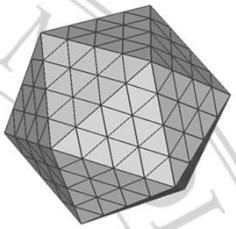
Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

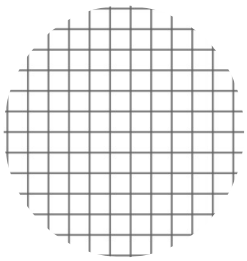
Mesh poligonali: regolarità



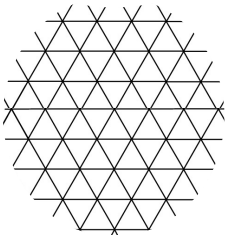
148

Regolarità di una mesh: concetto

- ✓ Osservazione: un piano può essere tassellato in modo **regolare** da quadrati ==>
- ✓ Tanto più la *connettività* di una **quad-mesh** si avvicina a questo caso, tanto più la consideriamo "**regolare**"

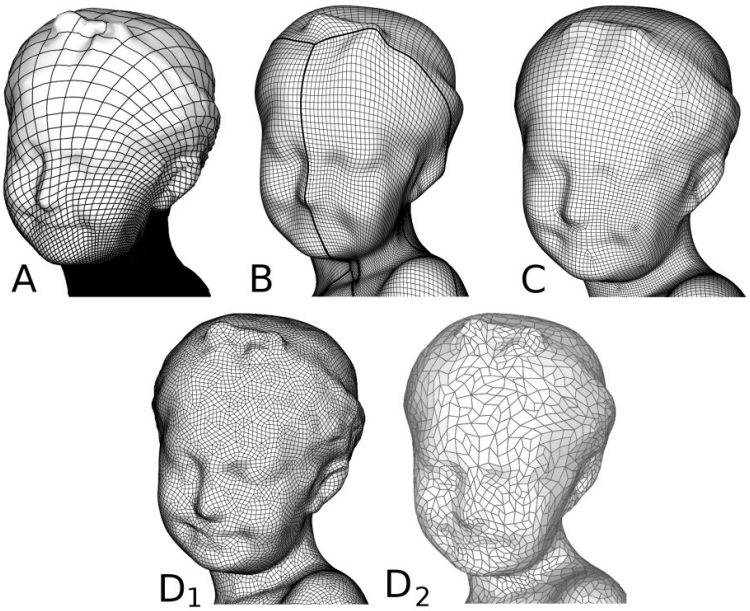


- ✓ Osservazione: un piano può essere tassellato in modo **regolare** da triangoli equilateri ==>
- ✓ Tanto più la *connettività* di una **tri-mesh** si avvicina a questo caso, tanto più la consideriamo "**regolare**"



