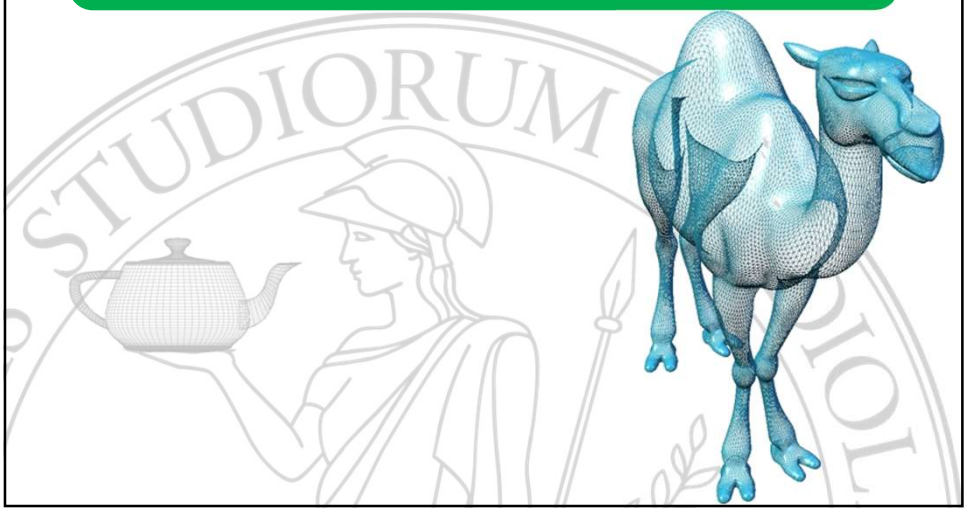
Marco Tarini - Computer Graphics 2025/2026
Università degli Studi di Milano

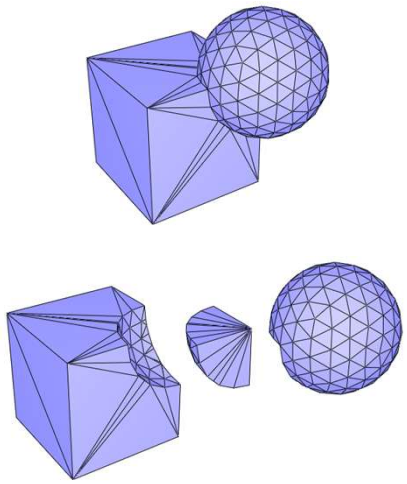
Mesh poligonali: Mesh Processing 2/2



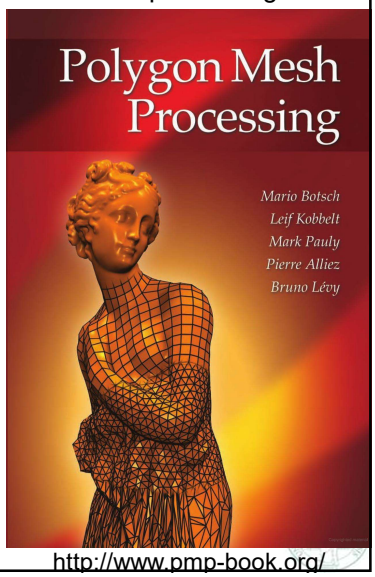
181

Mesh processing

Il **Geometry Processing** eseguito su **mesh poligonali**



Un buon manuale per le basi al mesh processing:



http://www.pmp-book.org/

182

Mesh Processing

- ✓ Un applicativo di mesh processing

MeshLab

183

Semplificazione automatica di una mesh

- ✓ Un applicativo generico di mesh processing:

MeshLab

- ✓ Mini-esercizio pratico: testa un algoritmo di mesh decimation
  - ⇒ Ottieni e apri MeshLab:
  - ⇒ Scarica una mesh ad alta risoluzione di esempio (ce ne sono anche sul sito)
  - ⇒ Applica uno di questi due filtri:
    filtro «quadric edge collapse decimation»
    filtro «clustering decimation»

due algoritmi che seguono approcci diversi, come abbiamo visto

184

Alcune librerie di mesh processing (C++, OpenSource)

VCG-Lib

vision and
computer graphic library

CNR ( )

CGAL

computational geometry
algorithms library

INRIA ( )

OpenMesh
+ OpenFlipper

RWTH ( )

