Algoritmo 1 per computo di normale per vertice

Passo 1: computo delle normali per faccia triangolare

⇒ come prodotto cross dei due *edge vector* della faccia, normalizzato

Passo 2: computo delle normali per vertice

✓ ∀ vertice **v**:
ciclo su tutte le facce adiacenti a **v**,
(cioè tutte le facce che annoverano l'indice di **v** fra i propri indici)
computo la somma di tutte le loro normali, e normalizzo il risultato
(per ottenere la normale di **v**)

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

✓ Se scandisco l'intero vettore della «lista-facce» in cerca di facce che contengano l'indice di **v**, il mio algoritmo diviene *quadratico*

⇒ se ho n vertici e $2n$ facce, l'algoritmo 2 richiede O($2n$^2) operazioni!

⇒ nel mesh processing, gli algoritmi quadratici non sono accettabili (perché il numero n può essere molto grande)

Esiste invece un algoritmo efficiente (lineare)
che usa solo la struttura «lista-facce».

Sapresti identificare questo algoritmo? (è nel prossimo lucido)

60

Algoritmo 2 per computo di normale per vertice

✓ Passo 1 ∀ vertice: azzero il suo vettore normale

⇒ Questo vettore servirà per cumulare le somme parziali

✓ Passo 2 ∀ faccia: calcolo la sua normale,
e la sommo alla normale di ciascun vertice ai suoi angoli

⇒ Alla fine di questo ciclo, su ogni vertice avrò accumulato un contributo da ciascuna faccia adiacente a quel vertice

✓ Passo 3 ∀ vertice: normalizzo il risultato

✓ La strategia generale dell'algoritmo 1 viene detta «*gathering*»

⇒ raccolglo (*gather*) su ogni vertice le normali delle facce adiacenti

✓ La strategia generale dell'algoritmo 2 viene detta «*scattering*»:

⇒ distribuisco (*scatter*) da ogni faccia la normale sui vertici adiacenti

✓ Anche se il risultato è lo stesso, l'algoritmo 2 è lineare:

⇒ Se ho n vertici e $2n$ facce, mi ci vogliono solo $\sim n + 2n + n \in O(n)$ operazioni

⇒ In geometry processing, gli algoritmi lineari sono *feasible*: cioè sono efficienti anche per mesh a risoluzione molto grande (per es, n = 10^9)

61

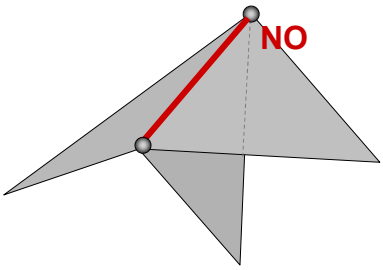
Mesh two-manifold e non

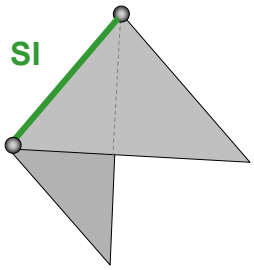
- ✓ una mesh è detta two-manifold (una "varietà due") quando *rappresenta in effetti una superficie*
  - ⇒ molti algoritmi di geometry processing necessitano che questo sia il caso!
- ✓ Non tutte le mesh (= insiemi di poligoni che condividono dei vertici e degli edge) lo sono!
  - ⇒ le facce di una mesh rappresentano sempre (pezzi di) superficie – tutto ok
  - ⇒ su edge e vertici le cose possono andare storte
  - ⇒ vediamo quali condizioni devono verificare gli edge e i vertici affinché la mesh rappresenti una superficie bidimensionale?

