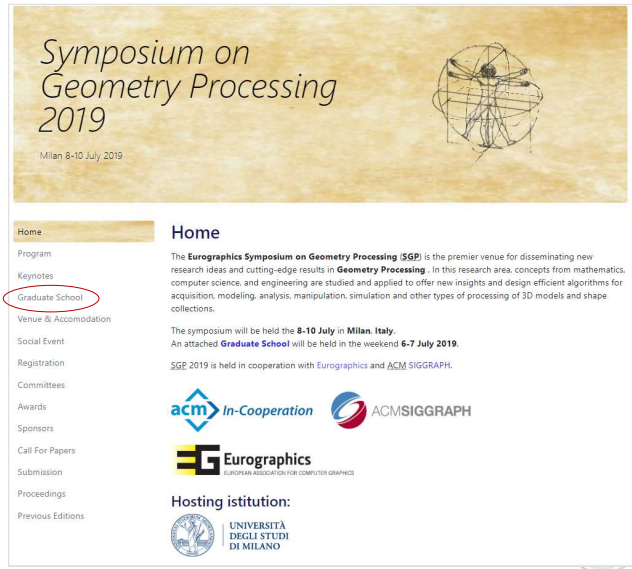
Mesh processing

Geometry processing eseguito su mesh poligonali

https://sgp2019.di.unimi.it/

Una conferenza internazionale su Geometry Processing che si è svolta nel nostro ateneo di recente:



Graduate school: lezioni introduttive ad argoemnti avanzati di geometry processing

(i video sono disponibili!)

78

Alcune lib 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 ( )

79

Mesh Processing

- ✓ Un buon applicativo di mesh processing

MeshLab

80

Task di Geometry Processing: meshing

- ✓ Meshing:
  dato una modello 3D,
  inizialmente non rappresentato come una mesh,
  costruire una sua rappresentazione mesh
- ✓ Detta anche poligonizzazione, segmentazione, tassellamento
  - ⇒ Per es,
    «meshing di una nuvola di punti», come abbiamo visto
  - ⇒ Vedremo altri esempi quando vedremo altre strutture dati

