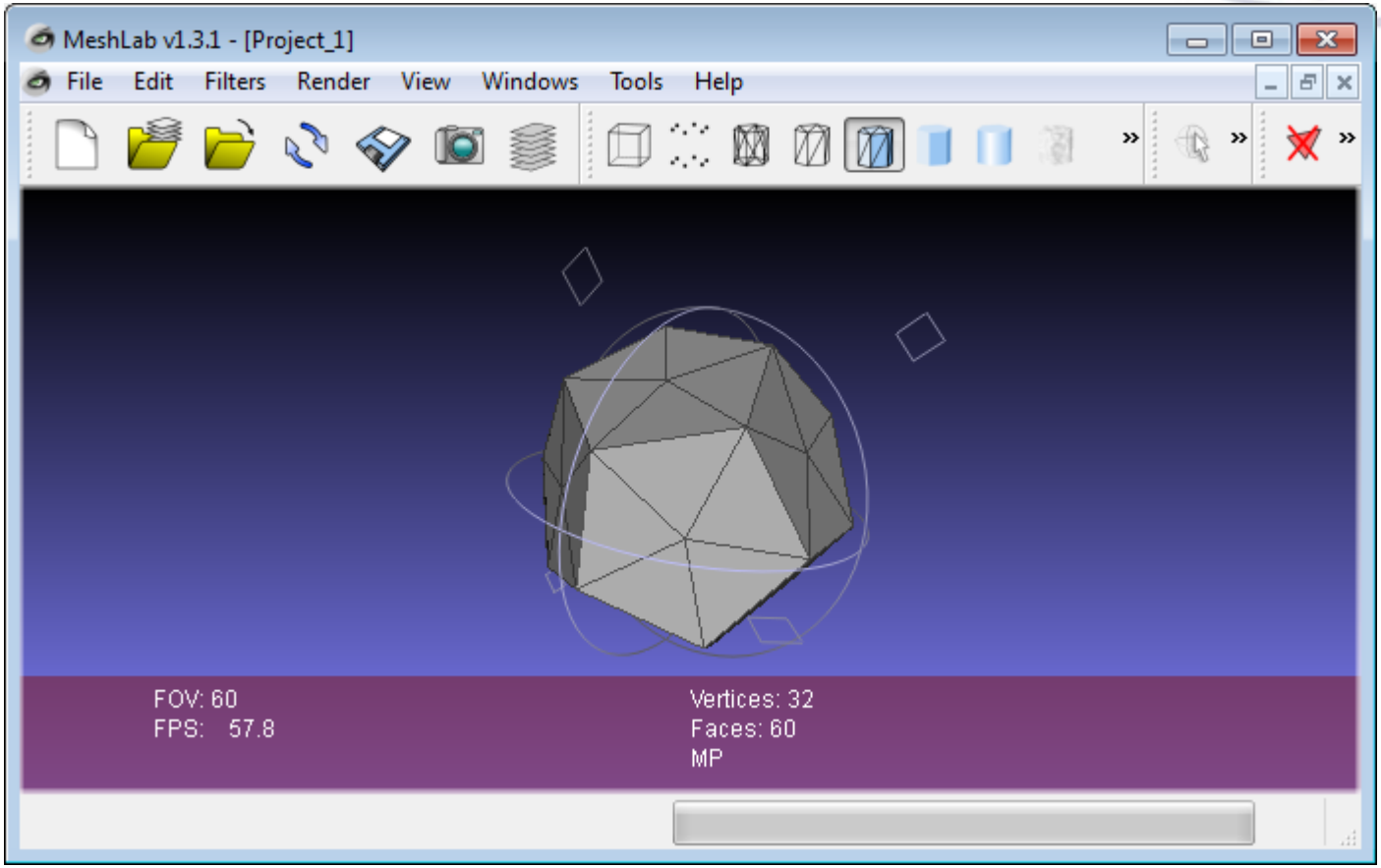
```
OFF
4  4  0
-1  -1  -1
1  1  -1
1  -1  1
-1  1  1
3  1  2  3
3  1  0  2
3  3  2  0
3  0  1  3
```