62

Edge two manifold

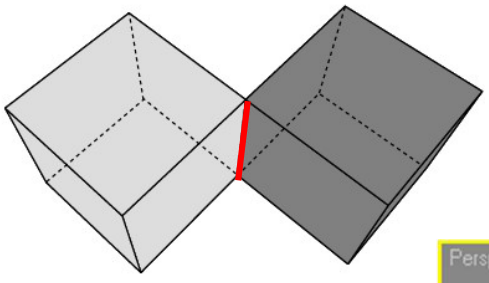
- ✓ un edge two-manifold se è condiviso da al più due facce

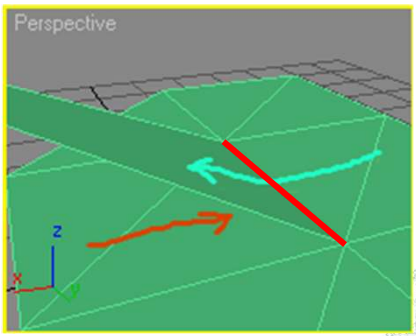




63

Esempi di edge non 2-manifolds



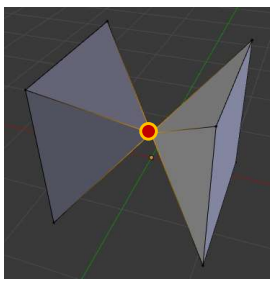


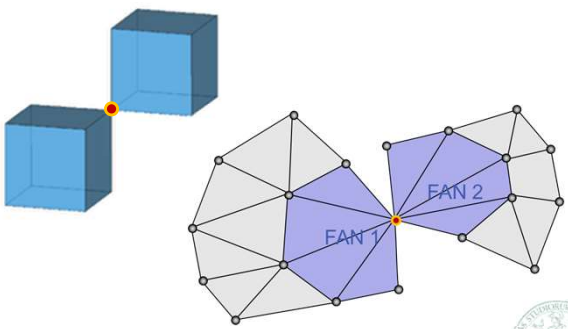
64

Vertice two-manifold

- ✓ Due facce sono dette *adiacenti* se condividono un edge
- ✓ L'insieme di facce che condividono un vertice, tutte adiacenti a coppie = un fan (letteralmente: un ventaglio)
- ✓ un vertice è two manifold se ha un solo fan di facce

Esempi di vertici non 2-Manifold:

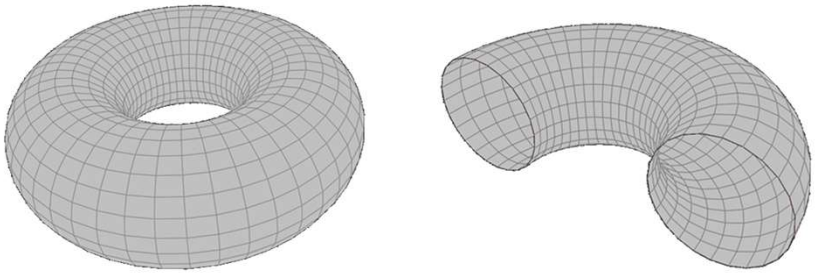




67

Mesh chiuse e aperte

- ✓ un edge condiviso da 1 faccia è «aperto», o di bordo;
- ✓ un edge condiviso da 2 faccie è «interno»;
- ✓ se una mesh non ha edge di bordo, è chiusa
- ✓ altrimenti, è aperta
  - ⇒ la distinzione ha senso solo se la mesh è two-manifold

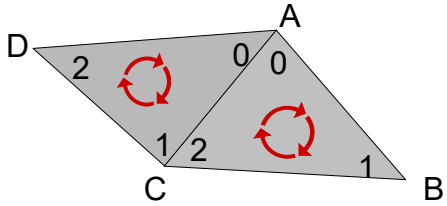


chiusa aperta

69

Orientamento delle facce

- ✓ Una mesh two manifold può essere «ben orientata» oppure no
- ✓ Ben orientata = l'ordinamento dei tre vertici dentro ogni faccia è consistente (sempre senso orario oppure sempre senzo antiorario)