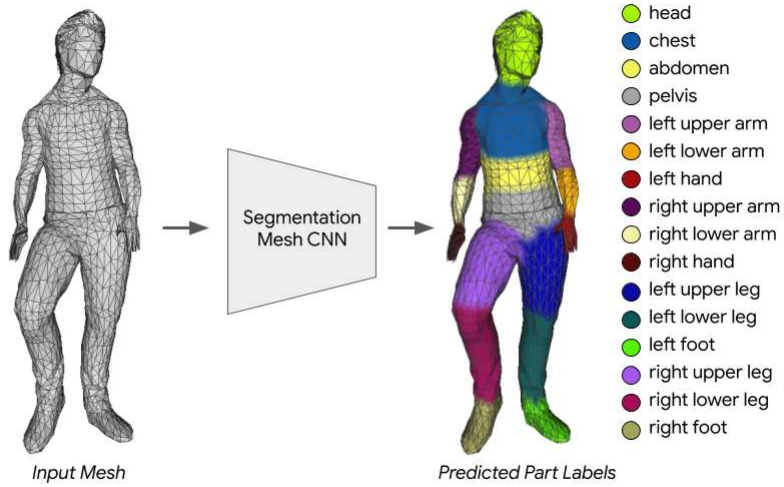
libigl

simple geometry
processing library

NYU ( )

185

Automatic mesh segmentation / labelling



186

Automatic mesh segmentation / labelling

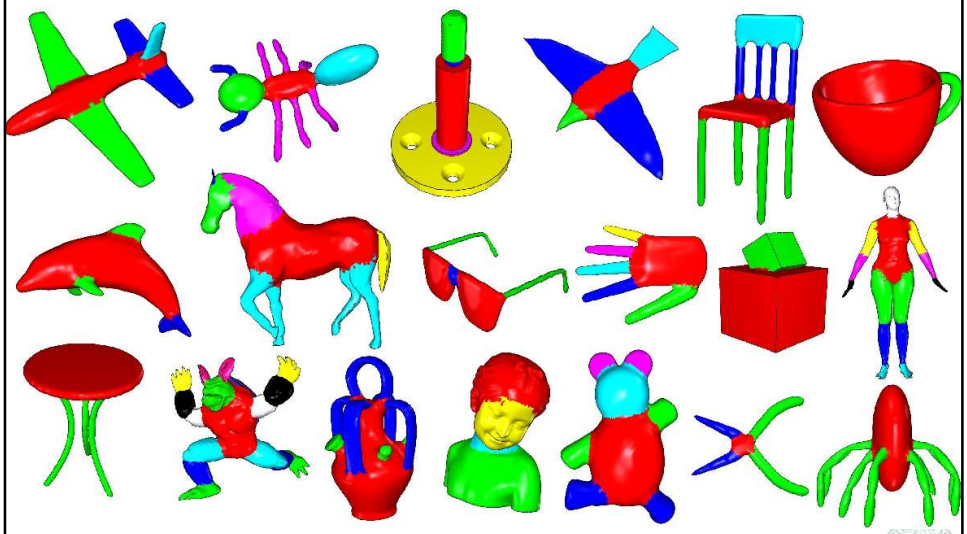


Image by Jiajun Lv, Xinlei Chen, Jin Huang, Hujun Bao

187

Automatic mesh segmentation / labelling

- ✓ Mesh segmentation (in generale):
  - ⇒ Data una mesh in input, identificare le zone semanticamente o geometricamente distinte
  - ⇒ (una partizione delle facce o dei vertici in zone contigue)
  - ⇒ Guidati da un'analisi della geometria, oppure data driven
- ✓ Mesh labelling
  - ⇒ Assegnare un'etichetta semantica ad ogni partizione
- ✓ Utilizzato come punto di partenza di molti altri task

188

Shape Retrival

Img by Tamy Boubekeur, and Marc Alexa, SIGGR 2010

189

Shape Retrival

- ✓ Il task:
  data una grossa collezione di modelli 3D, individuare quelli simili ad una forma target richiesta
  - ⇒ la forma target è specificata attraverso un disegno semplificato 2D (sketch-based shape retrieval)
  - ⇒ oppure “trovare le mesh che rappresentano oggetti simili a quello rappresentato da una mesh target data”.
  - ⇒ Oppure è specificata verbalmente, da un testo (prompt-based shape retrivial)