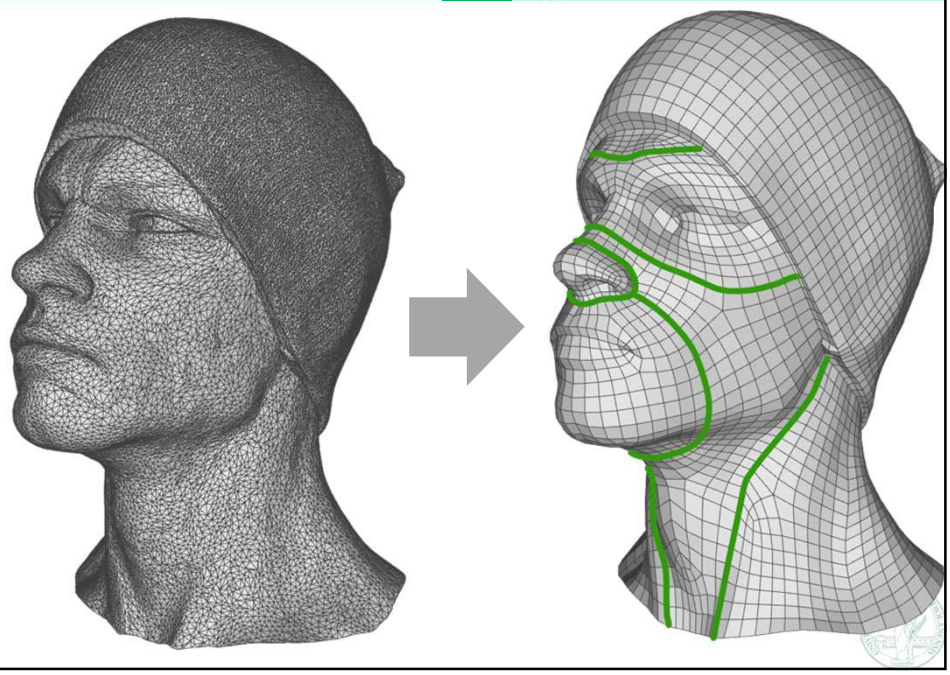
81

Geometry Processing: remeshing

- ✓ Remeshing:
  data una superficie rappresentata come una mesh, costruire una rappresentazione mesh diversa,
- ✓ In ambiente industriale, è detto “retopology”
  - ⇒specie quando fatto manualmente da un’artista
- ✓ La mesh di partenza differisce dalla mesh di arrivo in termini di, per es…
  - ⇒da tri a VS quad dominant VS quad («quad-remeshing»)
  - ⇒(semi) regular VS irregular («semiregular-remeshing»)
  - ⇒adaptive VS non adaptive resolution
  - ⇒bad element shapes VS good element shapes

82

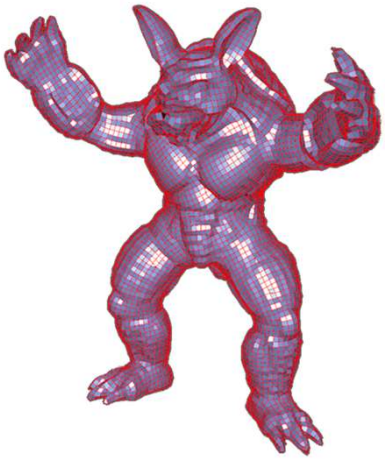
Semiregular Quad Remeshing: esempio



83

Semiregular Quad Remeshing: esempio

Input:
irregular tri mesh

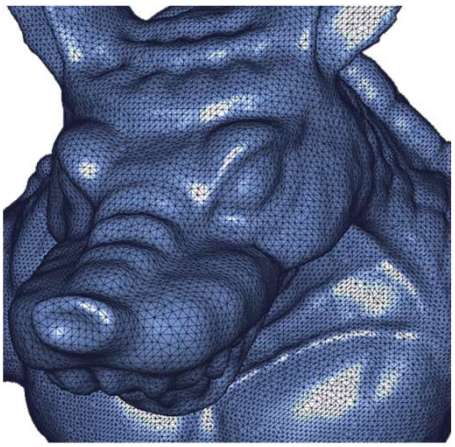


Output:
semiregular quad mesh

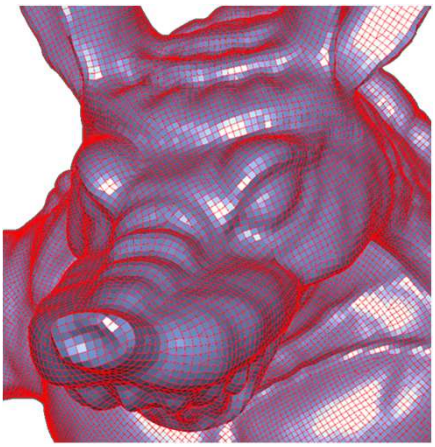
Esempio ottenuto con : «*instant field-aligned meshes*»

84

Semiregular Quad Remeshing: esempio



Input:
irregular tri mesh



Output:
semiregular quad mesh

Esempio ottenuto con : «*instant field-aligned meshes*»

85

Geometry Processing: remeshing

Spesso fatto per migliorare la qualità della mesh:

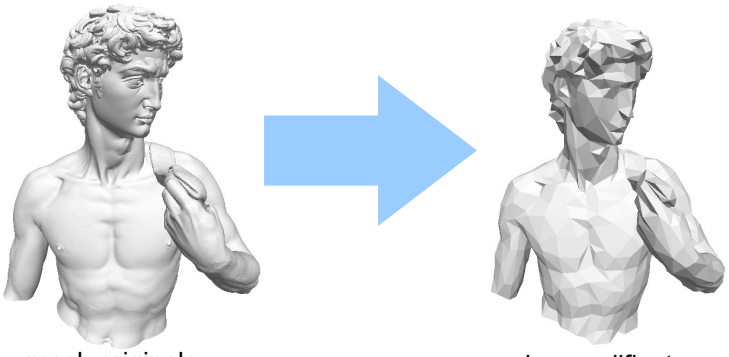
- ✓rendere la risoluzione adattiva
- ✓o, viceversa: rendere la risoluzione costante
  - ⇒dimensione facce costante
    (utile per es in una simulazione fisica)
- ✓o, passare da triangoli a quads
- ✓o, rendere la mesh regolare
  - ⇒«semiregular remeshing»
  - ⇒caso molto diffuso perchè:
    le mesh catturate dal vero sono spesso irregolari
    ma le mesh semi-regolari sono più utili:
    più facili da editare, animare, etc

86

Geometry processing: semplificazione automatica

Riduzione della sua risoluzione

Non necessariamente un «remeshing» completo:
alcune parti della mesh possono essere inalterate



mesh originale
500K triangoli

mesh semplificata
2K triangles

87

Semplificazione automatica di una mesh

- ✓ Da: mesh hi-res
  a: mesh low-res (detta anche low-poly mesh)
- ✓ Procedimento chiamato anche
  - ⇒ «Mesh coarsening»
  - ⇒ «Poly-reduction» (in ambiente industriale)
  - ⇒ «Mesh decimation»
- ✓ Motivo: la risoluzione deve essere adatta all'applicazione
  - ⇒ Molti dei procedimenti su una mesh (rendering compreso!) hanno una *complessità lineare* col numero di elementi
- ✓ Osservazione:
  - ⇒ in una nuvola di punti, bastava scegliere un sottoinsieme
  - ⇒ per mesh, è più complicato: non posso rimuovere elementi senza danneggiare le proprietà della mesh (es. chiusura)

88

Semplificazione automatica di una mesh

- ✓ Obiettivo: ottenere un buon bilancio fra:
  - ⇒ costo (risoluzione della mesh risultante, cioè numero di elementi residui)
  - ⇒ qualità (errore geometrico introdotto rispetto alla mesh originale)
- ✓ Q: come definisco l'errore introdotto?
  - ⇒ A: misuro la *distanza geometrica* fra le due superfici descritte da mesh originale e mesh semplificata
  - ⇒ cioè assumiamo:
    mesh originale = «ground truth»
- ✓ Q: come definisco la distanza fra due superfici?
  - ⇒ def matematica: la risposta esula da questo corso
  - ⇒ per approfondire: cercare «hausdorff distance»
- ✓ Q: come la posso calcolare?
  - ⇒ task del geometry processing – ma la risposta esula da questo corso
  - ⇒ per approfondire: google search for
    «Metro: measuring error on simplified surfaces»

89

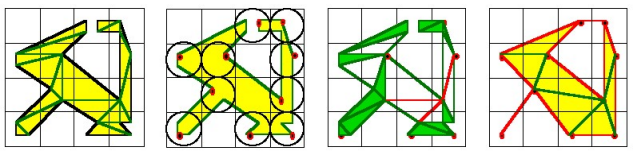
Semplificazione automatica di una mesh

- ✓ Molte algoritmi (eruistici!), seguono approcci diversi
- ✓ Le tecniche si differenziano in:
  - ⇒ Adattive e no
    - lasciano piu' triangoli dove c'e' bisogno (es non nelle zone piatte)
    - oppure riducono il numero di elementi ovunque
  - ⇒ Con garanzie su errore massimo introdotto, o no
    - viene misurato? è limitato superiormente?
    - oppure nessun limite / o nessuna misura
  - ⇒ Incrementali o discrete:
    - Ho una sequenza continua di mesh a risoluzione sempre inferiore?
    - Oppure ho solo una singola mesh finale
  - ⇒ Caratteristiche della mesh (2 manifoldness, chiusa):
    - sono sempre mantenute?
    - oppure, possono essere perse?
  - ⇒ e molto altro...
- ✓ Vediamo due esempio