Mesh processing

- Esistono molti algoritmi sviluppati in ambito geometria computazionale per il processing dei dati dei modelli
  - Meshing/remeshing: generare nuove rappresentazioni poligonali dalle nuvole di punti: per i modelli acquisiti da scanner non è affatto ovvia la triangolazione
  - Riparazione: le mesh derivate da ricostruzione algoritmica possono avere buchi e problemi topologici: esistono algoritmi per cercare di tappare/riparare
  - Decimazione: algoritmi per semplificare i modelli e risparmiare memoria, o generare modelli multirisoluzione
  - Densificazione: si cerca di aumentare il dettaglio

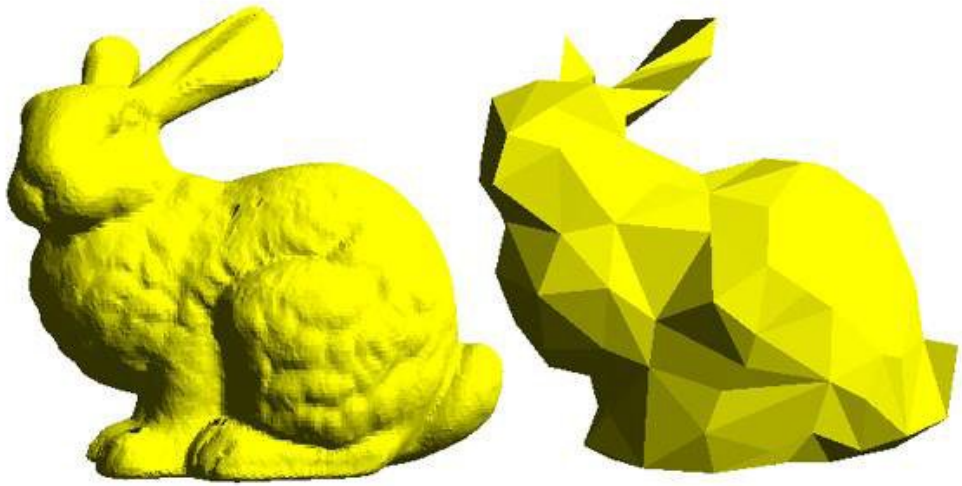
Meshlab

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



Semplificazione

- Le maglie poligonali, se molto accurate, possano essere computazionalmente troppo onerose da gestire.
- E' quindi importante poterle semplificare, se necessario.
- Idea semplice: rimuovo un vertice alla volta e riparo il buco.
- Idealmente vogliamo rimuovere il maggior numero possibile di vertici per cui la risultante maglia semplificata sia una buona approssimazione della maglia fine originale (devo calcolare l'errore introdotto!).

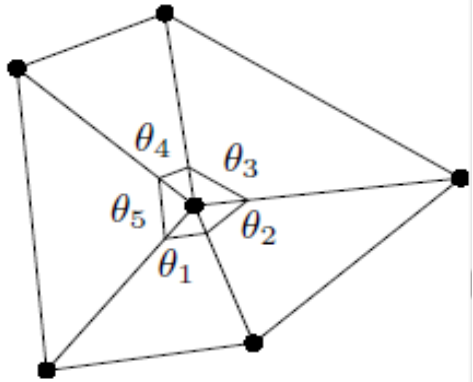


- Molte applicazioni richiedono modelli con un elevato dettaglio, al fine di mantenere un livello di realismo convincente o accuratezza di calcolo.
- La complessità però non è sempre richiesta in fase di rendering: il costo per disegnare un modello è direttamente legato alla sua complessità
  - Utile avere versioni semplificate del modello.
  - vorremmo che la semplificazione avvenisse in modo automatico.
- Un algoritmo di semplificazione prende in ingresso una maglia triangolare e ne produce una versione versione approssimata con meno triangoli. Vi sono due categorie di algoritmi:
  - Decimazione dei vertici.
  - Contrazione iterativa degli spigoli.