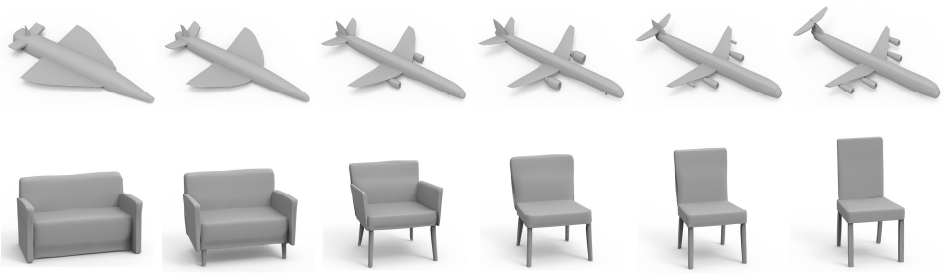
190

Procedural mesh generation

- ✓ Procedural mesh generation
  - ⇒ Nome generico per tutti quei procedimenti automatici che generano una mesh (geometria e connettività incluse)
  - ⇒ Utilizzato in videogames, VR, industria cinematografica, simulazioni…
  - ⇒ Molto studiati i casi di: piante, città, terreni, palazzi, manufatti di vario tipo, avatar umani, …
  - ⇒ Tipicamente, guidato da parametri controllati dall'utente che determinano le caratteristiche «ad alto livello» delle forme generate (ad esempio: nella generazione automatica di mesh che rappresentano ingranaggi, un parametro può identificare «il numero di denti») e / o da scelte pseudocasuali
- ✓ Generative Networks: approccio ML alla procedural generation
  - ⇒ Data driven, costruito da esempi
  - ⇒ Per es: data una collezione di mesh che rappresentano forme di una certa classe (sedie, aerei…), produrre un insieme di varianti
  - ⇒ Le mesh sono solo uno delle classi modelli 3D con cui si vuole far questo (si usano anche nuvole di punti, etc). Quale è la rappresentazione più adatta? (problema aperto

191

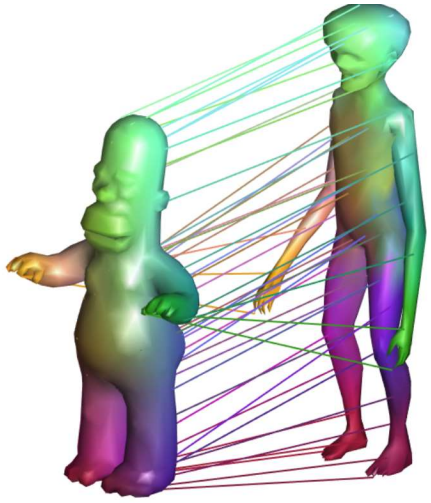
Generative Learning for Meshes



Img by Richard Zhang (2019)

192

Shape Correspondence



Img by Maks Ovsjanikov, Michael M. Bronstein, Emanuele Rodola

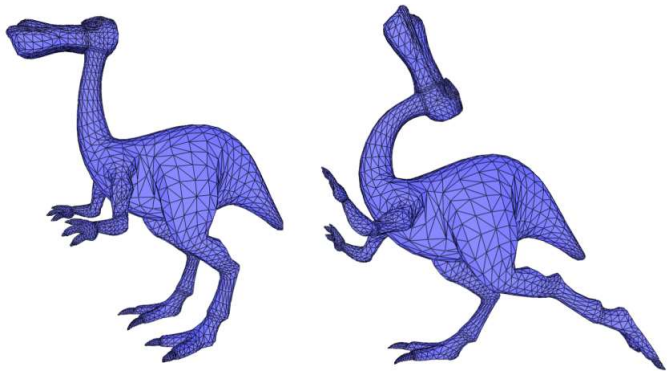
193

Shape correspondence

- ✓ Date due mesh che rappresentano oggetti di una stessa classe
  - ⇒ Per esempio, due forme umanoidi, o due quadrupedi
  - ⇒ Oppure, la scansione di uno stessa persona in due posizioni differenti…
- ✓ Identificare su ciascuna di essa il punto corrispondente sull'altra
  - ⇒ Ad esmepio, sotto forma di un mapping (non biunivoco) fra i vertici
- ✓ Basandosi su similitudine geometrica, topologica, e caratteristiche intrinseche della forma
  - ⇒ (ad esempio: la similarità di normale è poco significativa ma quella di curvatura intrinseca può esserlo di più)

194

Mesh Deformation



Img by Sébastien Loriot, Olga Sorkine-Hornung

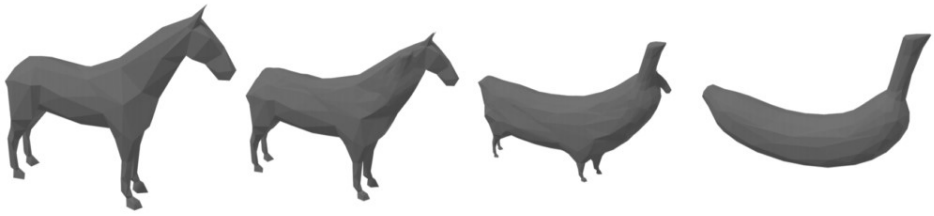
195

Mesh Deformation

- ✓ Data una mesh iniziale soggetta a degli stimoli o vincoli esterni, computare una sua deformazione spaziale
  - ⇒ cioè una nuova geometria, (la connettività della mesh è inalterata)
  - ⇒ La forma finale aderisce a princìpi fisici, geometrici (per es, conservazione dell'area, del volume, della forma dei triangoli, o semplicemente minimizzazione della distorsione subita dai singoli triangoli…)
  - ⇒ Esempio di vincolo esterno: una nuova la posizione xyz assegnata a solo alcuni vertici («questo vertice si sposta qui»)
- ✓ Utilizzata in animazione, design
  - ⇒ E' possibile effettuare questi computi in tempo reale