149

Mesh: regolarità



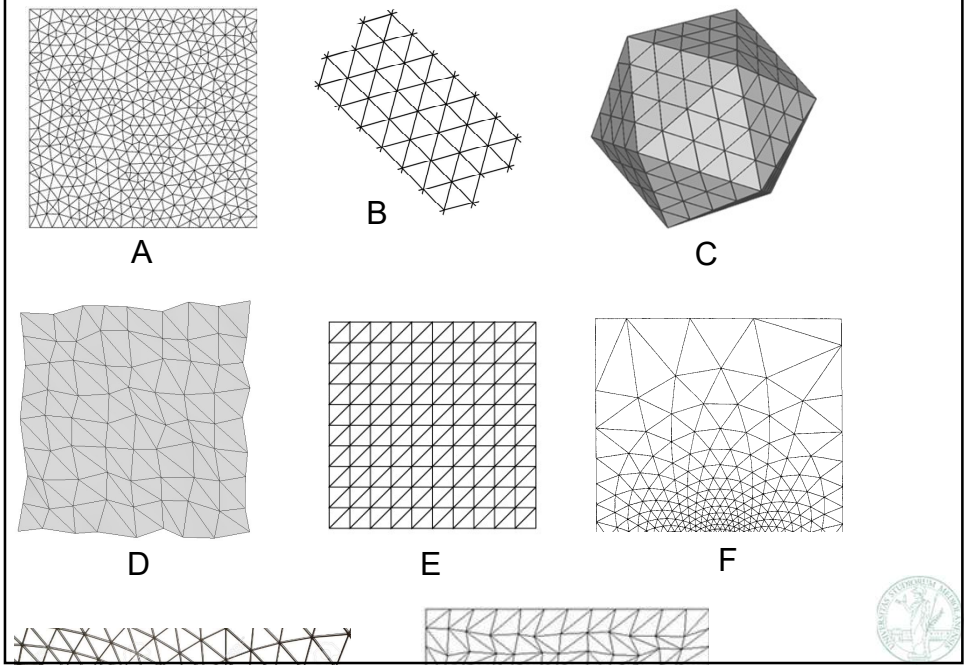
150

Regolarità di una mesh definizioni

- ✓ Vertice interno = vertice che non compare su un edge di bordo
- ✓ «Valenza» edge di un vertice
  - ⇒ numero di edge adiacenti ad quel vertice
- ✓ «Valenza» faccia di un vertice
  - ⇒ numero di facce adiacenti ad quel vertice
  - ⇒ per un vertice interno: le due valenze sono uguali
- ✓ «Vertice regolare» interno = un vertice di valenza…
  - ⇒ …4, in una quad-mesh
  - ⇒ …6, in una tri-mesh
- ✓ Quanti vertici in una mesh (tri o quad) sono regolari?
  - ⇒ Tutti => la mesh è (*perfettamente)* regolare (è detta anche «structured» mesh)
  - ⇒ Quasi tutti - per es il 99% => la mesh è «semi-regolare»
  - ⇒ Pochi (per es 2/3 o la metà) = è una mesh irregolare
  - ⇒ Nota: la regolarità di una mesh è una questione di grado
- ✓ Quad-mesh regolare = è un grigliato

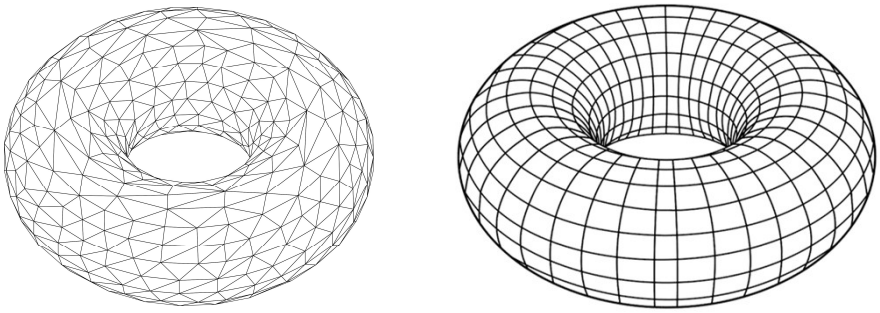
152

Regular VS irregular Triangular mesh



155

Regolare VS irregolare



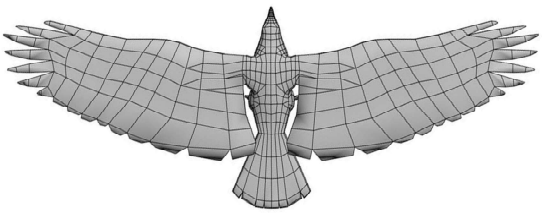
una tri-mesh
irregolare

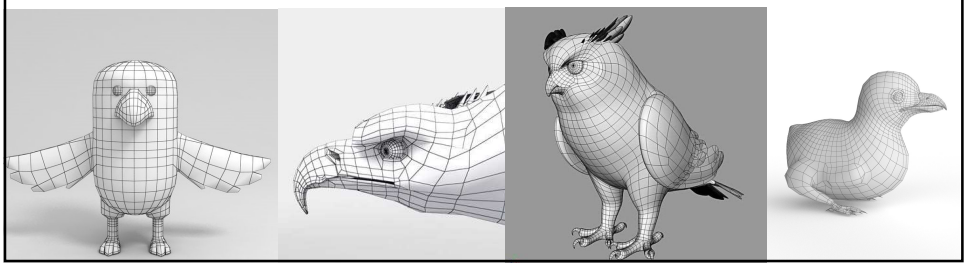
una quad-mesh
regolare

156

Regolarità delle mesh sul campo

✓ Le mesh modellate manualmente (nel CAD, nel direct poly editing [vedi dopo]) tendono ad essere quad-mesh **regolari**





157

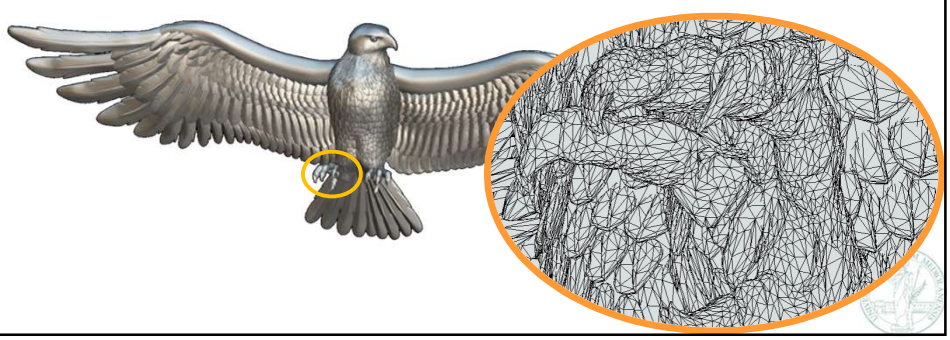
Regolarità delle mesh sul campo

✓ Sono di solito tri-mesh **irregolari** le mesh

⇒ **acquisite** 3D da modelli reali (per es, scanning o fotogrammetria)

