

1

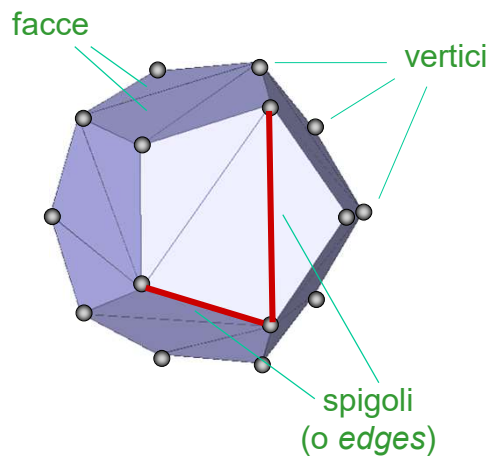
Una (imperfetta) categorizzazione dei tipi di modelli digitali 3D

		ELEMENTI DISCRETI			CONTINUI
		regolari «a griglia»	semi-regolari o irregolari		
			elementi simpliciali	elementi non simpliciali	
SUPERFICIALI	2-manifold «rappresenta una vera superficie»	Height Field Range Scan (Geometry Images)	Triangle Mesh	Polygonal Mesh Quad-Mesh Quad dominant Mesh	Subdivision surface Parametric Surface (es. B-splines)
	non-manifold «non rappresenta una sup»	Set di Range Scan	Point Cloud		
VOLUMETRICI	(3-manifold)	Voxels Solid Textures	Tetra Mesh	Hexa Mesh	Implicit model (es. CSG)

3

Mesh triangolare (o mesh simpliciale)

- ✓ Un insieme di poligoni adiacenti

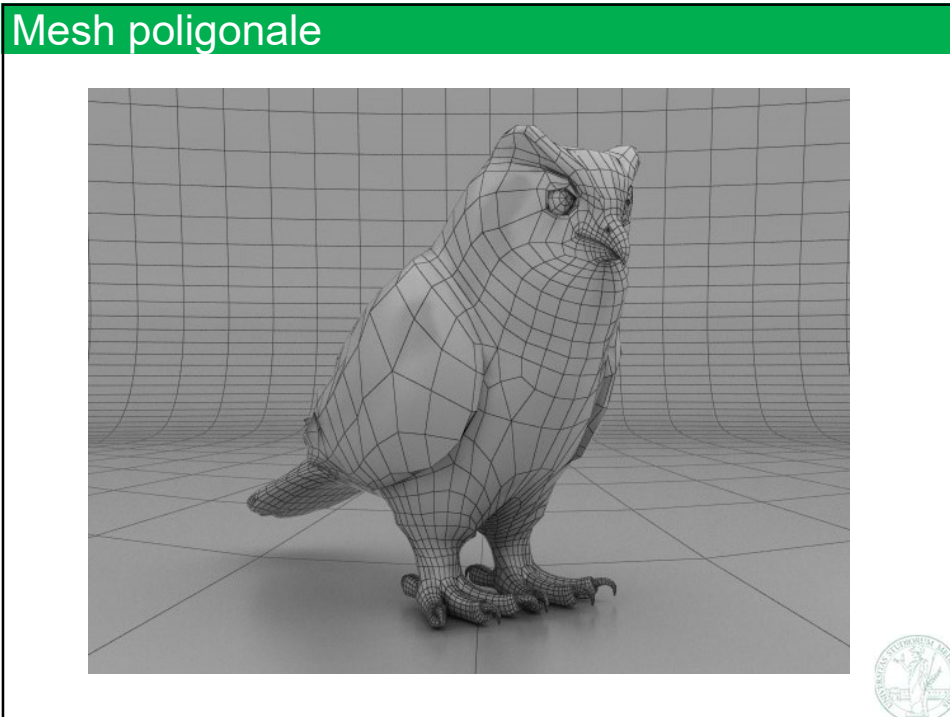


4

Polygonal Mesh (mesh poligonale)

- ✓ Superficie approssimata da “facce” poligonali
 - ⇒ adiacenti («incollate») lato a lato
- ✓ Rappresentazione “lineare a tratti” delle superfici
- ✓ Il modello 3D digitale più diffuso!
 - ⇒ spesso (per es, nei games) è un *sinonimo* di modello 3D
 - ⇒ E' GPU friendly: le schede video sono pensate per renderizzare le mesh (attraverso uno specifico algoritmo, che vedremo nella 2° metà del corso: rasterization-based)

5




7

Mesh di poligonale: struttura dati

✓ Componenti:

- ⇒ **geometria**
 - i vertici, ciascuno con pos (x,y,z)
 - un campionamento della superficie!
- ⇒ **connettività** (detta anche: "topologia")
 - come sono connessi i vertici
 - ogni poligono connette alcuni vertici
- ⇒ **attributi**
 - Definiti sulla superficie
 - es: colore, normali, UV, (indice di) materiali, ...