Algoritmi

- Si seleziona iterativamente con euristiche locali il vertice meno , si rimuove e si ri-triangola il buco. Criteri?
- Esempio: diradare le zone a bassa curvatura.
  - Calcolo sui vertici un'approssimazione della curvatura (Gaussiana), che prende il nome di angle deficit uguale a 2 meno la somma degli angoli interni di tutti i triangoli incidenti sul punto

$$\varepsilon(v) = 2\pi - \sum_i \theta_i(v)$$



- E' dimostrato che se $\varepsilon(\mathbf{v}) = 0$ allora il vertice v

e tutti i vertici ad esso connessi giacciono su un piano.

- Si vede che vertici con $\varepsilon(v)$ nullo (o molto piccolo) sono essenzialmente "inutili"

Contrazione iterativa degli spigoli

- Ad ogni iterazione viene eliminato per contrazione lo **spigolo** di costo minore, ed i costi dei vicini vengono aggiornati.
- I vari metodi differiscono per la metrica di errore impiegata.

Una metrica semplice:

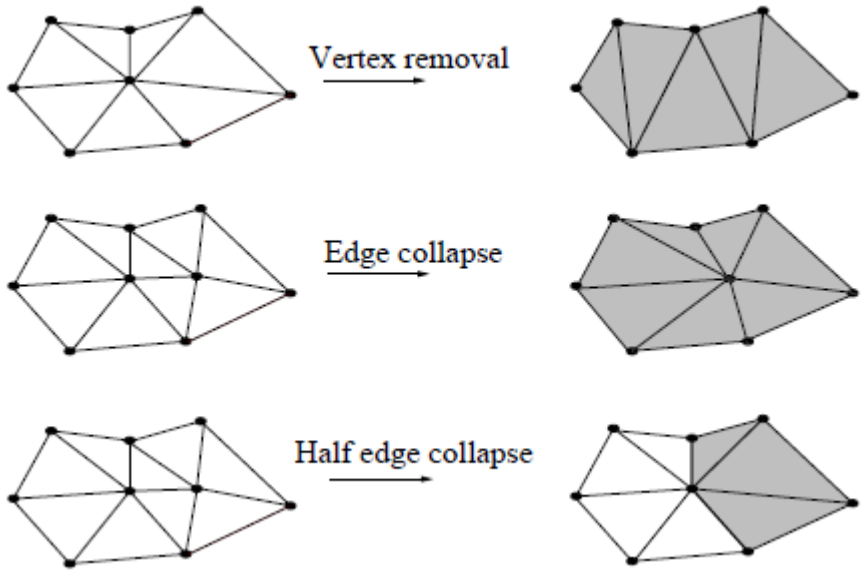
$$\varepsilon(\overline{P_1P_2}) = \frac{\|P_2 - P_1\|}{|\mathbf{n}_\ell \cdot \mathbf{n}_r|}$$

dove P_1 e P_2 sono i vertici incidenti sullo spigolo e $\mathbf{n}_l$ e $\mathbf{n}_r$ le normali delle facce incidenti.

R.

Rimozione vertici/spigoli

- Varie possibilità:
- vertex removal rimuove un vertice e triangola la cavità
- edge collapse rimuove uno spigolo fondendo due vertici in uno nuovo
- half edge collapse uno dei due vertici rimane fermo.
  - In figura, in grigio i triangoli modificati



Metodo di Garland-Heckbert

- Procede per contrazione iterativa degli spigoli.
- A ciascun vertice $\mathbf{v}$ è associato in insieme di piani, $\Pi(\mathbf{v})$: inizialmente sono i piani definiti dai triangoli incidenti sul vertice. Dopo una contrazione, al nuovo vertice si associa l'unione degli insiemi dei vertici che sono stati contratti
- A ciascun vertice $\mathbf{v}$ è associato un errore $\Delta(\mathbf{v})$, pari alla somma dei quadrati delle distanza di v dai piani di $\Pi(\mathbf{v})$.
  - Inizialmente l'errore è 0 per costruzione.