⇒ prodotte o ri-processate attraverso tecniche di **geometry processing** per esempio tutti quelli visti: semplificazione automatica, front advancing methods, triangolazioni di Delaunay, vertex clustering…
(Eccezione: tecniche preposte a questo specifico scopo come il semi-regular remeshing)

⇒ Prodotte da artisti digitali con tecniche di **digital sculpting** (vedi sotto)



158

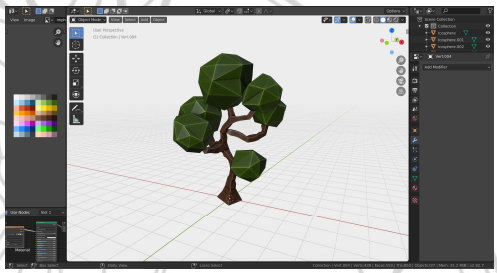
Molti vantaggi (e alcuni svantaggi) della regolarità

- ✓ Una mesh più regolare è
  - ⇒ Più agevole da modificare / o ri-editare manualmente (ad esempio, consente la selezione di stream di edge)
  - ⇒ Più adatta da animare
  - ⇒ Più efficiente da comprimere per risparmiare memoria (o da mandare in streaming)
  - ⇒ Maggiormente adatta ad applicazioni di machine learning
  - ⇒ Spesso maggiormente accurata geometricamente, a parità di risoluzione
  - ⇒ Meno soggetta a presentare artefatti di rendering
  - ⇒ Se è una quad-mesh, un vantaggio ulteriore consiste nell'avere gli edge allineati alle direzioni principali di curvatura
  - ⇒ Tuttavia, la risoluzione di una mesh regolare tende ad avere una risoluzione meno adattiva: l'adattività «si paga» in termini di regolarità
- ✓ In generale, la regolarità mesh è una caratteristica desiderabile
  - ⇒ ma spesso difficile da ottenere in modo automatico
  - ⇒ per l'efficienza di rendering, non fa differenza
  - ⇒ Q: cosa ottengo effettuando un diagonal-split a tutte le facce di una quad-mesh **regolare**?

159

Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

Modelli poligonali: alcuni cenni di modellazione manuale



165

Come vengono generate le mesh

- ✓ Cattura dalla realtà
  - ⇒ Cioè 3D Acquisition
  - ⇒ Per es, laser scanning, fotogrammetria…
- ✓ Generazione procedurale
  - ⇒ Per es, modelli di piante o vegetali generati attraverso una simulazione della loro cresita
- ✓ Simulazione
  - ⇒ Per es, una simulazione fisica di un liquido per generare la sua superficie in movimento
- ✓ Modellazione manuale
  - ⇒ Da parte di artisti digitali

166

Authoring delle mesh (cioè 3D modelling)

- ✓ Alcuni dei software più diffusi

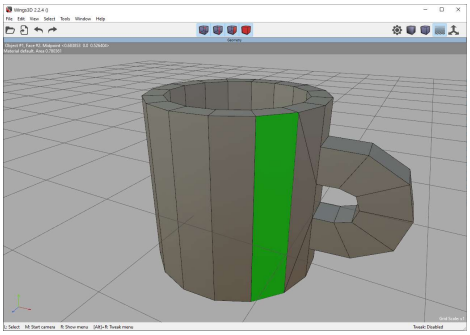
- ✓ **3D Studio Max** (autodesk) , **Maya** (autodesk) , **Cinema4D** (maxon) **Lightweight 3D** (NewTek), **Modo** (The Foundry) , **Blender** (open source!) ,
  - ⇒ all-purpose,completi
- ✓ **AutoCAD** (autodesk), **SolidWorks** (SolidThinking)
  - ⇒ Specializzati per il CAD
- ✓ **ZBrush** (pixologic), **Mudbox** (autodesk)
  - ⇒ Specializzati per lo scuplting
- ✓ **Wings3D**
  - ⇒ Solo low-poly modelling (& superfici di suddivisione) open-source, piccolo, specializzato
- ✓ **Sculptris Alpha**
  - ⇒ Solo sculpting open-source, piccolo, specializzato
- ✓ **[Rhinoceros]**
  - ⇒ parametric surfaces (NURBS)
- ✓ **FragMotion**
  - ⇒ small, specialized on animated meshes
- ✓ + molti altri con scopi più specifici
  - ⇒ editing of human models, of architectural interiors, environments, or specific editors for game-engines, etc...