9

Mesh: geometria

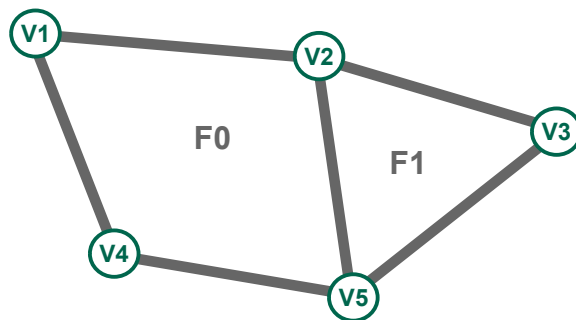
- ✓ Insieme di posizioni dei vertici
 - ⇒ Un vettore posizione (x,y,z) per ogni vertice



10

Mesh: connettività (o topologia)

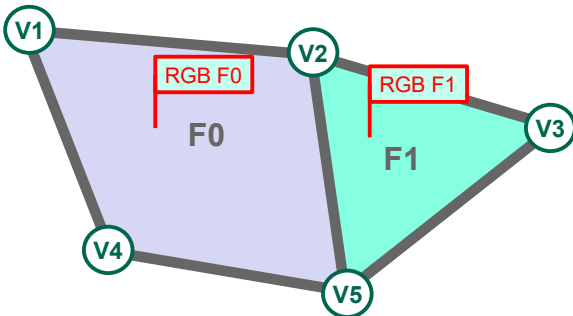
- ✓ Facce
 - ⇒ poligoni che connettono fra loro i vertici
 - ⇒ simile a: nodi connessi da archi, in un grafo



11

Mesh: attributi

- ✓ Quantità che variano sulla superficie
 - ⇒ es: colore RGB
 - ⇒ definiti per faccia:
rappresentano valori costanti su ogni faccia

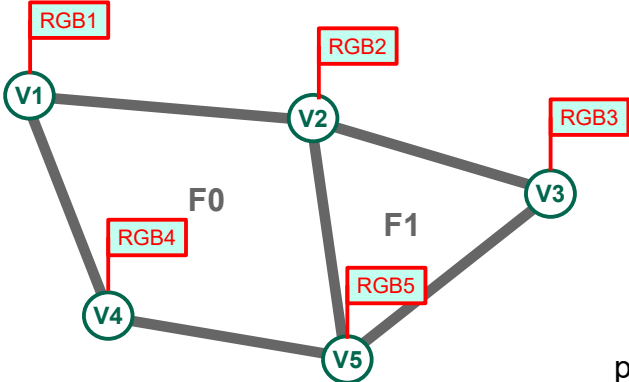


per faccia

12

Mesh: attributi

- ✓ Quantità che variano sulla superficie
 - ⇒ Definiti per vertice, e variabili sulle faccie
(vengono interpolati all'interno: vedi lez mat02)



per vertice

13

Mesh poligonali

I poligoni possono essere:

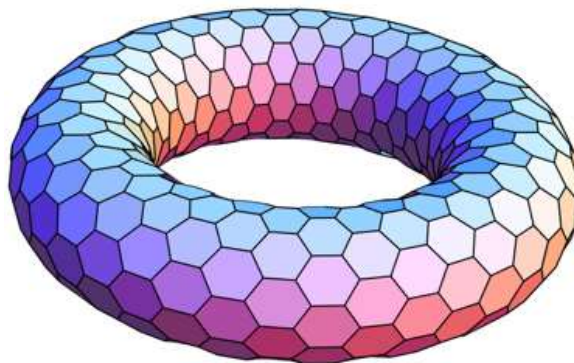
- ✓ triangoli:
 - ⇒ Triangular mesh, o tri-mesh, o “mesh simpliciale”
- ✓ quadrilateri (nello slang della CG: “quads”)
 - ⇒ Quad-qesh (a volte pure-quad mesh)
- ✓ *quasi* tutti quadrilateri
(ma alcuni triangoli, pentagoni, etc)
 - ⇒ ho una “quad-dominant” mesh
- ✓ poligoni generici
(triangoli, quadrilateri, pentagoni, esagoni, etc)
 - ⇒ mesh poligonale (generica)



14

Mesh poligonali

- ✓ volendo, esistono anche le hexa-mesh
(e hexa-dominant mesh) (ma assai poco usate)



15

Mesh Polionali: risoluzione

✓ Risoluzione:

il numero di facce (o di vertici) che compongono la mesh

⇒ Hi-res: più accuratezza

⇒ Low-res: più efficienza

⇒ Mesh low res:
detta anche low-poly mesh

⇒ La risoluzione di una mesh può essere **adattiva**:
tassellamento più fine (campionamento più fitto)
dove necessario

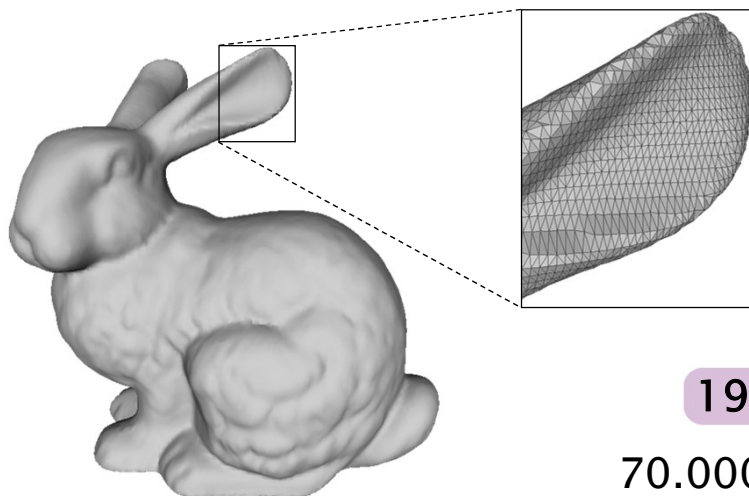
- Per es, dove la curvatura della mesh è alta
Dove la mesh è piatta, bastano meno triangoli
- Per paragone: la risoluzione di una immagine rasterizzata non è adattiva (rate costante di num pixel per unità di superficie)

num facce lineare con num vertici.
Statisticamente:
per tri-mesh:
num facce $\cong 2 \times$ num vertici
per quad-mesh:
num facce \cong num vertici