$$\Delta(\mathbf{v}) = \sum_{\mathbf{p}\in\Pi(\mathbf{v})} (\mathbf{p}^T\mathbf{v})^2 = \sum_{\mathbf{p}\in\Pi(\mathbf{v})} (\mathbf{v}^T\mathbf{p})(\mathbf{p}^T\mathbf{v}) = \sum_{\mathbf{p}\in\Pi(\mathbf{v})} \mathbf{v}^T(\mathbf{p}\mathbf{p}^T)\mathbf{v} = \mathbf{v}^T \underbrace{\left(\sum_{\mathbf{p}\in\Pi(\mathbf{v})} \mathbf{p}\mathbf{p}^T\right)}_{Q} \mathbf{v}$$

Metodo di Garland-Heckbert

- Per calcolare l'errore di un vertice basta mantenere la matrice Q (l'insieme dei piani è solo concettuale).
- Il costo di uno spigolo è l'errore $(\mathbf{v}_1,\mathbf{v}_2) = \bar{\mathbf{v}}^T (Q_1 + Q_2)\bar{\mathbf{v}}$, dove $\bar{\mathbf{v}}$ è la posizione del vertice risultante dalla contrazione di$(\mathbf{v}_1,\mathbf{v}_2)$
- Come scegliere $\bar{\mathbf{v}}$?
  - Scelta migliore:la posizione v che rende minima $\Delta(\mathbf{v})$ (si risolve con un sistema lineare).

Schema dell'algoritmo:

- 1. Calcola le matrici Q per tutti i vertici della maglia.
- 2. Per ciascuno spigolo $(\mathbf{v}_1,\mathbf{v}_2)$ calcola la posizione ottima per il vertice dopo l'ipotetica contrazione .
  - L'errore $\bar{\mathbf{v}}^T (Q_1 + Q_2)\bar{\mathbf{v}}$ `e il costo associato allo spigolo.
- 3. Costruisci uno heap con gli spigoli e chiave pari al costo.
- 4. Rimuovi lo spigolo $(\mathbf{v}_1,\mathbf{v}_2)$ di minor costo dalla cima dello heap, effettua la contrazione $(\mathbf{v}_1,\mathbf{v}_2)$ <- $\bar{\mathbf{v}}$
- 5. $Q_1 = Q_1 + Q_2$ e aggiorna i costi degli spigoli incidenti in $\mathbf{v}_1$
- 6. Ripeti da 4.
  - Bisogna prestare attenzione a non creare incoerenze nella maglia, come p.es. il fold-over.

Multirisoluzione

Un algoritmo di semplificazione come questo genera una piramide di modelli a complessità crescente, ovvero una rappresentazione multirisoluzione dell'oggetto.

- Si parte da un modello M_0 (maglia triangolare)
- Si semplifica il modello, ottenendone uno nuovo M_1 con meno triangoli
- Si itera il procedimento n volte, ottenendo una piramide di modelli $M_0 \ldots M_n$
  - Tale piramide può essere usata per svariati compiti
- Non archivio tutti i modelli, ma solo M_n e tutte le mosse di semplificazione inverse.

Multirisoluzione

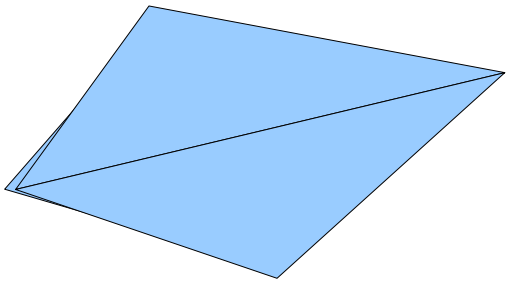
- Livello dinamico di dettaglio: utilizzo la piramide per variare la complessità dell'oggetto a seconda del punto di vista (inutile proiettare un modello di 5000 triangoli su 4 pixel).
- Compressione di una maglia: con una opportuna fase di codifica, lo spazio occupato da M_n e dalla lista di movimenti di ricostruzione è più piccolo dello spazio occupato da M_0.
- Rendering progressivo:se trasmetto il modello coarse e poi i dettagli posso fare un rendering progressivo, ovvero ottenere una prima immagine di M_n, e poi progressivamente "migliorarla" percorrendo la piramide dei modelli. Analogo a quanto siamo abituati a vedere nella trasmissione e visualizzazione di immagini JPEG su internet.