90

Semplificazione automatica

- ✓ Strategie a Vertex clustering:
  - ⇒ dividi lo spazio 3D in una griglia regolare di cubetti
  - ⇒ tutti i vertici in un cubetto vengono "collassati" in un solo vertice rappresentativo del cluster (a posizione interpolata)
  - ⇒ Ogni triangolo passa da indicizzare 3 vertici a indicizzare i 3 corrispondenti rappresentanti
  - ⇒ rimuovere i triangoli che sono diventati degeneri (hanno solo 1 o 2 vertici rimasti)
  - ⇒ Errore e numero di elementi dipendono da dimensione griglia (parametron del metodo)



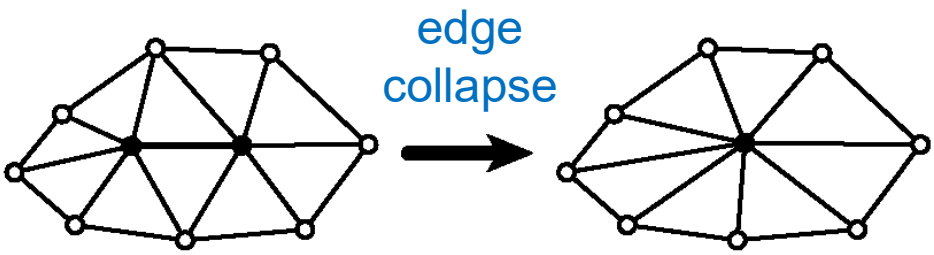
91

Semplificazione automatica di una mesh

✓Strategia a *operazioni locali* iterate

⇒Operazione locale (di coarsening):
modificare la mesh (connettività+geometria)
solo in un intorno, lasciando il resto inalterato

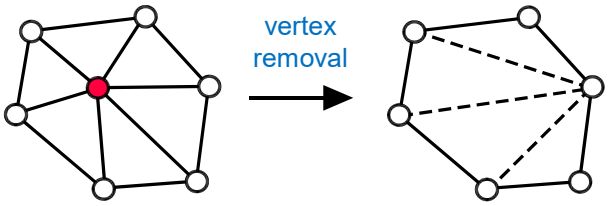
⇒Esempio più usato:



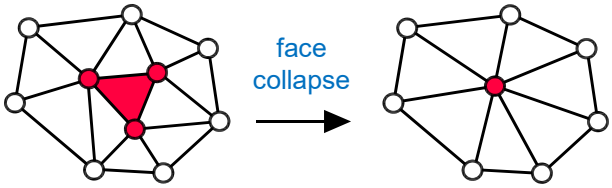
92

Semplificazione automatica di una mesh

✓Altri esempi di operazioni locali:



nota:
si può considerare
un caso particolare
di edge collapse



nota:
si può considerare
una successione
di due
edge collapse

93

Semplificazione automatica di una mesh

✓Strategia a *operazioni locali* iterate

⇒repeat

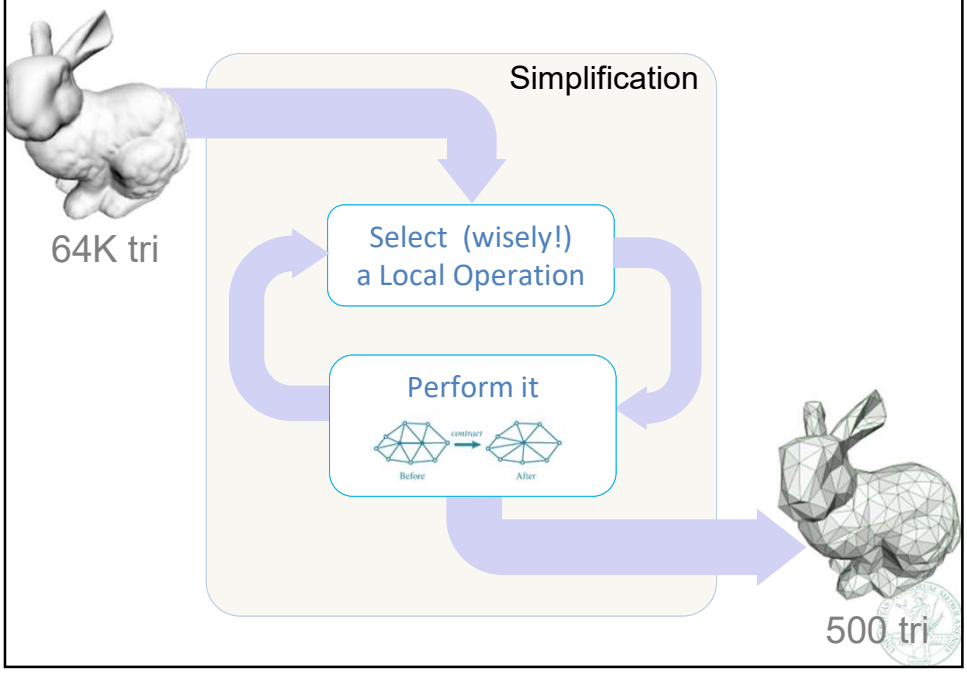
- scegli un operazione locale (secondo un criterio)
- esegui operazione locale

⇒until obiettivo raggiunto

- es: risoluzione target raggiunga
- es: errore massimo superato

94

Semplificazione automatica di una mesh

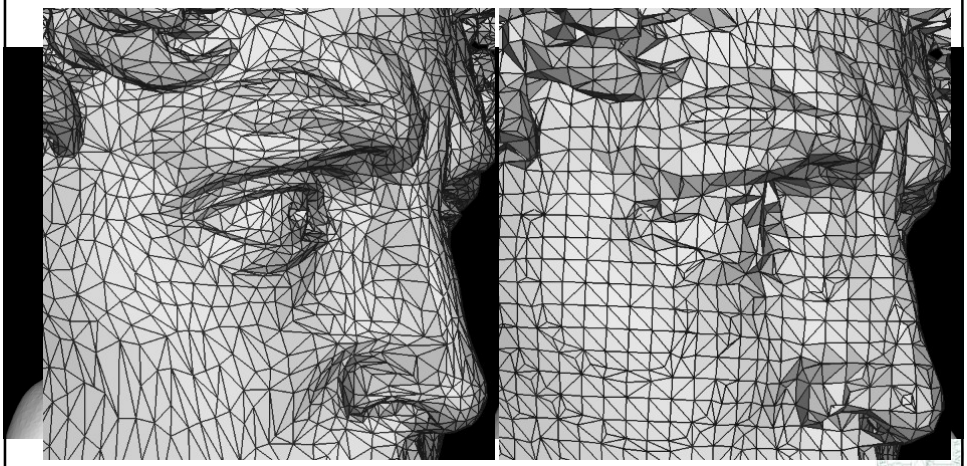


95

Due strategie a confronto

Simplificazione ad edge collapse
(adattva)
215k faces

Semplificazione a clustering
(non adattiva)
235k faces



96

Due strategie a confronto

Simplificazione ad operaz locali

- ✓ adattiva
- ✓ continua
- ✓ possible mirare a triangoli di forma buona
- ✓ possibile specificare errore massimo
- ✓ possibile specificare numero di facce target
- ✓ può mantenere caratteristche mesh
  - ⇒ two-manifoldness, chiusura, classe topologia
- ✓ spesso *richiede* mesh two-manifold in partenza