t_a = {A,C,D}
t_b = {A,B,C}

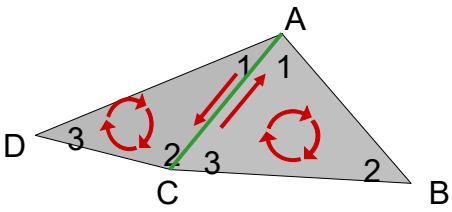
71

Mesh

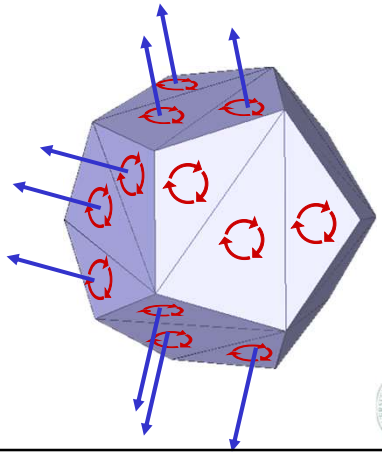
✓Orientabile, non orientabile

⇒è possibile assegnare un orientamento ad ogni faccia coerentemente?

⇒orientabile → normali coerenti



senso opposto, edge coerente



72

Osservazione

```
OFF
12 10 0
0.0 0.0 0.0
3.0 0.0 0.0
3.0 1.0 0.0
1.0 1.0 0.0
1.0 5.0 0.0
0.0 5.0 0.0
0.0 0.0 1.0
3.0 0.0 1.0
3.0 1.0 1.0
1.0 1.0 1.0
1.0 5.0 1.0
0.0 5.0 1.0
4 3 2 1 0
4 5 4 3 0
4 6 7 8 9
4 6 9 10 11
4 0 1 7 6
4 1 2 8 7
4 2 3 9 8
4 3 4 10 9
4 4 5 11 10
4 5 0 6 11
```

geometria

connettività

Le caratteristiche di essere: two-manifold, chiusa, aperta, ben orietnata, bel orientabile… dipendono esclusivamente dalla connettività della mesh (non la sua geometria). Per es:

Esercizio:
Puoi dire, guardando il testo del file ←qui accanto, se si tratti una mesh:

- quad-dominant, pure-quad, o tri?
- two-manifold o no?
- chiusa o aperta?
- ben orientata o no?
- low-poly o hi-res?

73

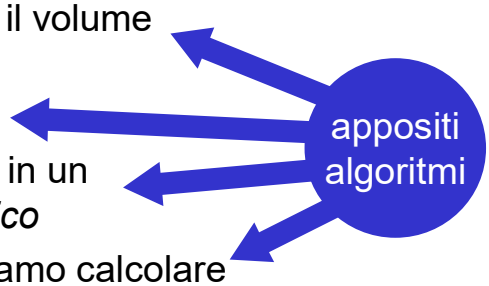
Note sull'Esercizio

- ✓ Ogni coppia di indici consecutivi in ogni faccia individua un edge
  - ⇒ compreso: dall'ultimo vertice della faccia al primo. E' un array ciclicl o!
- ✓ L'edge è two-manifold se: compare in max due facce.
- ✓ Due facce che condividono un edge sono orientate consistentemente se: l'edge appare, nei due casi, in versi opposti (es: 1 => 5 nella prima faccia, e 5 => 1 nella seconda)
- ✓ Edge di bordo (o «aperto») se: appare solo in una faccia
- ✓ Mesh two-manifold se: tutti gli edge sono two-manifold
- ✓ Mesh chiusa se: non ci sono edge aperti
- ✓ Mesh ben orientata se: tutti le coppie di facce adiacenti sono orientate consistentemente

75

Osservazione

- ✓ Una mesh two-manifold, ben-orientata e chiusa separa uno spazio *interno* da uno *esterno*
- ✓ Quindi:
  - ⇒ Rappresenta un oggetto solido!
  - ⇒ Possiamo calcolarne il volume (oltre che l'area)
  - ⇒ …e il suo baricentro
  - ⇒ Possiamo convertirla in un modello 3D *volumetrico*
  - ⇒ Dato un punto, possiamo calcolare se è esterno o interno
  - ⇒ Possiamo stamparla in 3D!