167

Generazione manuale di mesh (cenni)

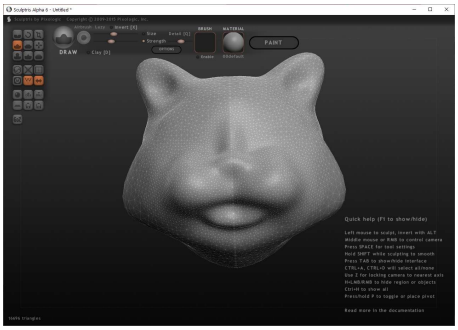
✓Due paradigmi di modellazione manuale:

Direct-poly modelling



esempio con Wings3D

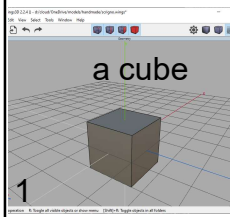
Digital Sculpting

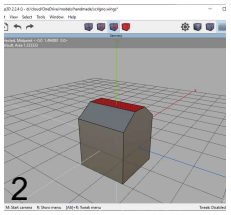


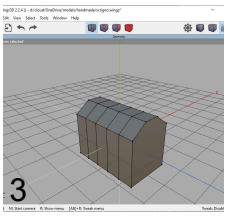
esempio con Sculptris Alpha

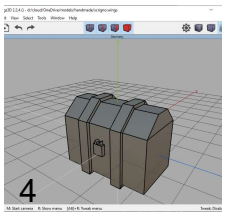
168

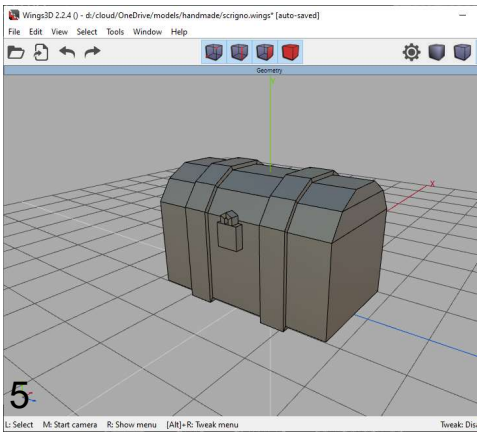
Direct Low-poly Modelling











with wings

169

Direct (Low-poly) Modelling: note

- ✓ Il modellatore umano manipola direttamente, attraverso un'apposita interfaccia, gli elementi della mesh
  - ⇒ come facce, edge, vertici
- ✓ L'operatore edita esplicitamente
  - ⇒ La connettività della mesh (per es, quali edge connettono quali vertici)
  - ⇒ La posizione dei vertici (singolarmente, o a piccoli gruppi)
- ✓ Le operazioni mantengono automaticamente two-manifold-ness (e a volte la chiusura) delle mesh
- ✓ Vengono usati operazioni di suddivisione (vedi più avanti)
- ✓ Tipico risultati: mesh poligonali, spesso quad, spesso semi-regolari, a bassa risoluzione

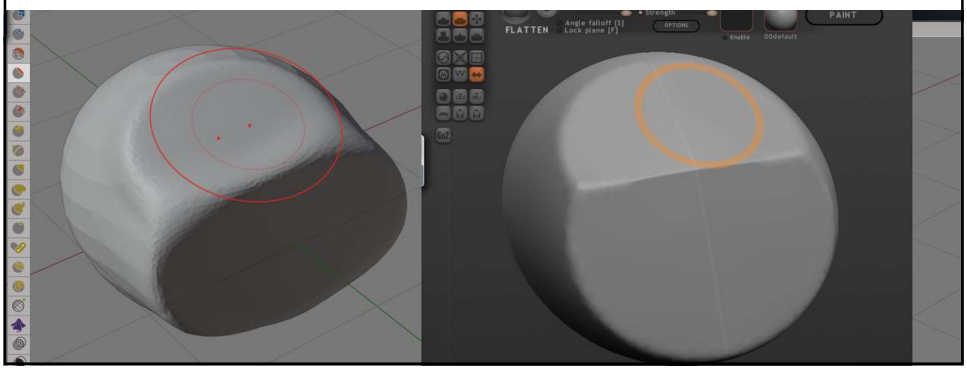
170

Digital Sculpting

mouse
(or stylus)