16

Risoluzione tipica: crescente negli anni

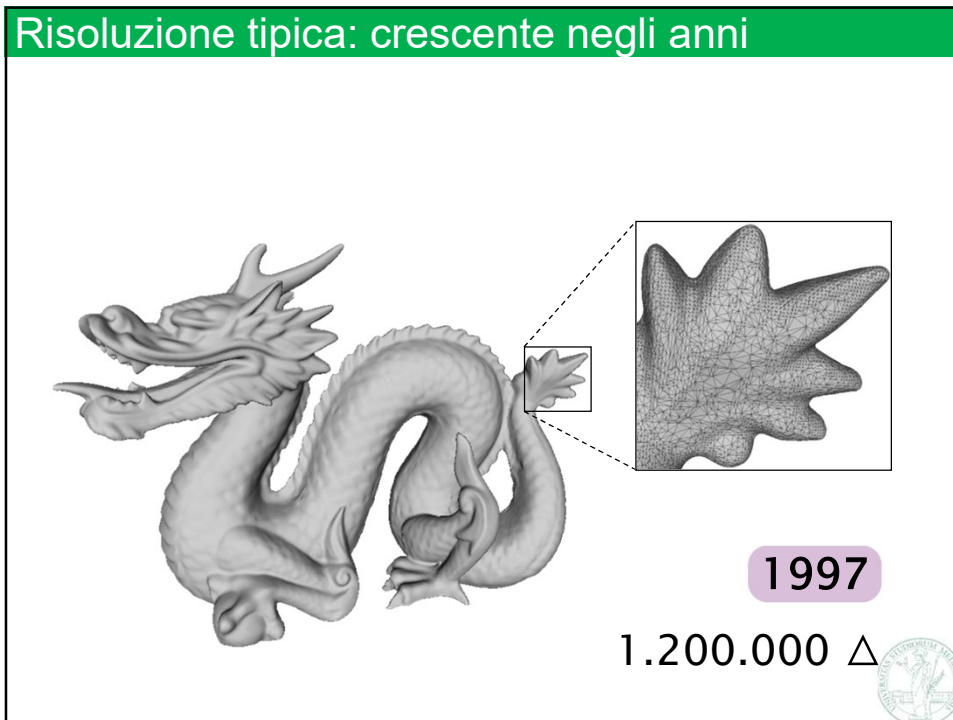


1994

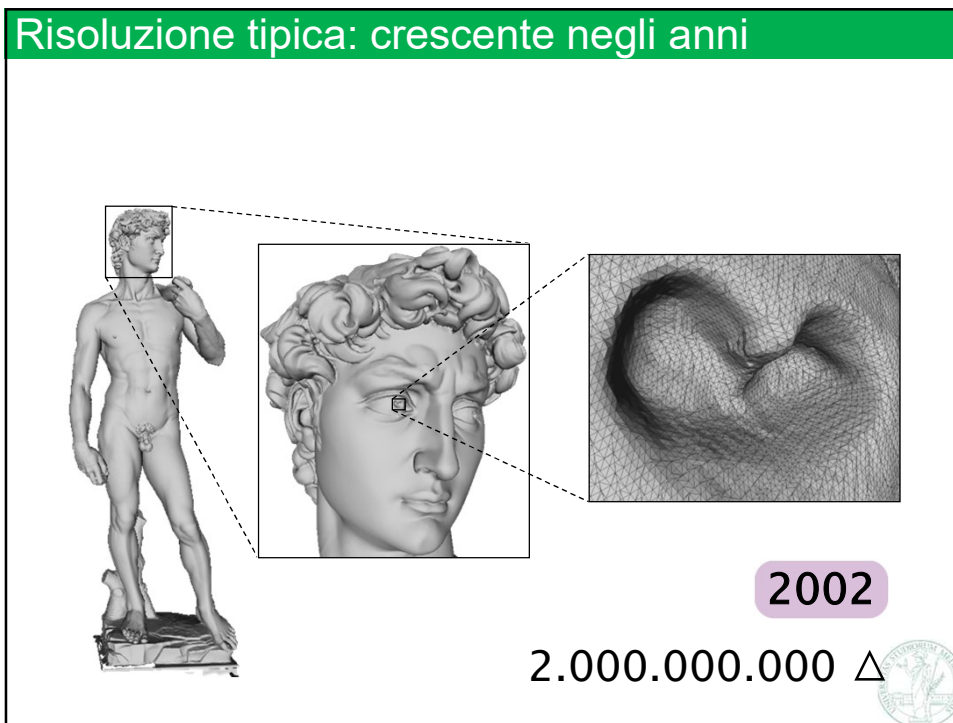
70.000 Δ



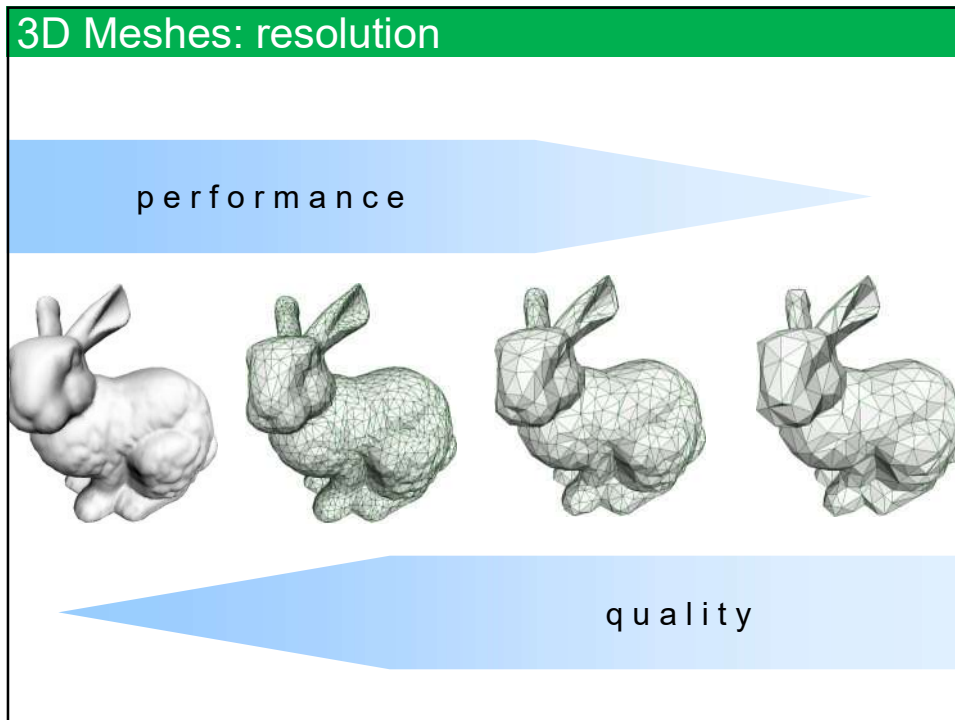
17



18




19



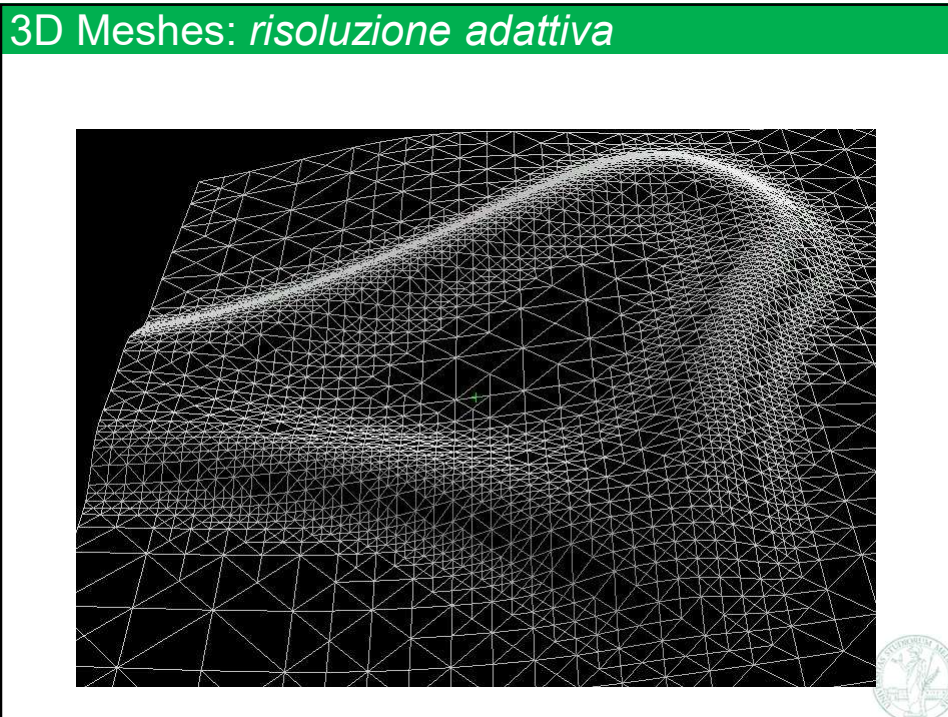
20

Mesh Polionali: risoluzione

- ✓ La risoluzione di una mesh può essere adattiva: tassellamento più fine (campionamento più fitto) dove necessario
 - ⇒ Per es:
 - dove la curvatura della mesh è alta: più triangoli;
 - dove invece la mesh è piatta, meno triangoli
 - ⇒ Per es:
 - dove la mesh è semanticamente importante (es sul volto di un personaggio): più triangoli



21



22

Mesh Triangolare o Tri-meshes