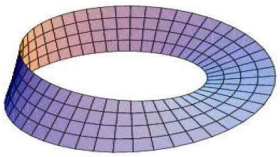
76

Circa la connettività di una mesh

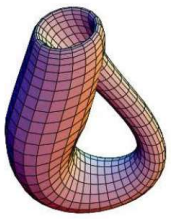
✓ Orientabile, non orientabile

⇒esempi di mesh non orientabili:

- mesh non two-manifold
- e...



Nastro di Moebius
(non orientabile, aperta)



Bottiglia di Klein
(non orientabile, chiusa)

77

Generazione di mesh poligonali

✓ Occupiamoci ora dei metodi esistenti per *generare* **mesh poligonali**

✓ Una caso comune è la generazione di **tri-mesh** a partire da **nuvole di punti**

⇒ Task detto il «meshing» di una point cloud

⇒ Un task di geometry processing, naturalmente

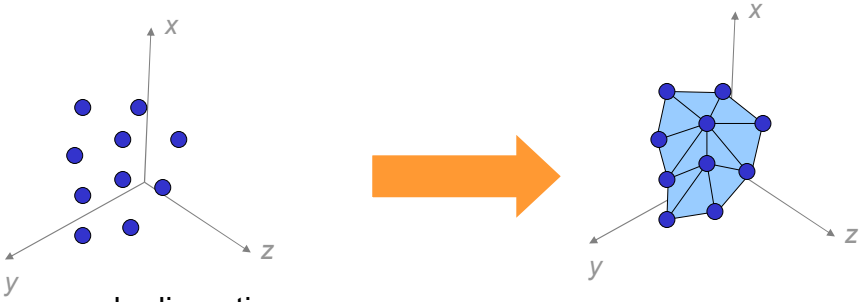
⇒ Ad esempio, la fotogrammetria genera nuvole di punti, che vanno tipicamente convertite in mesh per il loro utilizzo

✓ All'interno di questo caso, un approccio possibile consiste nel considerare i punti della nuvola come la geometria della mesh (l'insieme dei suoi vertici), e creare la connettività producendo triangoli che li connettono

⇒ Alcuni punti della nuvola possono essere ignorati e scartati, quando sono considerati «outliers»