Semplificazione a clustering

- ✓ non adattiva
- ✓ discreta
- ✓ mira a facce di dimenione simile
- ✓ possibile solo specificare dimensione griglia (solo indirettamente, errore o numero di facce)
- ✓ non mantiene carattersitiche mesh
- ✓ veloce, semplice da implementare, robusta (funziona su qualsiasi mesh)

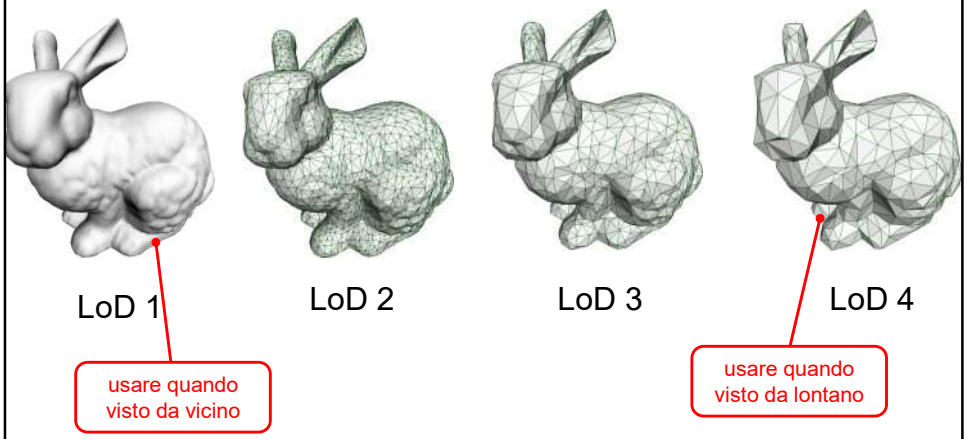
97

Semplificazione automatica di una mesh

- ✓Paragona le strategia in azione!
- ✓Apri MeshLab:
- ✓Scarica una mesh di esempio (vedi sito)
- ✓Applica:
  filtro «quadric edge collapse decimation»
  filtro «clustering decimation»

98

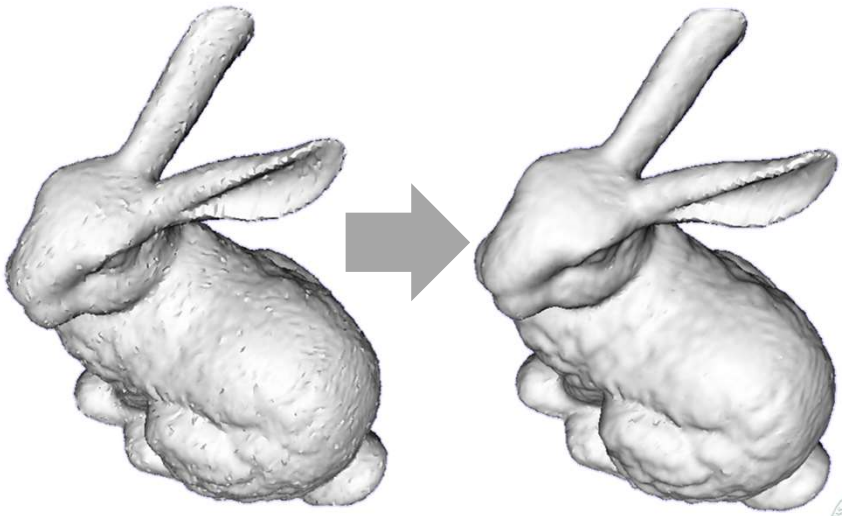
Struttura multires: “Level of Detail (LoD) Pyramid”