196

Mesh Morphing

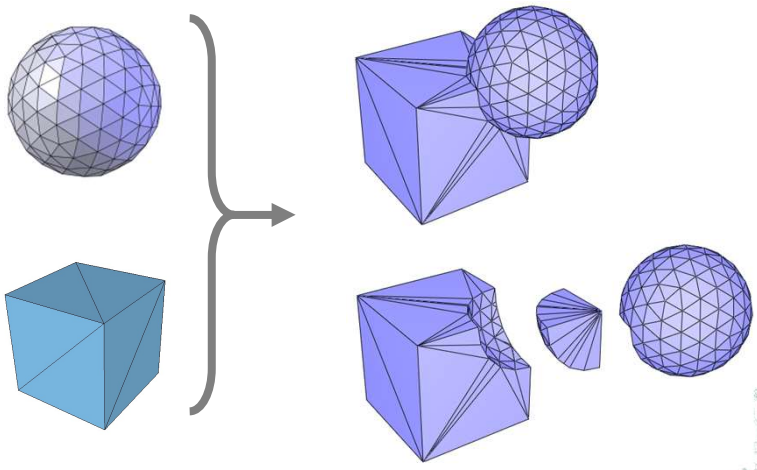


Img by Kuba Perlin

197

“Operazioni booleane” su mesh

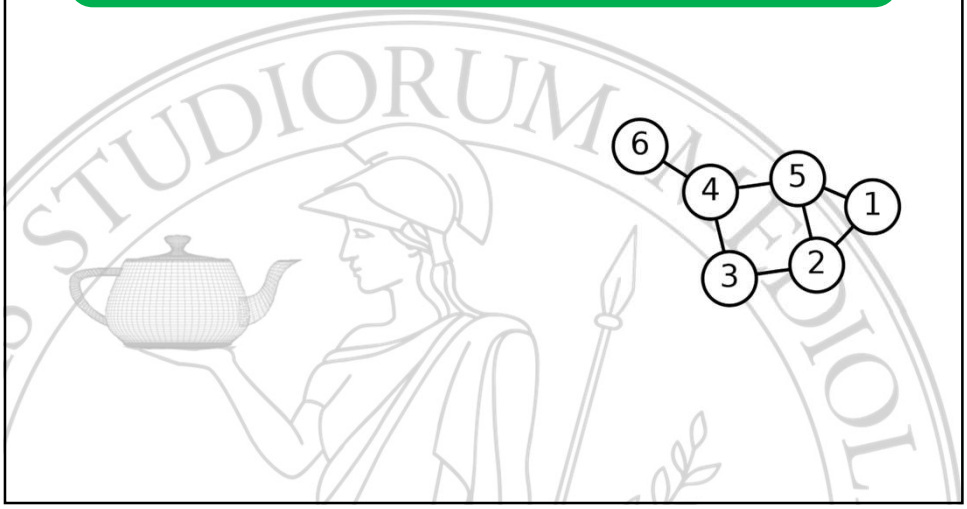
Date due (o più) mesh **ben orientate** e **chiuse**,
(che quindi delimitano un volume interno ciascuna)
torvare la mesh che delimita l’intersezone (“AND”),
l’unione (lo “OR”), etc, fra i due volumi



198

Marco Tarini - Computer Graphics 2025/2026
Università degli Studi di Milano

Mesh Processing 3/3: algoritmi e strutture dati



200

Mesh Processing: le basi algoritmche

- ✓ Gli esempi visti (e altri che vedremo) sono solo alcuni dei molti task affrontati nel contesto del mesh processing.
- ✓ Gli algoritmi che risolvono questi task necessitano (quasi tutti) operazioni base comuni sulla connettività (operazioni di navigazione su mesh) come:
  - ⇒ Data una faccia, processa tutte le facce **adiacenti** (cioè quelle divise da un edge)
  - ⇒ Dato una faccia ed un edge, trova la faccia (se esiste) che sta dall'altro lato di quell'edge
  - ⇒ Dato un vertice, elenca tutte le facce che includono quel vertice (detta la «**stella**» del vertice)
  - ⇒ Dato un vertice, enumera tutti i vertici che sono connessi a quel vertice da un edge
  - ⇒ Dato un edge, determina se si trattta di un edge di bordo.
  - ⇒ Dato un edge di bordo, enumera tutti gli altri edge che fanno parte dello stesso quel bordo della mesh
  - ⇒ Dato una faccia, elenca tutti i vertici che fanno parte di quella faccia
- ✓ E' necessario che queste operazioni basilari siano effettuate efficientemente (idealmente, in tempo costante)