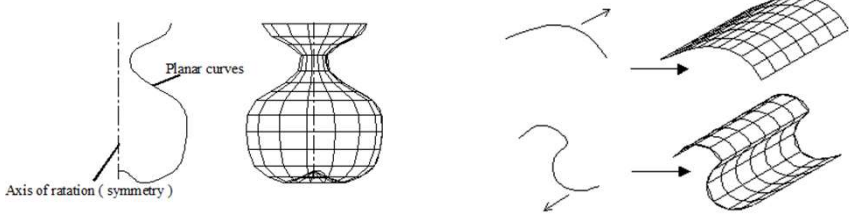
- ✓ Vantaggio: le facce sono sempre planari
 - ⇒ Perché tre punti nello spazio sono sempre co-planari
 - ⇒ matematicamente: la mesh è una approssimazione “lineare a tratti” della superficie che stiamo rappresentando
 - ⇒ Come una linea curva può essere approssimata da *segmenti dritti*, una superficie curva può essere approssimata da *triangoli piatti*
- ✓ Per questo, detta anche mesh simpliciale
- ✓ Di gran lunga il tipo di mesh più diffuso nel rendering
 - ⇒ Vantaggio: GPU hardware support (come vedremo)
 - ⇒ Cioè, questo tipo di mesh può essere renderizzato dalla GPU
 - ⇒ altri poligoni vengono scomposti in triangoli!