99

Geometry Processing tasks: examples

✓Mesh denoising

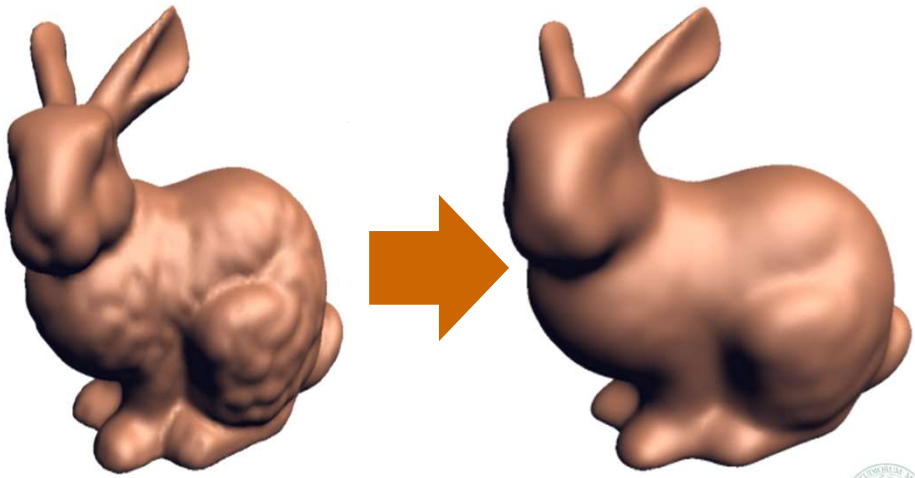


(img by Iman Sadeghi)

100

Geometry Processing tasks: examples

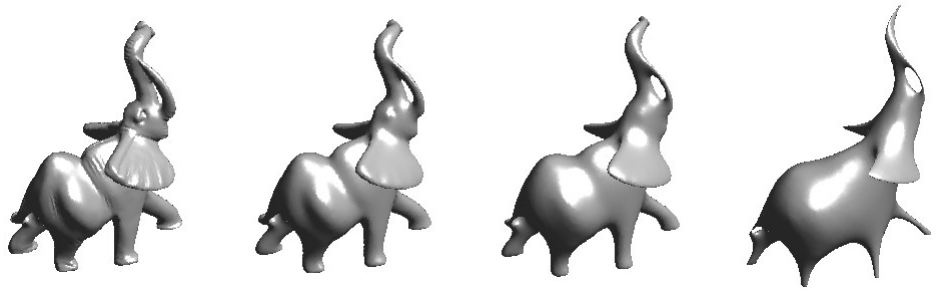
✓Mesh smoothing



101

Geometry Processing: mesh smoothing

- ✓ Mesh smoothing



102

Geometry Processing: mesh cleaning

- ✓ Mesh cleaning:
  un insieme di task per la rimozione dei difetti della connettività di una mesh, cioè
  - ⇒ le situazioni non two-manifold,
  - ⇒ buchi (per es un poligono mancante)
  - ⇒ Orientamento non consistente delle face (possibile, se mesh ben orientabile)
  - ⇒ Auto-intersezioni, etc
- ✓ Molte categorie di mesh, come le mesh scansionate (acquisizione 3D), presentano spesso molti di questi difetti
- ✓ Spesso, un preprocessing necessario a altri task di geometry processing

103

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

104

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» 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)