79

Da Nuvola di punti a Triangle Mesh

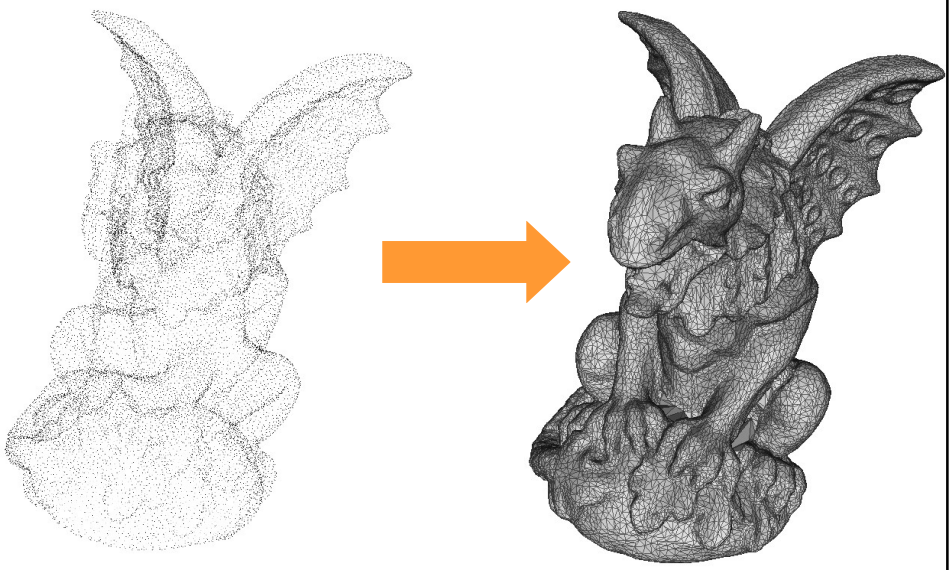


nuvola di punti

triangoli

80

Da Nuvola di punti a Triangle Mesh

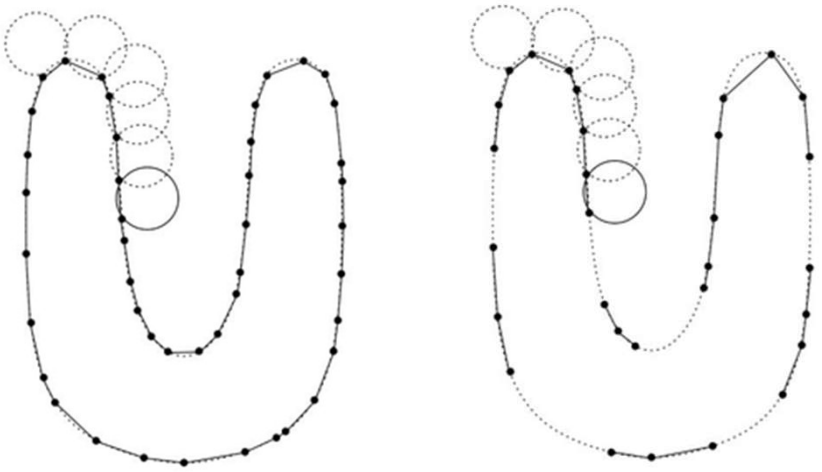


81

Da Nuvola di punti a Triangle Mesh

✓ Una classe di algoritmi: Front Advancing:

⇒Come per es «ball pivoting»



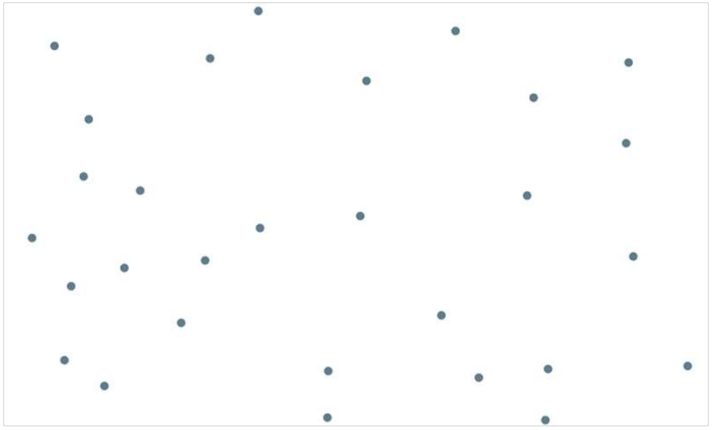
[Bernardini, Mittleman, Rushmeier, Silva TVCG 99]

82

Da Nuvola di punti a Triangle Mesh… in 2D

✓ In 2D: il problema ha un'ottima soluzione:
la triangolazione di Delaunay

⇒Ben nota dagli anni '30 (in geom. computazionale)



Vertici (sul piano x,y)

83

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ In 2D: il problema ha un'ottima soluzione: la triangolazione di Delaunay
  - ⇒ Ben nota dagli anni '30 (in geom. computazionale)