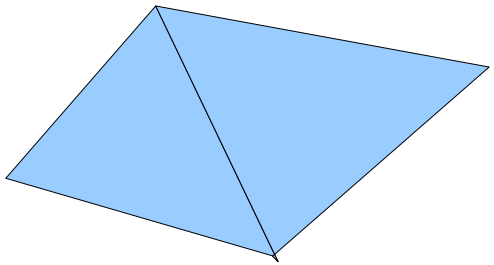
Ottimizzazione mesh

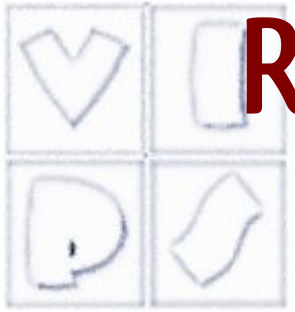
Miglioro il modello della maglia senza rimuovere o spostare i vertici, ma solo la connettività

Passo elementare: l'edge flipping (o edge swapping)

Il criterio di ottimizzazione può riguardare la minimizzazione degli angoli diedrali penalizzando triangoli adiacenti le cui normali sono troppo diverse.







Rappresentazioni geometriche alternative

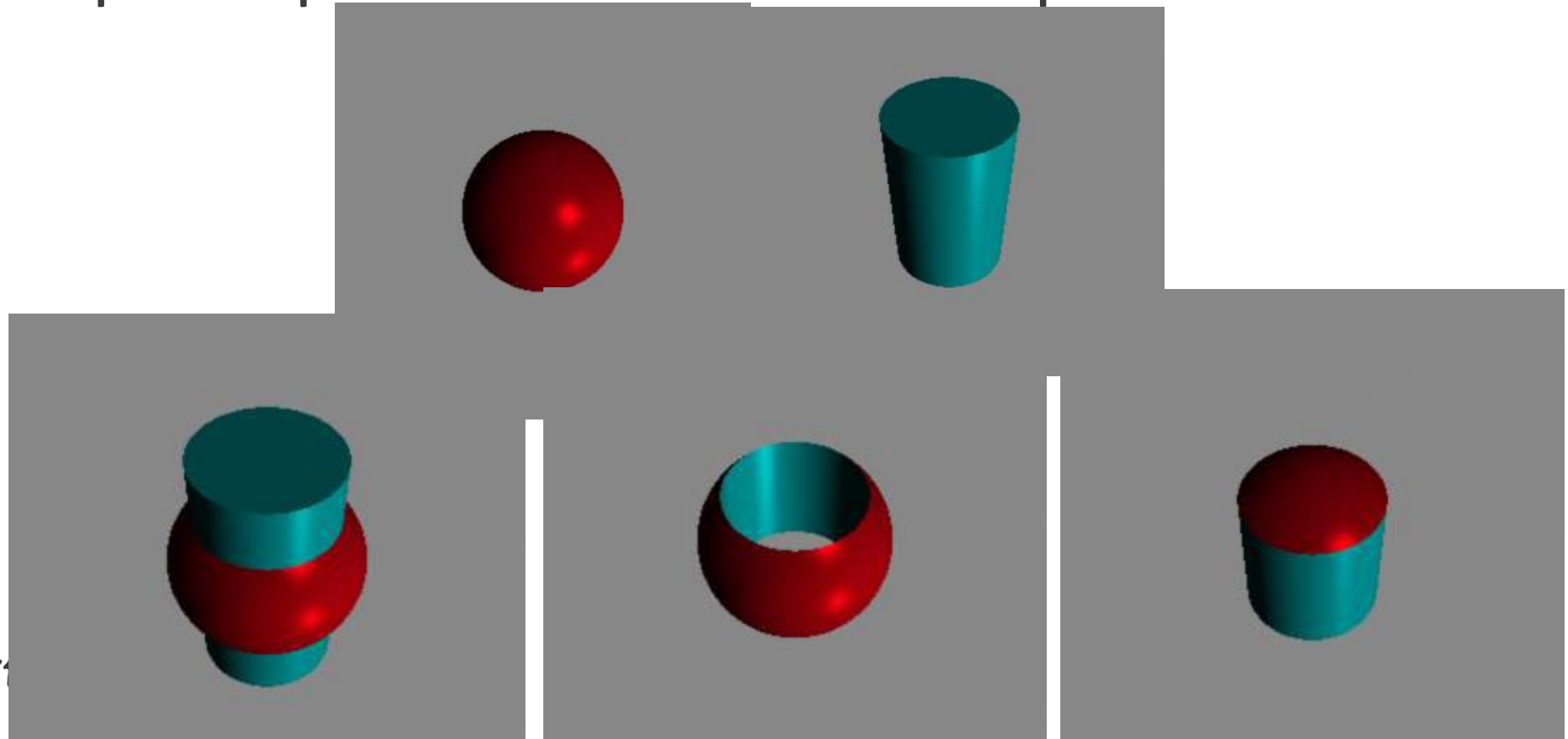
- Superfici parametriche
- Geometria costruttiva solida
- Volumi rasterizzati

