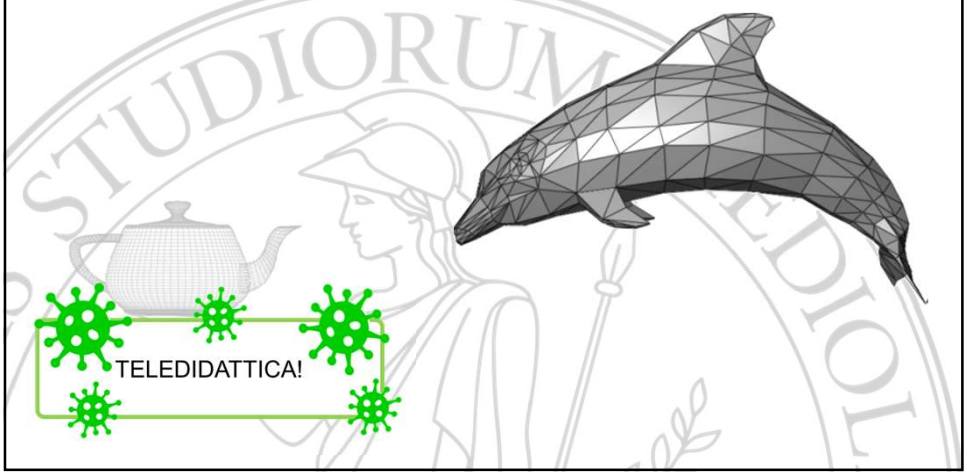
Marco Tarini - Computer Graphics 2021/2021
Università degli Studi di Milano

3D Models: Polygonal Representations



1

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

		ELEMENTI DISCRETI			CONTINUI
		regolari *«a griglia»*	semi-regolari o irregolari: elementi simpliciali	semi-regolari o irregolari: 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)**

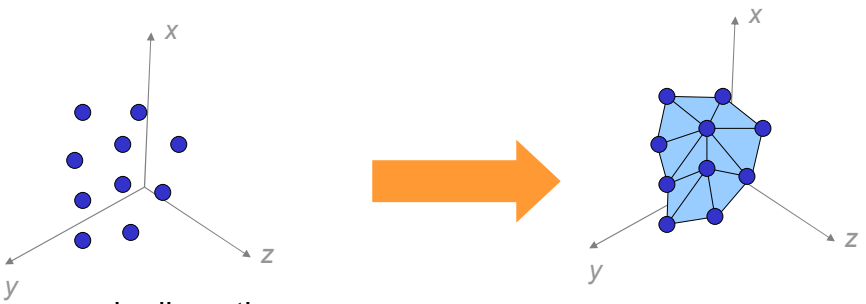
2

Polygonal Mesh (mesh poligonale)

- ✓Superficie approssimata da “facce” poligonali
  - ⇒adiacienti lato a lato (di solito)
- ✓Rappresentazione “lineare a tratti” delle superfici
- ✓Il modello 3D digitale più diffuso!
  - ⇒spesso (e.g.: nei games) è un *sinonimo* di modello 3D
  - ⇒GPU friendly: le schede video sono fatte per renderizzare le mesh (attraverso uno specifico algoritmo, che vedremo: raterization based)

3

Da Nuvola di punti a Triangle Mesh



nuvola di punti

triangoli

problema abbastanza studiato,

Classi di algoritmi: “Front Advancing” (come “Ball Pivoting”), o volumetric reconstructions

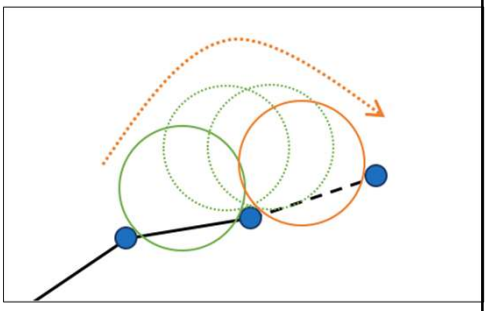
5

Da Nuvola di punti a Triangle Mesh

✓Algoritmi per farlo

⇒Front Advancing:

- es:
  «ball pivoting»

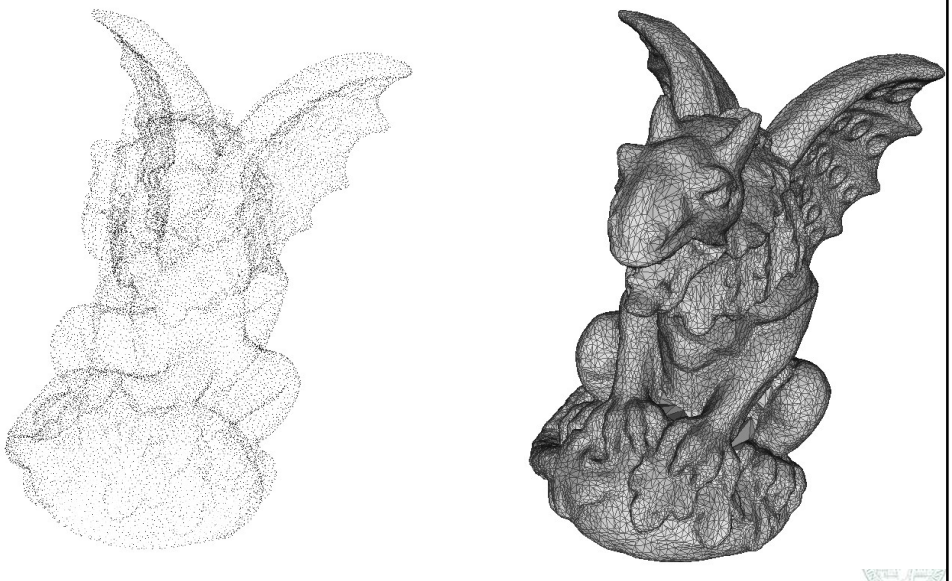


⇒Oppure: Volumetric reconstruction

- Idea:
  1) da nuvola di punti a modello volumetrico,
     (es: attraverso «Poisson reconstruction»),
  2) da modello volumetrico a mesh poligonale
     (es: attraverso «marching cubes»)
- Vedere lezione su modelli volumetrici