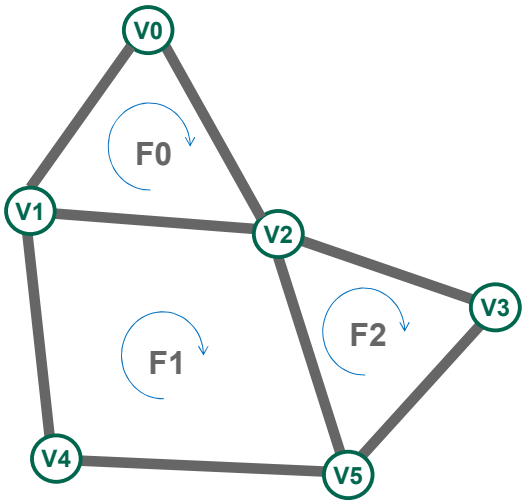
202

Strutture dati per Mesh Processing

- ✓ La struttura «array di facce» della mesh indexed, usata per memorizzare la connettività, non consente di effettuare queste operazioni in modo efficace
  - ⇒ se non attraverso una approccio forza bruta: scansione dell'intero vettore delle facce, che richiede ovviamente un tempo lineare col numero di facce – questo rende quadratico (quindi proibitivo) il costo computazionale del task affrontato
  - ⇒ eccetto una operazione: «data una faccia, processa tutti i vertici che fanno parte di quella faccia»
- ✓ Per effettuare mesh processing, dobbiamo dotarci di strutture più adatte per memorizzare la connettività di una mesh
  - ⇒ che consentano di «navigare» sulla mesh molto più agevolmente
- ✓ Vediamo la più diffusa e flessibile di queste strutture: l'array di half-edge

203

Struttura dati per la connettività: array di facce

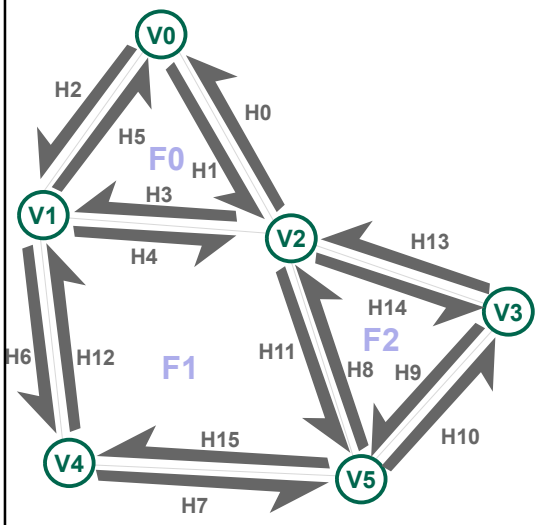


$F_0 \rightarrow$ 0 , 2 , 1
$F_1 \rightarrow$ 1 , 2 , 5 , 4
$F_2 \rightarrow$ 2 , 3 , 5

CONNETTIVITA'
(come lista facce)

204

Array di half-edge



	V	F	Next	Opp
H_0 →	2	–	2	1
H_1 →	0	0	3	0
H_2 →	0	–	6	5
H_3 →	2	0	5	4
H_4 →	1	1	11	3
H_5 →	1	0	1	2
H_6 →	1	–	7	12
H_7 →	4	–	10	15
H_8 →	5	2	14	11
H_9 →	3	2	8	10
H_{10} →	5	–	13	9
H_{11} →	2	1	15	8
H_{12} →	4	1	4	6
H_{13} →	3	–	0	14
H_{14} →	2	2	9	13
H_{15} →	5	1	12	7