Curve e superfici liscie

- Il vantaggio principale nell'uso di superfici per modellare un oggetto sarebbe l'assenza del problema della tessellazione visibile (ma in realtà dipende anche da come si fa il rendering)
- Fino a pochi anni fa l'uso di curve, o superfici liscie, era impensabile per applicazioni in tempo reale; per lo più venivano utilizzate in fase di modellazione o per rendering non interattivo
- Negli ultimi tempi le cose sono cambiate ed oggi cominciano ad apparire applicazioni di grafica avanzata che usano superfici curve anche in tempo reale
- Esistono vari tipi di superfici parametriche utili

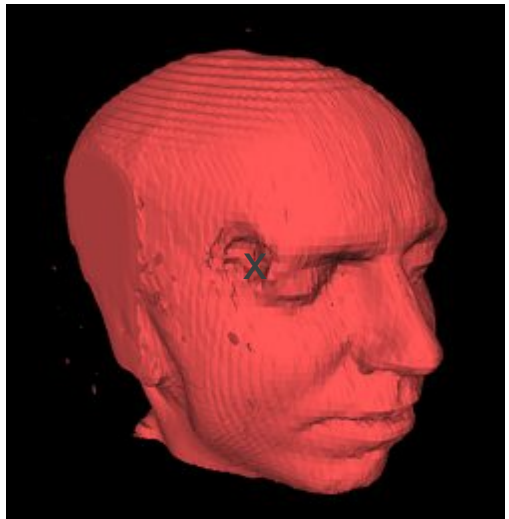
Geometria costruttiva solida

- 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



Partizionamento spaziale

- Lo spazio viene suddiviso in celle adiacenti dette in 3D voxel (equivalente dei pixel delle immagini): una cella è "piena" se ha intersezione non vuota con la regione, è detta vuota in caso contrario. Oppure contiene un valore di densità (tipico dei dati diagnostici es. TAC)
- Una rappresentazione di una scena complessa ad alta risoluzione richiederebbe l'impiego di un numero enorme numero di voxel, per cui questa rappresentazione è in genere limitata a singoli oggetti.
- Da una rappresentazione volumetrica voxelizzata si può passare efficientemente a una rappresentazione poligonale della superficie mediante l'algoritmo detto marching cubes.