=

chisel



171

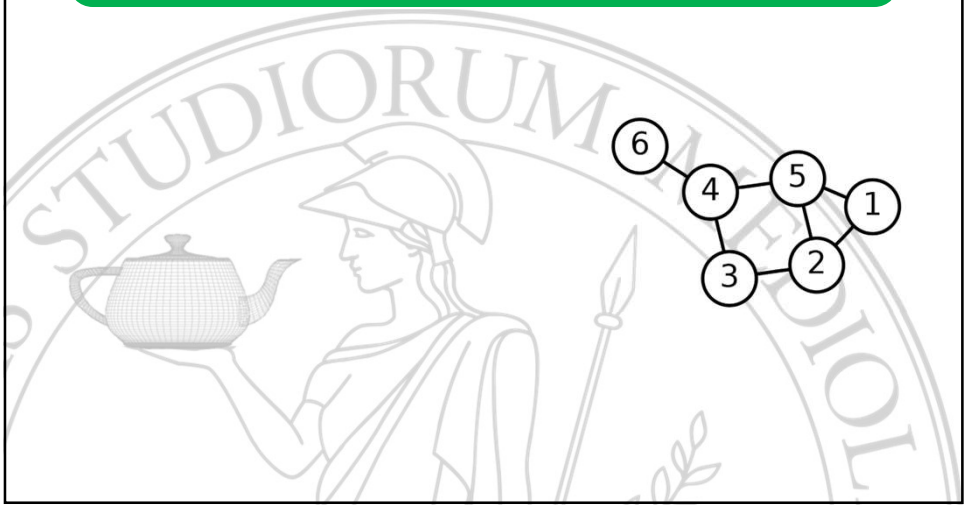
Digital sculpting: note

- ✓ Artista scolpisce la forma con pennellate («brush strokes») che imitano la manipolazione di materiale plasmabile come creta / pongo / etc
  - ⇒ Ma ovviamente senza alcun vincolo fisico: per es, è possibile rimuovere o creare materiale a piacere
- ✓ La rappresentazione interna della mesh (facce, vertici, edge) non è esposta al modellatore, ma viene gestita internamente
- ✓ «Pennelli» (anzi, scalpelli) digitali con effetti diversi, per es, local smoothing (rendere più liscio), appiattire, estrudere, «pizzicare», scavare, etc
- ✓ Sistema gestisce il meshing automaticamente in background,
  - ⇒ per es effettuando edge-collapse o edge split (l'operazione inversa) per mantenere una risoluzione adattiva adatta alla forma che viene generata
- ✓ Maggiormente adatta per modelli biologici / naturali
- ✓ Tipico risultato: una tri-mesh irregolari ad alta risoluzione

172

Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

Mesh Processing: algoritmi e strutture dati (cenni)



173

Geometry Processing: le basi algoritmche

- ✓ Gli esempi visti sono solo alcuni dei molti esempi di task affrontati dal geometry processing.
- ✓ Gli algoritmi che risolvono questi task necessitano (quasi tutti) operazioni base sulla connettività come:
  - ⇒ Data una faccia, enumera le facce adiacenti
  - ⇒ Dato un vertice, elenca tutte le facce che includono quel vertice
    Dato un vertice, enumera tutti i vertici che sono connessi a quel vertice da un edge (la «stella» del vertice)
  - ⇒ Dato una faccia, elenca tutti i vertici che fanno parte di quella faccia
  - ⇒ Dato un edge, scopri se è un edge di bordo.
  - ⇒ Dato un edge di bordo, enumera tutti gli altri edge che fanno parte di quel bordo

174

Mesh Processing: le basi algoritmche

- ✓ Gli esempi visti sono solo alcuni dei molti esempi di task affrontati nel contesto del mesh processing.
- ✓ Gli algoritmi che risolvono questi task necessitano (quasi tutti) operazioni base sulla connettività come per esempio:
  (operazioni di navigazione su mesh)
  - ⇒ Data una faccia, enumera tutte le facce adiacenti (separate da un edge)
  - ⇒ Dato una faccia ed un edge, trova la faccia (se esiste) che sta dall'altra faccia di quell'edge
  - ⇒ Dato un vertice, elenca tutte le facce che includono quel vertice (detta la «stella» o «stella-1» del vertice)
  - ⇒ Dato un vertice, enumera tutti i vertici che sono connessi a quel vertice da un edge
  - ⇒ Dato un edge, scopri se è un edge di bordo.
  - ⇒ Dato un edge di bordo, enumera tutti gli altri edge che fanno parte di quel bordo
  - ⇒ Dato una faccia, elenca tutti i vertici che fanno parte di quella faccia
- ✓ E' necessario che queste operazioni base siano effettuate efficientemente (idealmente, in tempo costante)