23

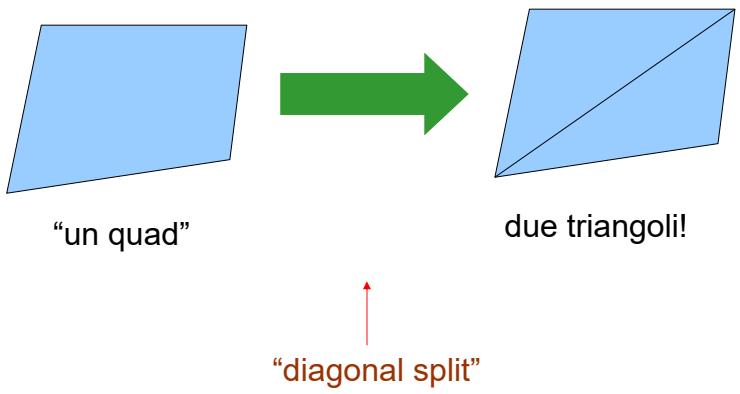
Pure quad-mesh

- ✓ Generalmente, più difficili da generare automaticamente
 - ⇒ Ma molto usate da artisti e modellatori (ad esempio, nel CAD)
 - ⇒ Per esempio, superfici «di spazzata»



24

Da Quad meshes a Triangle meshes




“un quad”

“diagonal split”

due triangoli!

26


Da Polygonal meshes a triangle meshes



un poligono a n lati?

$(n-2)$ triangoli!

↑
triangolarizzazione di poligono:
(in 3D, un problema non del tutto banale...)




27

Come rappresento internamente una mesh?

Modo diretto: (anche detto: zuppa di triangoli)

- ✓ un lungo vettore di poligoni
 - ⇒ ogni poligono (di n lati)
è rappresentato un vettore di n vertici:
 - posizione (x,y,z)
 - attributi (colore r,g,b , e/o altro)
- ✓ Grosso difetto: replicazione dei vertici!
 - ⇒ poco efficiente in spazio
 - ⇒ complicato fare updates
 - (bisogna mantenere le copie uguali durante il processing)
- ✓ Metodo non molto usato



29

Come rappresento internamente una mesh?

✓ **Modo indexed mesh:**

- ⇒ **Geometria:** array di N vertici $V[0] \dots V[N-1]$
 - in ogni vertice: posizione e attributi
- ⇒ **Connettività:** (o: "topologia")
 - Array di poligoni
 - Ogni poligono rappresentato da una sequenza di indici di vertice, uno per «angolo di faccia» (o «corner», o «wedge»)

ordinata e ciclica

es.: una faccia pentagonale: (5 corners)

6	3	0	1	4
---	---	---	---	---

indice del vertice $V[3]$

c'è un edge da $V[1]$ a $V[4]$
 nota: anche da $V[4]$ a $V[6]$

30

Struttura dati: indexed mesh

$V_0 \rightarrow$	x_0, y_0, z_0	r_0, g_0, b_0
$V_1 \rightarrow$	x_1, y_1, z_1	r_1, g_1, b_1
$V_2 \rightarrow$	x_2, y_2, z_2	r_2, g_2, b_2
$V_3 \rightarrow$	x_3, y_3, z_3	r_3, g_3, b_3
$V_4 \rightarrow$	x_4, y_4, z_4	r_4, g_4, b_4
$V_5 \rightarrow$	x_5, y_5, z_5	r_5, g_5, b_5

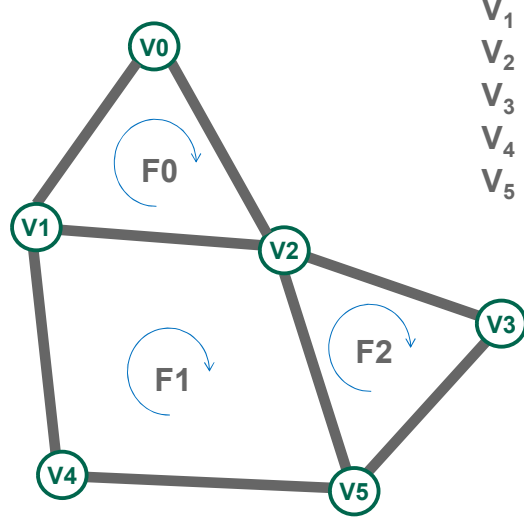
GEOMETRIA + ATTRIBUTI

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

CONNETTIVITA' (lista facce)

31

Struttura dati: indexed mesh (piccola variante)



$V_0 \rightarrow$	x_0, y_0, z_0	r_0, g_0, b_0
$V_1 \rightarrow$	x_1, y_1, z_1	r_1, g_1, b_1
$V_2 \rightarrow$	x_2, y_2, z_2	r_2, g_2, b_2
$V_3 \rightarrow$	x_3, y_3, z_3	r_3, g_3, b_3
$V_4 \rightarrow$	x_4, y_4, z_4	r_4, g_4, b_4
$V_5 \rightarrow$	x_5, y_5, z_5	r_5, g_5, b_5

GEOMETRIA ATTRIBUTI
(due vettori paralleli)

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

CONNETTIVITA'
(lista facce)

32

File formats per indexed mesh:

Per esempio: OFF ("Object File Format")

facce

edge

vertici

x,y,z
2ndo
vert

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

← indice 0

← indice 1

← indice 2

← indice 3

prima faccia:
4 vertici:
con indici
3, 2, 1 e 0

33

File formats per indexed mesh

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

header

geometria

connettività

Mesh in formato OFF
 ("Object File Format")

Quale oggetto rappresenta?