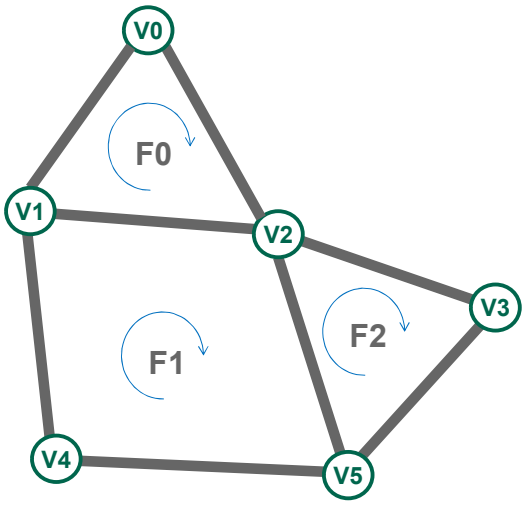
105

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

106

Struttura dati: connettività di una indexed mesh

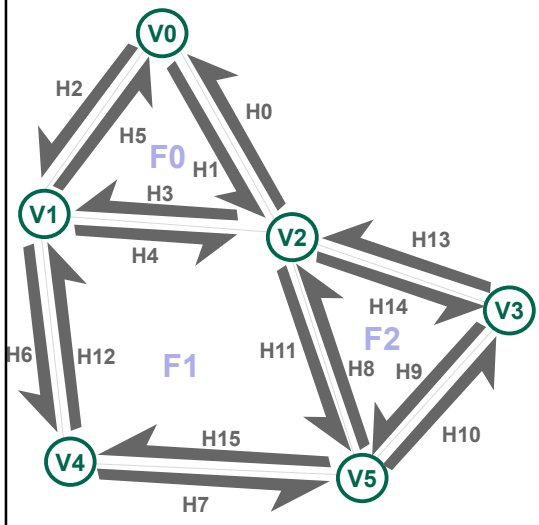


$F_0 \rightarrow$ 0 , 2 , 1
$F_1 \rightarrow$ 1 , 2 , 5 , 4
$F_2 \rightarrow$ 2 , 3 , 5

CONNETTIVITA'
(come lista facce)

107

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

111

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)

112

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;
```

113

Esempio di uso base

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

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

114

Esempio: conta quanti lati ha una faccia

- ✓ Dato un half-edge di indice `i`, corrispndente ad una data faccia, trova quanti `lati` ha questa faccia

(era un half edge di bordo)

```
int fi = he[i].opp;
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)

115

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].v;

int start = i; // half edge di partenza
int lati = 0;
do {
   i = he[i].opp;
   int vj = he[i].v;

   /* qui: fai qualcosa con vertice vj */

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

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

117

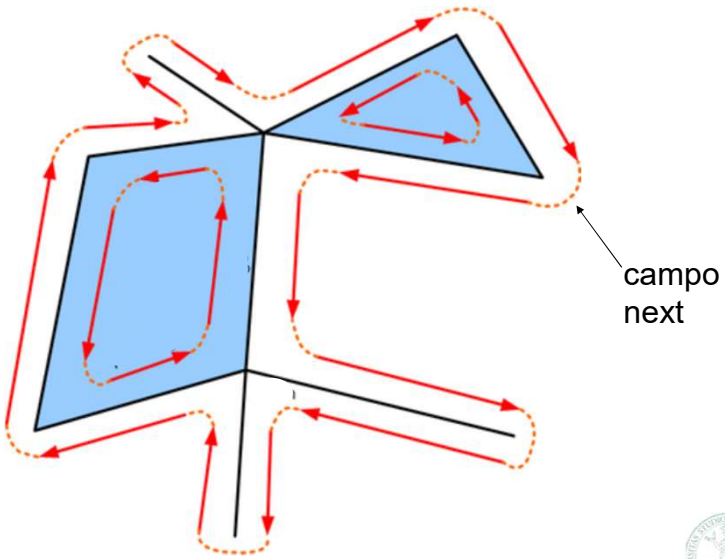
Esempio: contare la faccie adiacenti

✓ dato un indice di halfedge `i`, se confina con una faccia `fi`, allora conta il `quante_facce` di faccie 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);
```

118

Struttura di Half edges



119

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)

120

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
  - ⇒ Lista di elementi sempre omogenea: 4 elementi per half-edge
    (nella variante vista), anche su mesh poligonali miste
  - ⇒ Consente di «navigare sulla mesh», con operazioni di pasaggi al vicino 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 di una stesso oggetto (la connettività della mesh).
  - ⇒ Una si può costruire a partire dall'altra

121