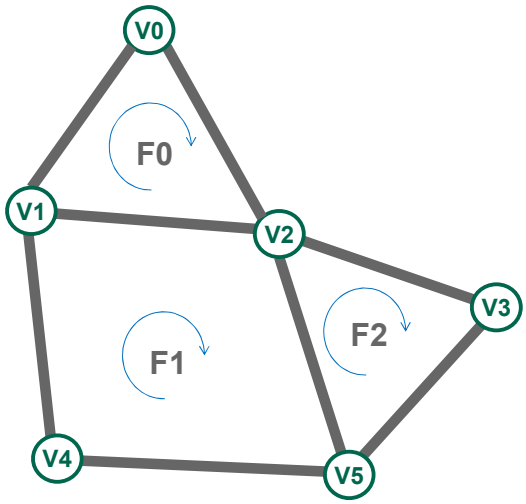
175

Strutture dati per Mesh Processing

- ✓ La struttura «lista facce» della mesh indexed, usata per memorizzare la connettività non consente di effettuare queste operazioni
  - ⇒ (se non attraverso una scansione dell'intero vettore delle facce, che richiede ovviamente un tempo lineare col numero di faccie)
  - ⇒ (eccetto: «data una faccia, elenca 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
  - ⇒ svantaggio: più prolisse, onerose da mantenere durante le modifiche
  - ⇒ ma consentono di navigare sulla mesh molto più agevolmente
- ✓ Vediamo una di queste strutture

176

Struttura dati: connettività di una indexed mesh

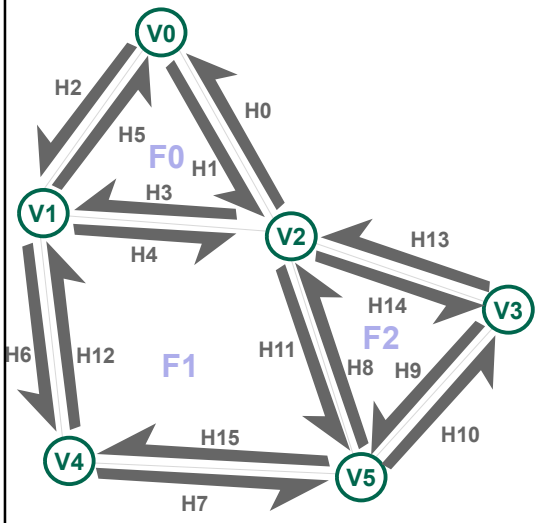


$F_0 \rightarrow$ 0 , 2 , 1
$F_1 \rightarrow$ 1 , 2 , 5 , 4
$F_2 \rightarrow$ 2 , 3 , 5

CONNETTIVITA'
(come lista facce)

177

Struttura dati: half edges



	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

182

Lista di half edge

- ✓ La connettività è rappresentata da un vettore di Half Edge
- ✓ Per ogni half edge memorizzo i campi (sono tutti indici):
  - ⇒ Indice di Vertice: da quale vertice parte
  - ⇒ 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:
  - ⇒ In ogni vertice, posso memorizzare l'indice di un Half-Edge che parte da quell vertice (uno qualsiasi)
  - ⇒ Posso avere un vettore di facce, per ogni faccia l'indice di un half edge appartenente a quella faccia (uno qualsiasi)

184

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( .... );
```

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…

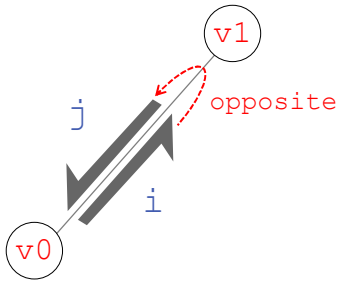
```
he[7].fi;
```

185

Esempio di uso base

✓ dato un halfedge 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;
```



186

Strutture dati per connettività a confronto

- ✓ 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)
  - ⇒ Il redering classico eseguito dalle GPU
    è pensato per questa struttura dati
  - ⇒ E' generale: non richiede ad esempio two-manifoldness o orientamento consistente delle faccie
    (quindi: capace di rappresentare strutture inconsistenti – è un vantaggio e uno svantaggio)
  - ⇒ Lista di elementi non omogenea, (alcune facce hanno un numero di vertici diverso da altre).
    eccetto che per tri-mesh o pure quad meshes: per loro, la lista facce è una matrice 3xN o 4xN (comodo)