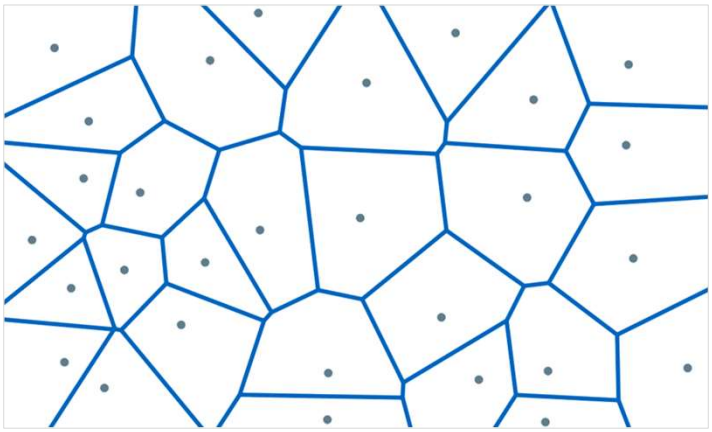
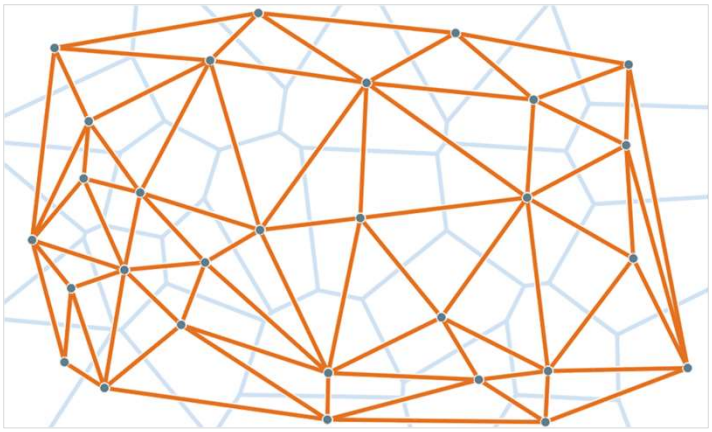


Diagramma di Voronoi

84

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ In 2D: il problema ha un'ottima soluzione: la triangolazione di Delaunay
  - ⇒ Ben nota dagli anni '30 (in geom. computazionale)

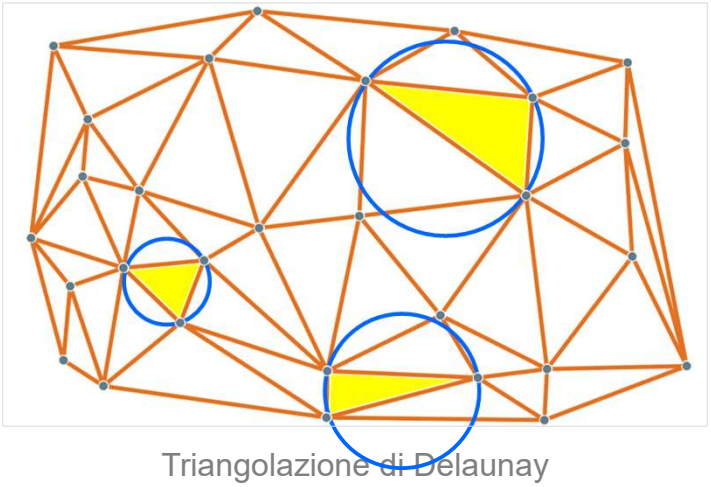


Triangolazione di Delaunay

85

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ E', una «buona» triangolazione
  - ⇒ Una buona proprietà è garantita:



Triangolazione di Delaunay

86

Triangolazione di Delaunay: note

- ✓ Diagramma di Voronoi (in 2D)
  - ⇒ Una partizione in "regioni" indotta da n punti (detti "seed") $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \ldots$
  - ⇒ Ogni seed produce una regione
  - ⇒ Regione di un seed $\mathbf{s}_i$ = insieme di punti $\mathbf{p}$ del piano che sono più vicini a $\mathbf{s}_i$ che ad ogni altro seed
- ✓ Triangolazione di Delaunay
  - ⇒ Il "duale" di un diagramma di Voronoi cioè:
  - ⇒ La connettività ottenuta connettendo con un edge ogni coppia di seed di regioni *confinanti fra loro*
  - ⇒ E' una triangolazione con ottime proprietà, come ad esempio: *il circocentro che inscrive ogni triangolo non include nessun altro vertice*

88