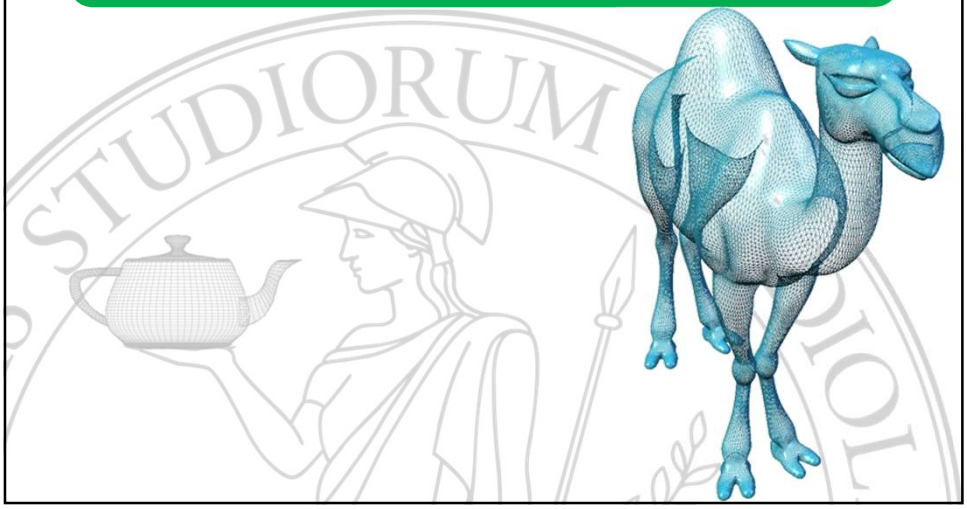
Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

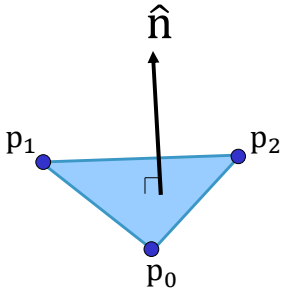
Polygonal Representations: cenni di Mesh Processing 1/2



88

Computo della normale di un triangolo

(Cioè del suo orientamento nello spazio)



due "edge vectors"

$$\vec{n} = (p_1 - p_0) \times (p_2 - p_0)$$

"area vector"
un vettore ortogonale al triangolo,
lungo quanto la doppia area del triangolo

normalizzazione

$$\hat{n} = \frac{\vec{n}}{\|\vec{n}\|}$$

"normale" del triangolo
(vett unitario)

E' la
doppia area
del triangolo

89

Calcolo delle normali per faccia di una mesh (note)

(vedi pseudocodice [1] sul sito)

- ✓ La normale delle facce triangolari è data dal semplice computo visto sopra
  - ⇒ ripetuto per ciascuna faccia
- ✓ E' anche detta normale «geometrica» della faccia
  - ⇒ perché corrisponde effettivamente all'orientamento del piano su cui giace il triangolo (ed è costante)
  - ⇒ esiste sempre (nota: questo non vale per facce quads – a meno che non siano planari)
  - ⇒ un'eccezione: triangoli «degeneri» (quelli con area 0) (con 3 vertici allineati o, 2 vertici coincidenti, etc) cosa succede se tento l'algorimto sopra?
- ✓ La normale per faccia è orientata consistentemente (nello stesso verso) solo se la mesh è ben orientata
  - ⇒ (cosa succede altrimenti?)
  - ⇒ normale verso l'interno oppure l'esterno? Dipende della formula che uso per il suo computo (quali edge-vector scelgo e in che ordine li moltiplico)

90

Calcolo delle normali per vertice di una mesh (note)

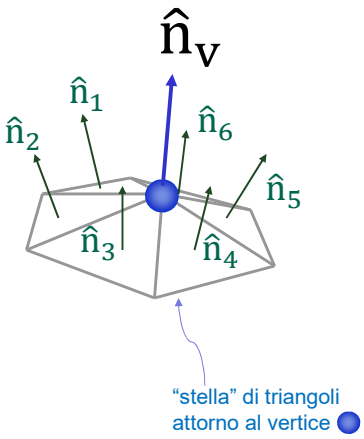
- ✓ La normale per vertice di una mesh non è definita in modo univoco per via geometrica
  - ⇒ quindi dobbiamo «inventarcene» una.
- ✓ La domanda che ci dobbiamo porre è:
  - ⇒ «immaginando che questa mesh (che è composta da facce *piatte*) modelli invece una superficie invece *curva*, quale sarebbe la normale di questa superficie curva, sul vertice?»
  - ⇒ Non esiste una risposta univoca!
  - ⇒ La risposta dipende da quale superficie si intende rappresentare.
- ✓ Strategia spesso utilizzata in pratica:
  - ⇒ usare una interpolazione delle facce adiacenti (per es, la media)
- ✓ *le normali (per vertice) fanno parte del modello* e spesso vengono costruite insieme al modello.
  - ⇒ Solo nei casi di una mesh sprovvista di normali, sarà necessario computarle dalla geometria (e dalla connettività)