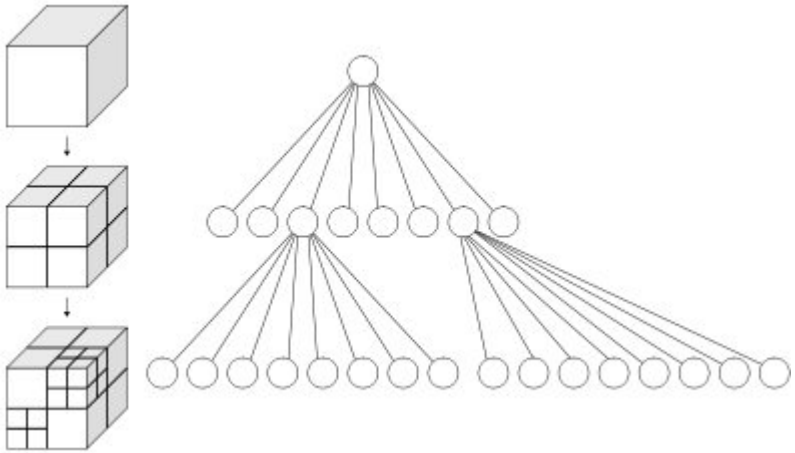


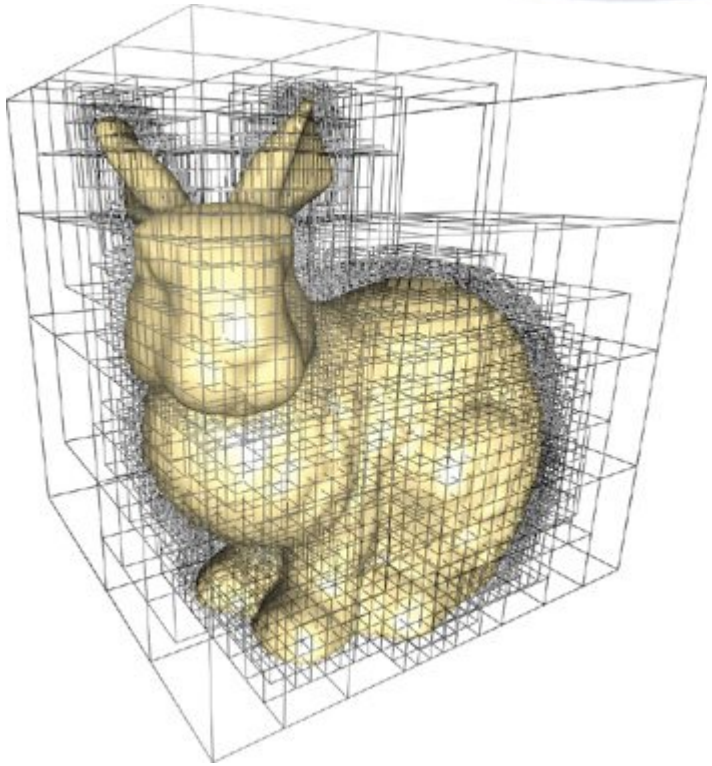
Rappresentazioni compatte

- Se ho solo i valori pieno/vuoto, posso rappresentare in modo compatto il volume con una struttura octree
- Si parte con un cubo contenente la regione e si suddivide ricorsivamente. Ci si ferma ogni volta che un ottante contiene tutte celle piene o tutte celle vuote.
- Più economica rispetto alla enumerazione delle singole celle, poiché grandi aree uniformi (piene o vuote) vengono rappresentate con una sola foglia (anche se nel caso peggiore il numero delle foglie è pari a quello delle celle).
- L'octree supporta in modo efficiente ricerche di intersezione voxel-raggio (per il ray-casting, vedremo).

Octree

- Naturalmente può servire anche come struttura di supporto per il calcolo delle intersezioni con mesh





Riferimenti

- Wikipedia voce polygon mesh
-

Domande di verifica

- Cos'è un simplesso?
- Cosa sono le relazioni di incidenza?
- Cosa si intende per varietà (2-manifold)
- Come si può ridurre la complessità di una mesh?