205

Array (o lista) di half-edge

- ✓ Idea: posso rappresentare la connettività di una mesh come un array di Half-Edge
- ✓ Un half-edge è un «edge orientato»
  - ⇒ E' detto half-edge perché rappresenta "la metà di un edge": ogni edge della mesh è composto da due half-edge in direzioni opposte (come una strada a due corsie è composta da due corsie)
- ✓ Un half-edge è una struttura dati che consiste dei campi:
  - ⇒ Indice di Vertice: da quale vertice parte quell'half-edge
  - ⇒ Indice di Faccia: di quale faccia è un bordo
  - ⇒ Next: l'indice dell'half-edge che incontro proseguendo nella direzione dell'half-edge (senza cambiare faccia o bordo della mesh)
  - ⇒ Opposite: indice all'altro half-hege che condivide lo stesso edge
- ✓ Strutture a contorno:
  - ⇒ Per ogni vertice, memorizzo l'indice di un half-edge che parte da quell vertice (uno qualsiasi)
  - ⇒ Per ogni faccia, memorizzo l'indice di un half-edge appartenente a quella faccia (uno qualsiasi)

210

HalfEdge: pseudocodice Java

```
class HalfEdge {
    int vi;   // indice di vertice
    int fi;   // indice di faccia (o -1)
    int next; // indice di halfHedge
    int opp;  // indice di halfHedge
}

// la tabella
HalfEdge[] he = new HalfEdge( n );
```

ed es, il valore *opposite* del halfedge di indice 12 è…

```
he[12].opp;
```

ed es, l'half edge di indice 7 fa parte della faccia…

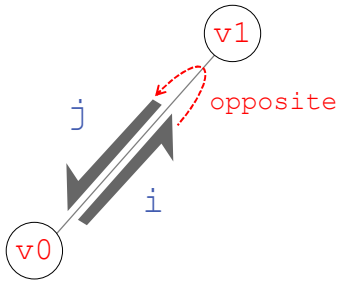
```
he[7].fi;
```

211

Esempio di uso base

✓ dato un half-edge di indice `i`, quali sono i due vertici `v0` e `v1` che delimitano l'edge corrispondente?

```
int v0 = he[i].vi;
int j = he[i].opp;
int v1 = he[j].vi;
```



212

Strutture dati per connettività a confronto

- ✓ Array di facce (anche nota come «Lista facce»):
  - ⇒ Compatta
    (quindi, adatta a storing, ad esempio su disco o streaming)
  - ⇒ Sufficiente per ad alcuni task di processing
    (e in questo caso, preferibile per la sua leggerezza)
  - ⇒ Il **rendering** eseguito dalle GPU
    è pensato per questa struttura dati
  - ⇒ E' generale: non richiede ad esempio di assumere che la mesh sia two-manifold o che abbia l'orientamento consistente delle facce
    (quindi: capace di rappresentare strutture inconsistenti – è un vantaggio e uno svantaggio)
  - ⇒ Lista di elementi non omogenea
    (alcune facce hanno un numero di vertici maggiore da altre).
    eccetto che per **tri-mesh** o **pure quad meshes**:
    per loro, la lista facce è comodamente una matrice 3xN o 4xN di interi

214

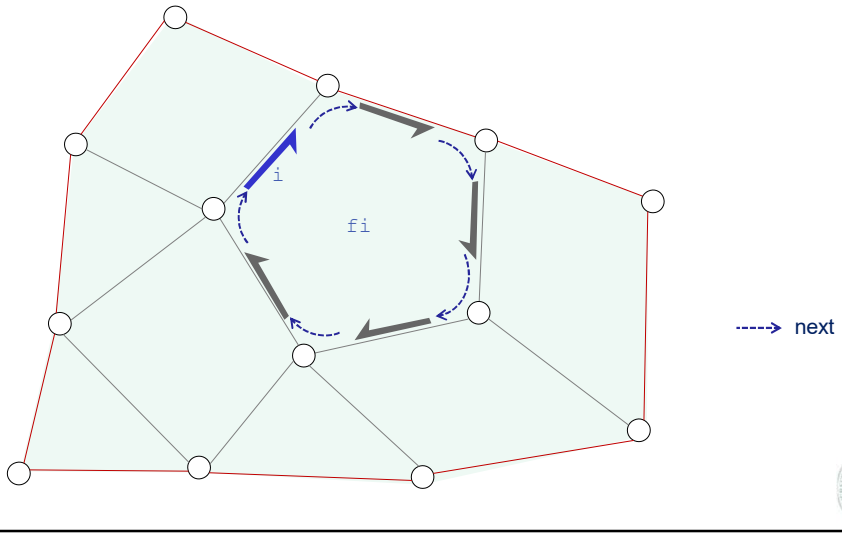
Strutture dati per connettività a confronto