187

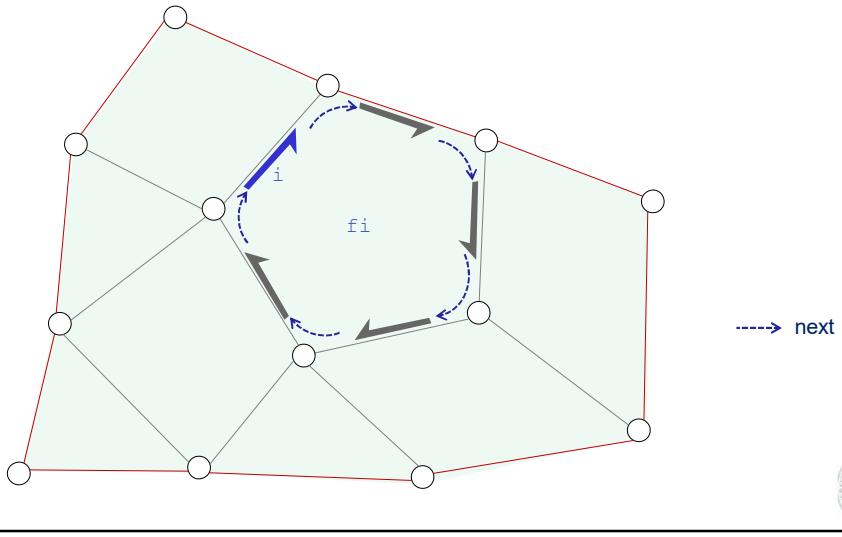
Strutture dati per connettività a confronto

- ✓ Lista di half hedge:
  - ⇒ Prolissa
    (calcolo: quanti interi è necessario memorizzare in media, rispetto ad una lista facce?
    Ipotesi: in una tri mesh, ho vertici , facce, edge tipicamente in proporzione 1x, 2x, 3x. In una quad mesh: 1x, 1x, 2x)
  - ⇒ 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'emento 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

188

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



189

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)

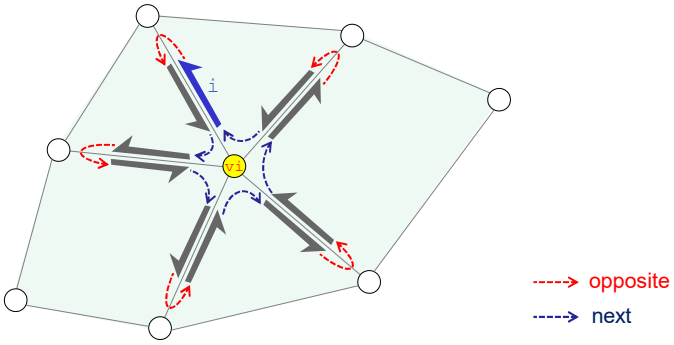
```
int fi = he[i].fi;
if (fi == -1) … /* non esiste la faccia */
int start = i; // half edge di partenza
int lati = 0;
do {
   lati ++;
   i = he[i].next;
} while (i!=start);
```

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

190

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)

191

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`

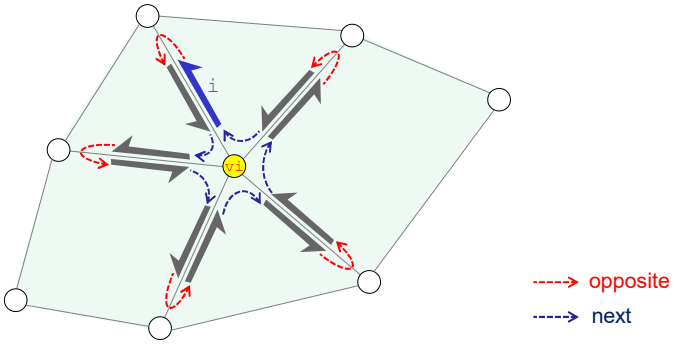
```
int vi = he[i].vi;

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)

192

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)

193

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`

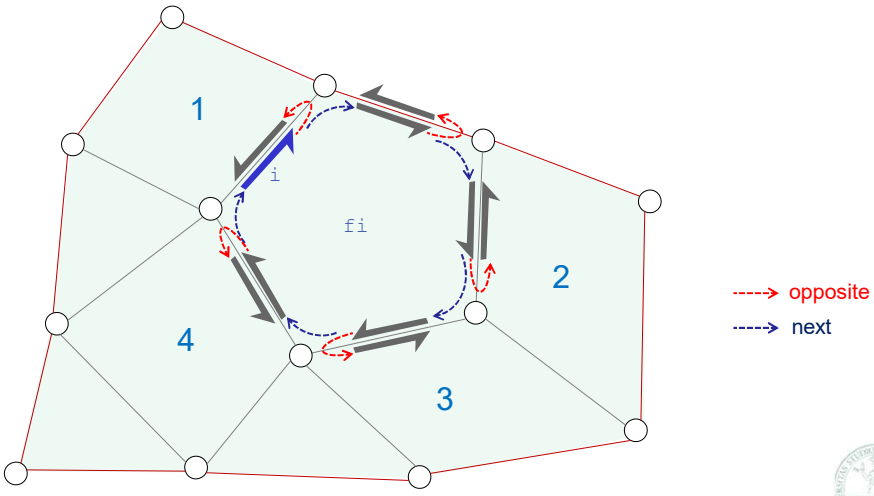
```
int vi = he[i].v;