34

Mesh: strutture dati

✓ indexed mesh di triangoli

vert	X	Y	Z	R	G	B
V1	x1	y1	z1	r1	g1	b1
V2	x2	y2	z2	r2	g2	b2
V3	x3	y3	z3	r3	g3	b3
V4	x4	y4	z4	r4	g4	b4
V5	x5	y5	z5	r5	g5	b5

GEOMETRIA + ATTRIBUTI

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

CONNETTIVITA'

35

Esempio: una classe (in C++) per un oggetto Mesh

```
class Vertex {
    vec3 pos;
    rgb color; /* attribute 1 */
    vec3 normal; /* attribute 2 */
};

class Face{
    vector<int> vertexIndex;
};

class Mesh{
    vector<Vertex> vert; /* geom + attr */
    vector<Face> face; /* connettivita' */
};
```



36

Indexed mesh: as a class (here: in C++)

```
class Vertex {
    vec3 pos;
    rgb color; /* attribute 1 */
};

class Face{
    int vertexIndex[4];
    vec3 normal; /* attribute 2 */
};

class Mesh{
    vector<Vertex> vert; /* geom + attr */
    vector<Face> face; /* connettivita' */
};
```

Esempio di una variante:
le normali sono attributi memorizzati per faccia,
e la mesh è pure-quad (4 indici di vertice per faccia)



37

Mesh: attributi

- ✓ Modellano quantità che variano sulla superficie
- ✓ Esempi:
 - ⇒ Colore («diffusivo»)
 - tipicamente espresso come RGB
 - Parte della descrizione del materiale
 - ⇒ Vettore «Normale»
 - quale orientamento ha la superficie in quel punto?
 - ⇒ Le cose più varie, dipendenti dall'applicazione:
 - per es: temperatura e pressione alla superficie (in una simulazione fisica)
 - per es: «coefficiente di vulnerabilità» (in un gioco)
 - per es: qualità della ricostruzione, in un modello di un oggetto reale («quanto è accurata la superficie qui?»)



38

Mesh: attributi

- ✓ Possono essere di qualsiasi tipo:
 - ⇒ scalari (es: temperatura, qualità, etc),
 - ⇒ vettori (es: colore, normale, etc)
- ✓ E possono essere memorizzati...
 - ⇒ ...per ogni vertice
 - ed estesi dentro le faccie
 - è questo caso più comune
 - ⇒ ...oppure, per ogni faccia
 - vengono considerati costanti su quella faccia
 - quindi discontinuità C0 fra le facce in corrispondenza, cioè, degli edge



39

Mesh: attributi per vertice (mesh simpliciale)

- ✓ L'interpolazione degli attributi definiti sui vertici è definita dentro a facce *triangolari*
 - ⇒ Vedremo come, nella track matematica del corso
- ✓ Quindi, per prima cosa, gli altri poligoni devono essere suddivisi in triangoli

per vertice

40

Interpolazione degli attributi per vertice dentro ai triangoli

- ✓ Se gli attributi sono definiti **per vertice** (il caso più comune) allora sono implicitamente interpolati dentro le facce
- ✓ Ad ogni punto q in un triangolo T viene implicitamente attribuita un'interpolazione di dei tre attributi definiti sui vertici di T , usando, come peso dell'interpolazione, le **coordinate baricentriche** di q in T
 - ⇒ Gli attributi, definiti esplicitamente solo sui vertici, sono così estesi implicitamente su tutta la superficie
 - ⇒ Nota: questo è definito solo su facce triangolari: facce poligonali devono essere scomposte in triangoli
- ✓ GPU support:
 - ⇒ La GPU è specializzata nei task quali il computo delle coordinate baricentriche di punti dentro ai triangoli, e la conseguente l'interpolazione degli attributi definiti sui vertici
 - ⇒ Durante il rendering, questo procedimento viene effettuato per ogni pixel coperto da un triangolo
 - ⇒ (nota: i pixel corrispondono a punti interni delle facce)

41

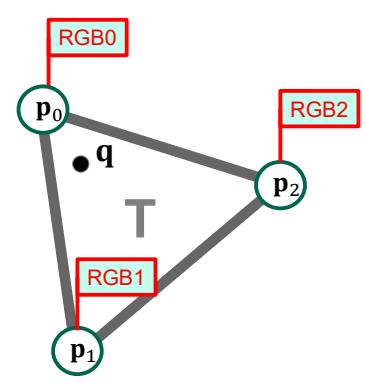
Interpolazione degli attributi per vertice dentro ai triangoli


Esempio:

- ✓ Un triangolo T ha questi tre attributi colore assegnati ai suoi vertici p_0, p_1, p_2
- ✓ Prendiamo un punto q su T di coordinate baricentriche k_0, k_1, k_2 in T , cioè con

$$q = k_0 p_0 + k_1 p_1 + k_2 p_2$$
- ✓ Allora a q assegno il colore

$$k_0 \text{RGB0} + k_1 \text{RGB1} + k_2 \text{RGB2}$$

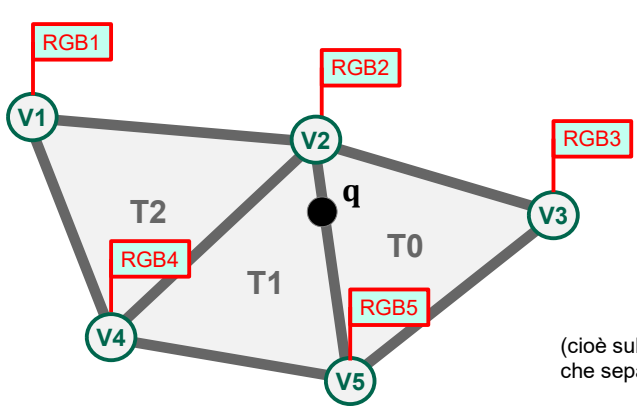


 per vertice


42

Gli attributi per vertice sono continui sulla mesh

- ✓ Gli attributi definiti per vertice (interpolati dentro le facce) sono per costruzione continui (C^∞) dentro alle facce
- ✓ Anche fra coppie di facce adiacenti, abbiamo una continuità di attributo (C^0)



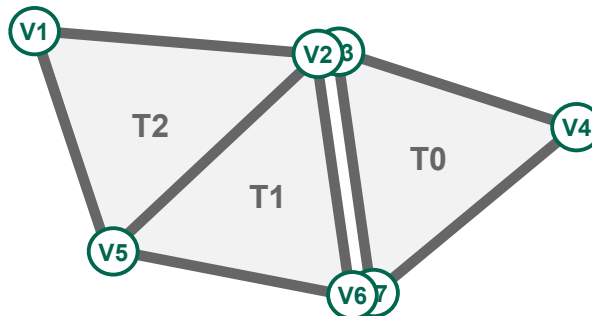
- ✓ Infatti: si consideri un punto q a cavallo fra T_0 e T_1 :
- ✓ quale attributo è associato a q , se lo si considera parte di T_0 ?
- ✓ quale attributo è associato a q , se lo si considera parte di T_1 ? (lo stesso!)



43

Attributi per vertice con discontinuità

- ✓ Se è necessario, è possibile modellare delle discontinuità (C0) di attributo lungo un edge, duplicando i vertici ai suoi estremi
 - ⇒ La mesh avrà due vertici geometricamente coincidenti (stessa posizione) ma con attributi associati differenti
 - ⇒ Due edge tecnicamente aperti saranno quindi perfettamente sovrapposti, causando una superficie continua (senza «spiragli»)
 - ⇒ Facce adiacenti useranno una oppure l'altra di questa coppia di vertice



- ⇒ Questa configurazione si chiama un «seam» (letteralmente: cucitura) o «vertex seam»



44

Mesh two-manifold e non

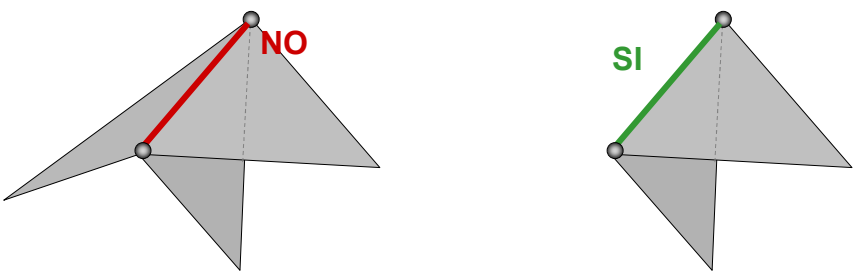
- ✓ una mesh è detta two-manifold (una "varietà due") se *rappresenta in effetti una superficie*
 - ⇒ molti algoritmi di geometry processing necessitano che questo sia il caso!
- ✓ Non tutte le mesh (= insiemi di poligoni che condividono dei vertici e degli edge) lo sono!
 - ⇒ le **facce** di una mesh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




45

Edge two manifold

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

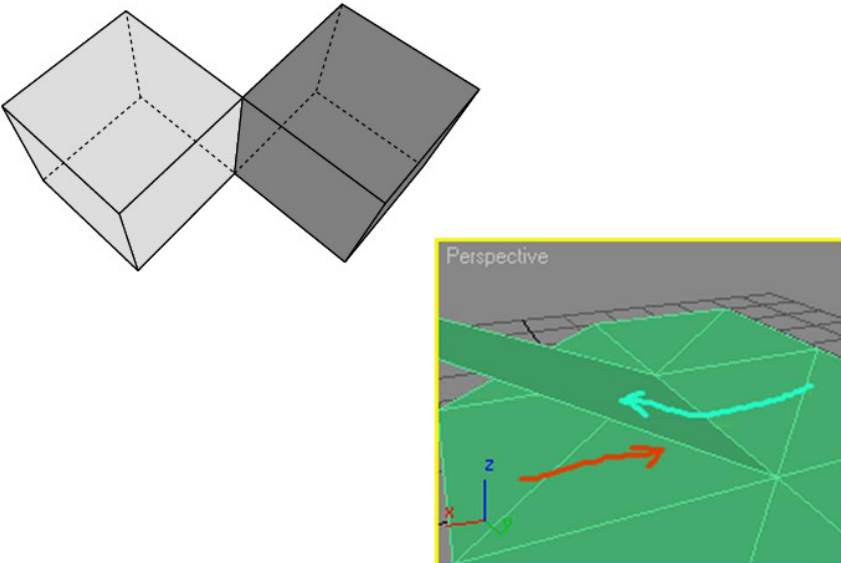


The left diagram shows a red edge shared by three triangular faces, labeled "NO". The right diagram shows a green edge shared by two triangular faces, labeled "SI".