91

Normali come attributo per vertice

Un modo per calcolare di un vertice condiviso da k triangoli: la media delle k normali definite sui triangoli

k è detta la valenza del vertice, (nel disegno, k = 6)



$$\hat{n}_v = \frac{\hat{n}_1 + \cdots + \hat{n}_6}{\|\hat{n}_1 + \cdots + \hat{n}_6\|}$$

Nota: dato che si normalizza il risultato della sommatoria, non c'è necessità di dividere per il numero di delle elementi sommati

92

Algoritmo per computo di normale per vertice

Passo 1: computo delle normali per faccia

- ✓ Per facce non-triangolari: si possono mediare le normali costruite su ogni wedge della faccia
  - ⇒ prodotto cross dei due edge vectors adiacienti al wedge
  - ⇒ Per trovarli, navigo gli half edge attorno alla faccia

Passo 2: computo delle normali per vertice

- ✓ ∀ vertice **v**:
  ciclo su tutte le facce adiacenti a **v**,
  sommando la loro normale in un vettore, e normalizzo questo vettore

Problema: come trovo tutte le facce adiacenti ad un vertice dato?

- ✓ Se scandisco l'intero vettore della «lista-facce»
  il mio algoritmo diviene *quadratico*
  - ⇒ se ho k vertici e 2k facce, richiede ~ 2k^2 operazioni!
  - ⇒ nel mesh processing, gli algortmi quadratici non sono feasible (k = molto grande)

Esiste un algoritmo efficiente (lineare)
che usa solo la struttura «lista-facce» per la.
Sapresti identificare questo algoritmo?

93

Algoritmo per computo di normale per vertice

(vedi pseudocodice [2] sul sito)

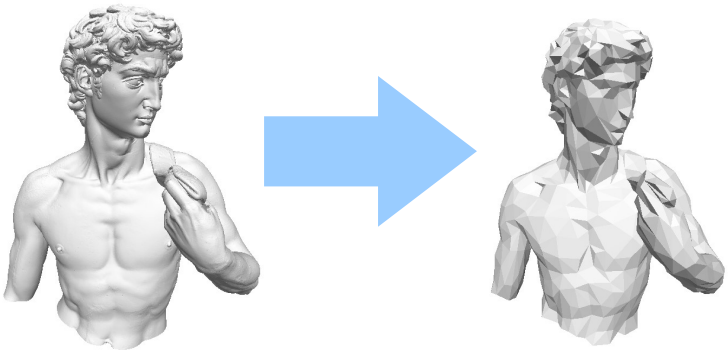
- ✓ Passo 1 ∀ vertice: azzero il suo vettore normale
  - ⇒ Questo vettore servirà per mantenere le somme parziali
- ✓ Passo 2 ∀ faccia: calcolo la sua normale,
  e la sommo alla normale di ciascun vertice ai suoi corners
  - ⇒ Alla fine di questo ciclo, su ogni vertice avrò accumulato un vettori da ciascuna faccia adiacente a quel vertice
- ✓ Passo 3 ∀ vertice: normalizzo il risultato

- ✓ La strategia generale viene a volte detta scattering:
  - ⇒ Al passo 2, invece che raccogliere (*gather*) su ogni vertice le normali dalle facce adiacenti,
    distribuisco (*scatter*) da ogni faccia le normali ai vertici adiacenti
  - ⇒ La strategia dell'algoritmo precedente viene detta *gathering*
- ✓ Anche se il risultato è lo stesso, questo algoritmo è linare:
  - ⇒ Se ho k vertici e 2k facce, mi ci vogliono solo ~ k + 2k + k ∈ O(n) operazioni
  - ⇒ Nel geometry processing, gli algoritmi lineari sono *feasible*: sono veloci anche per mesh a risoluzione molto grande (per es, k = 10^9)

94

Geometry processing: semplificazione automatica

Riduzione della sua risoluzione



mesh originale
500K triangoli

mesh semplificata
2K triangles

95

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»
- ✓ Perché è utile: la risoluzione deve essere adatta all'applicazione che usa la mesh
  - ⇒ 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)

96

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»

97

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...

98

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, che vedremo sommariamente la prossima volta

100