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)

194

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



195

Esempio: contare la faccie adiacenti da una faccia

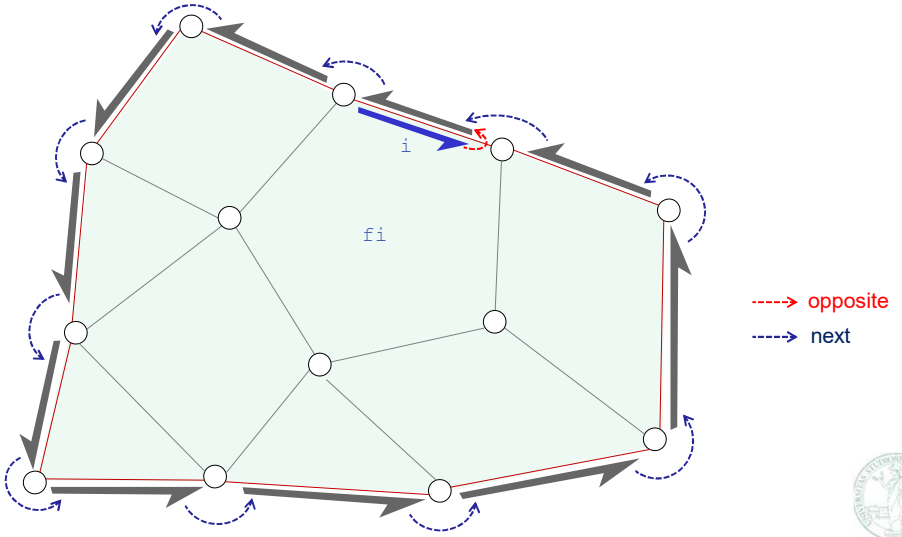
✓ dato un indice di halfedge `i`, se confina con una faccia `fi`, allora conta `quante_facce` adiacenti a `fi` ci sono

```
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);
```

196

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:



197

Esempio: visita di un bordo (insieme di edge)

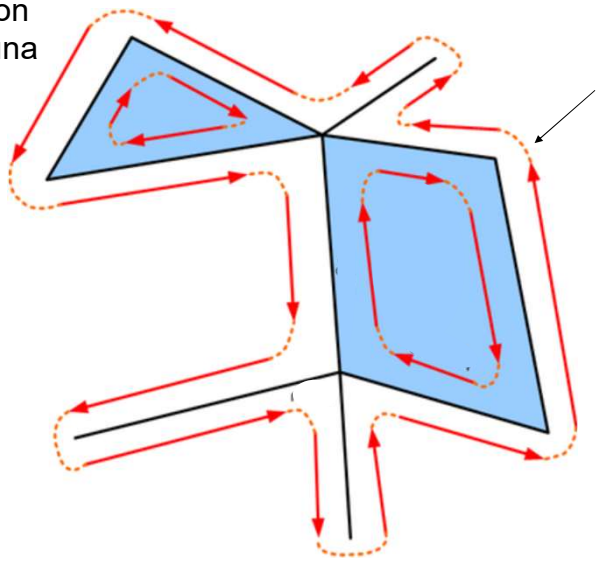
✓ Sia dato un indice di halfedge `i`, confinante con una faccia `fi`;
se l'edge corrispondente è di bordo, allora scandici tutti gli edge che fanno parte dello stesso bordo:

```
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);
}
```

198

Esempio: visita di un bordo (insieme di edge)

Funziona anche su mesh con “dangling edges” (edges non adiacenti ad alcuna faccia)



199

Strutture per connettività a confronto: sommario

INDEXED	HALF-EDGES
✓ HW friendly	✓ Rendering… complicato
✓ Navigazione… complicata ⇒ richiede strutture dati ulteriori (“di adiacenza”)	✓ Navigazione semplice ⇒ es: trovare tutti i vertici nella “1-star” di un vertice
✓ Ideale solo per mesh “pure” ⇒ tri-meshes, pure quad meshes	✓ poligoni misti a piacere
✓ Compatta: 3 indici x tri	✓ Prolissa: 12 indici per tri
✓ > adatta per rendering	✓ > adatta per geometry processing

200