- ✓ Array (o lista) di half-hedge:
  - ⇒ Prolissa
    (quanti interi bisogna memorizzare in media, rispetto ad una lista facce?
    Ipotesi: in una tri-mesh, ho vertici , facce, ed edge tipicamente in proporzione 1x, 2x, 3x. In una quad mesh: 1x, 1x, 2x.
    Quindi, in una tri mesh: 3 interi per tri. Half edge: 12 interi!)
  - ⇒ Più complicata da mantenere coerente durante le operazioni di modifica della connettività
  - ⇒ Non adatta per il rendering (su GPU)
  - ⇒ Vantaggio: lista di elementi sempre omogenea: 4 elementi per half-edge (nella variante che abbiamo visto), anche su mesh poligonali miste
  - ⇒ Consente di «navigare sulla mesh», con salti all'elemento adiacente in tempo costante (consentendo algoritmi di mesh processing in tempo lineare o pseudolineare piuttosto che quadratico)
  - ⇒ Richiede adattamenti se la mesh non è two-manifold e ben orientata
- ✓ Nota: sono due rappresentazioni alternative di una stessa cosa (la connettività della mesh).
  - ⇒ Una si può costruire a partire dall'altra

215

Esempio: conta quanti lati ha una faccia

✓ Dato un half-edge di indice `i` (adiacente ad una faccia `fi`), trova quanti `lati` ha questa faccia



217

Esempio: conta quanti lati ha una faccia

✓ Dato un half-edge di indice `i` (adiacente ad una faccia `fi`), trova quanti `lati` ha questa faccia

(era un half edge di bordo)

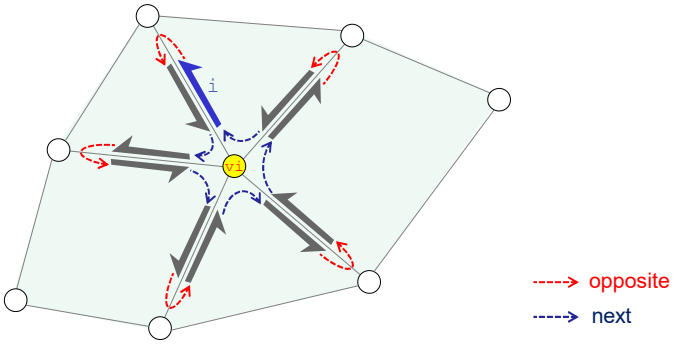
```
int fi = he[i].fi;
if (fi == -1) … /* non esiste la faccia */
int start = i; // half edge di partenza
int lati = 0;
do {
   lati ++;
   i = he[i].next;
} while (i!=start);
```

(nota: il ciclo finisce quando torno al punto di partenza)

218

Esempio: scansione di vertici attorno ad un vert

✓ Dato un half-edge di indice `i`, che emana da un vertice `vi` , trova tutti i vertici `vj` nella stella di `vi`



(nota: il ciclo finisce quando torno al punto di partenza)

219

Esempio: scansione di vertici attorno ad un vert

✓ Dato un half-edge di indice `i`, che emana da un vertice `vi` , trova tutti i vertici `vj` nella stella di `vi`

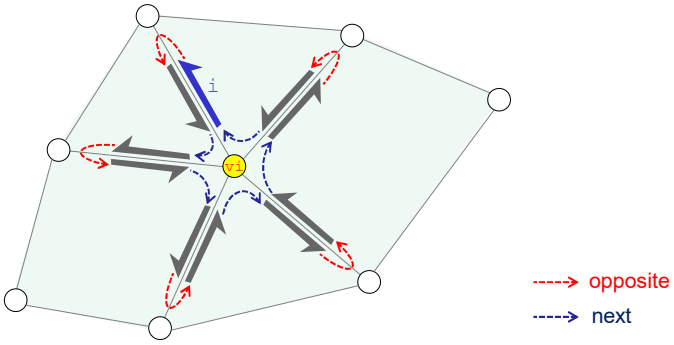
```
int vi = he[i].vi;

int start = i; // half edge di partenza

do {
   i = he[i].opp;
   int vj = he[i].vi;

   /* qui: fai qualcosa con vertice vj */

   i = he[i].next;
} while (i!=start);
```

(nota: il ciclo finisce quando torno al punto di partenza)

220

Scansione di facce attorno ad un vert

✓ Dato un half-edge di indice `i`, che emana da un vertice `vi` , trova tutte le faccie `fj` adiacenti a `vi`



(nota: il ciclo finisce quando torno al punto di partenza)

221

Esempio: scansione di facce attorno ad un vert

✓ Dato un half-edge di indice `i`, che emana da un vertice `vi` , trova tutte le faccie `fj` adiacenti a `vi`