46

Esempi di edge non 2-manifolds



The top diagram shows two cubes sharing a face. The bottom diagram is a perspective view of a mesh with a red arrow pointing to a non-manifold edge and a cyan arrow pointing to a manifold edge.

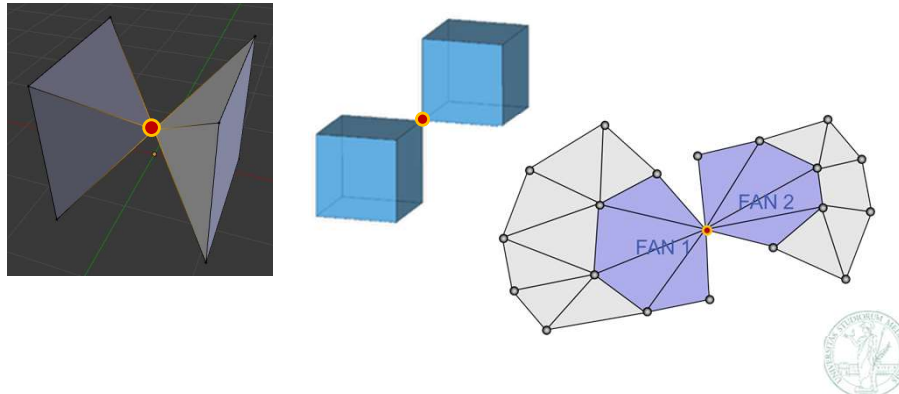


47

Vertice two-manifold

- ✓ due facce sono 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

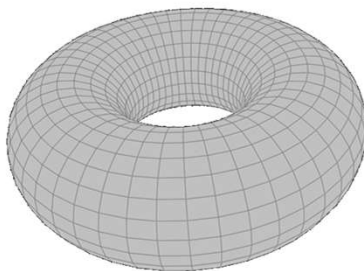
Esempi di vertici non 2-Manifold:



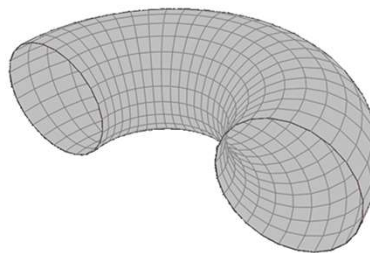
50

Mesh chiuse e aperte

- ✓ un edge condiviso da solo 1 faccia è di bordo;
- ✓ un edge condiviso da 2 facce 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

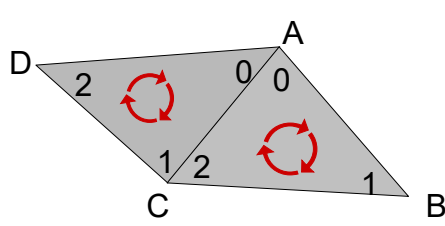


aperta


52

Orientamento delle facce

✓ Una mesh two manifold può essere ben orientata oppure no



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

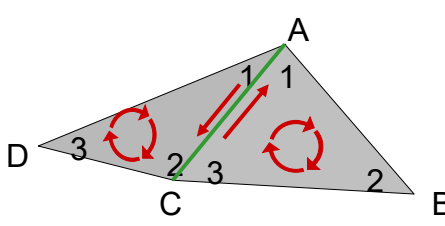


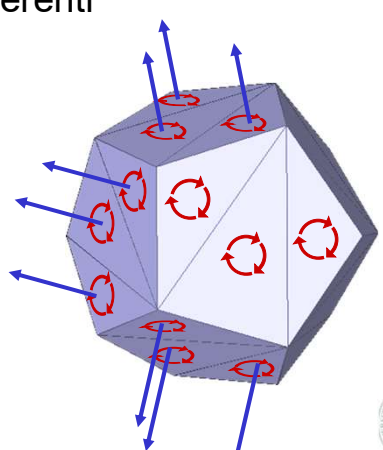
54

Mesh


✓ **Orientabile**, non orientabile

- ⇒ è possibile assegnare un orientamento ad ogni faccia coerentemente?
- ⇒ orientabile → normali coerenti





senso opposto, edge coerente



55

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

File formats per indexed mesh


header
 geometria
 connettività

Mesh in formato OFF
 (“Object File Format”)

Es 0: costruisci un file di testo con
 ←questo contenuto (cut and paste),
 rinominato «test.off»,
 e visualizzalo con meshlab

Es 1: 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?



56

Note sull'Esercizio 1

- ✓ Ogni coppia di indici consecutivi in ogni faccia individua un edge
 ⇒ compreso: dall'ultimo vertice della faccia al primo. E' un array ciclico!
- ✓ 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 vs 5-1)
- ✓ Edge di bordo (o «aperto») se: appare solo una faccia
- ✓ Mesh two-manifold richiede: 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
- ✓ NOTA: tutto questo dipende solo dalla connettività.
 Nulla di questo dipende dalla geometria.



58

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!

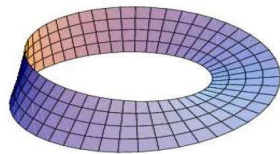
appositi algoritmi



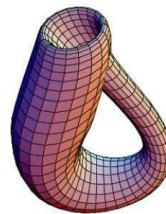
59

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)



60