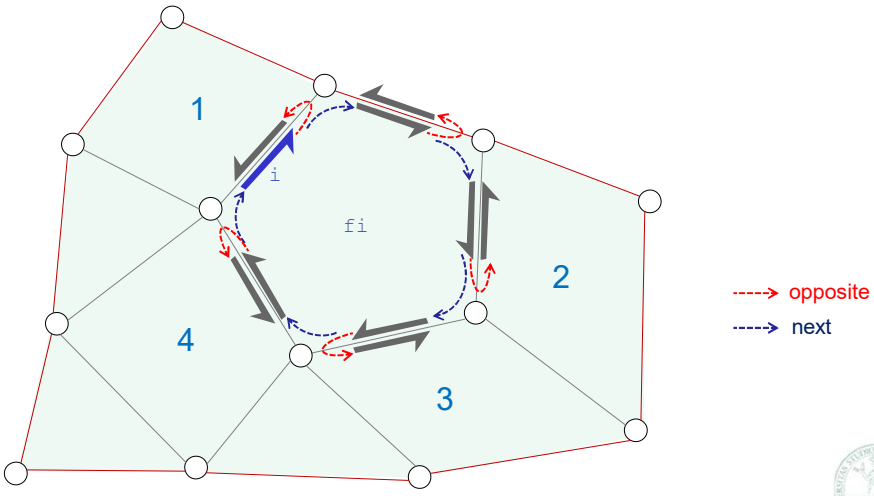
```
int vi = he[i].v;

int start = i; // half edge di partenza

do {
   int fi = he[i].fi;
   /* qui: fai qualcosa con faccia fi */
   /* se esiste: potrebbe essere -1 */

   i = he[i].opp;
   i = he[i].next;

} while (i!=start);
```

(nota: il ciclo finisce quando torno al punto di partenza)

222

Esempio: contare la faccie adiacenti da una faccia

✓ dato un indice di halfedge `i`, se confina con una faccia `fi`, allora conta `quante_facce` adiacenti a `fi` ci sono



223

Esempio: contare la faccie adiacenti da una faccia

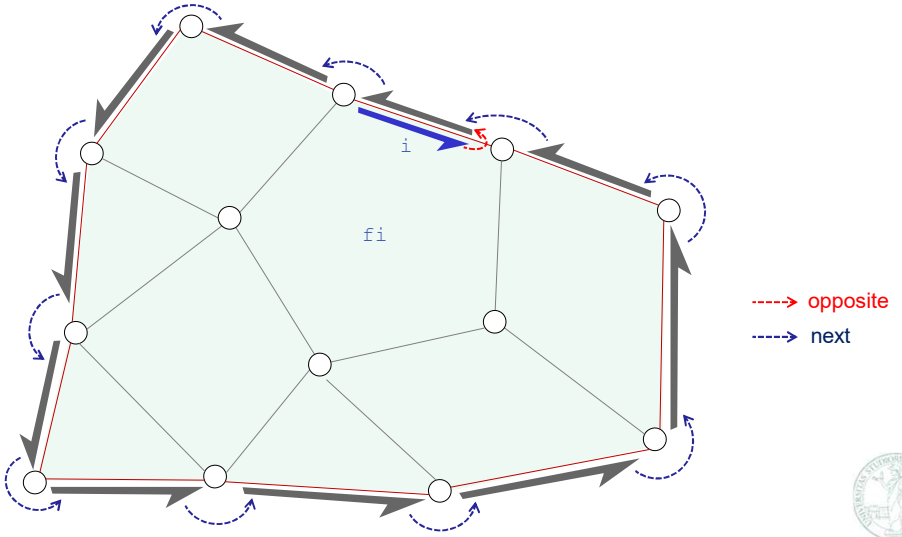
✓ dato un indice di halfedge `i`, se confina con una faccia `fi`, allora conta `quante_facce` adiacenti a `fi` ci sono

```
int fi = he[i].f;
if (fi == -1) … /* non esiste la faccia */
int start = i;
int quante_facce = 0;
do {
    int j = he[i].opp;
    if (he[j].fi!=-1) quante_facce ++;
    i = he[i].next;
} while (i!=start);
```

224

Esempio: visita di un bordo (insieme di edge)

✓ Sia dato un indice di halfedge `i`, confinante con una faccia `fi`;
se l'edge corrispondente è di bordo, allora scandici tutti gli edge che fanno parte dello stesso bordo:



225

Esempio: visita di un bordo (insieme di edge)

✓ Sia dato un indice di halfedge `i`, confinante con una faccia `fi`;
se l'edge corrispondente è di bordo, allora scandici tutti gli edge che fanno parte dello stesso bordo:

```
i = he[i].opp;
int fi = he[i].f;
if (fi == -1) {
     int start = i;
     do {
          int i = he[i].next;
          /* fa qualcosa con i... */
     } while (i!=start);
}
```

226