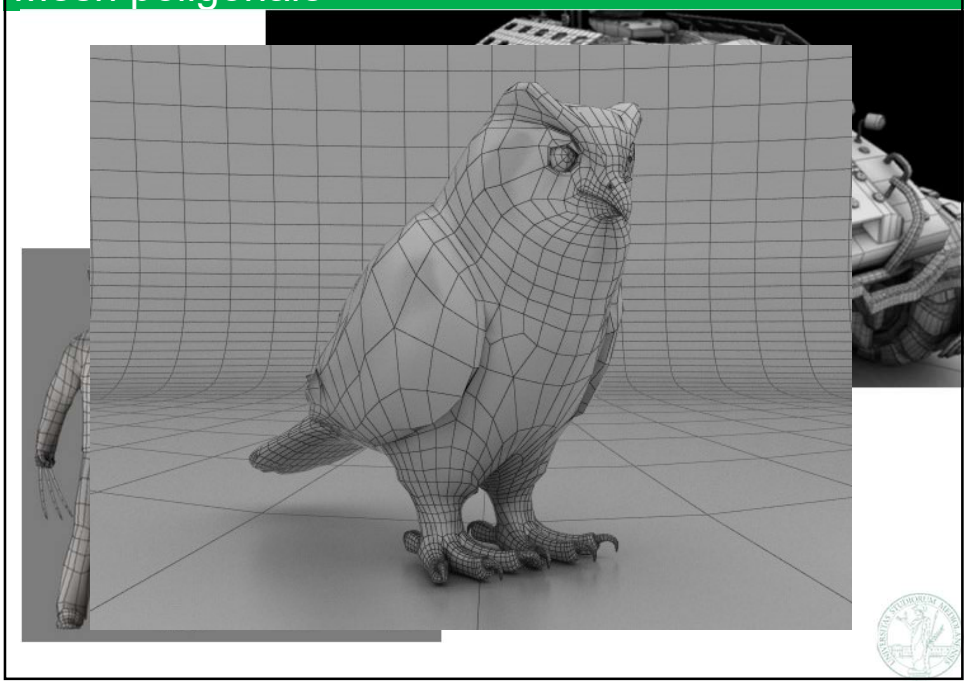
6

Esempio



7

Mesh poligonale



8

10

Mesh poligonali

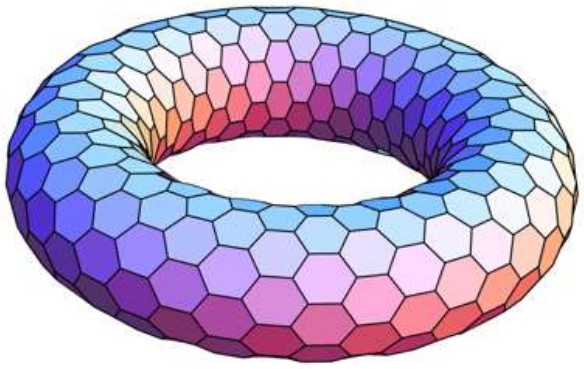
I poligoni possono essere:

- ✓tutti triangoli:
  - ⇒Triangular mesh, o tri-mesh, o “mesh simpliciale”
- ✓tutti 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)

11

Mesh poligonali

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



12

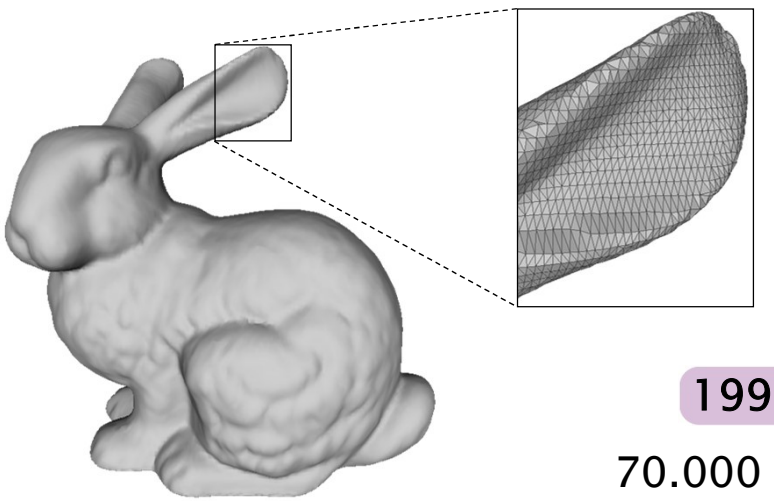
Mesh Polionali: risoluzione

- ✓ 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 ≅ 2 × num vertici
per quad-mesh:
num facce ≅ num vertici

13

Risoluzione tipica: crescente negli anni

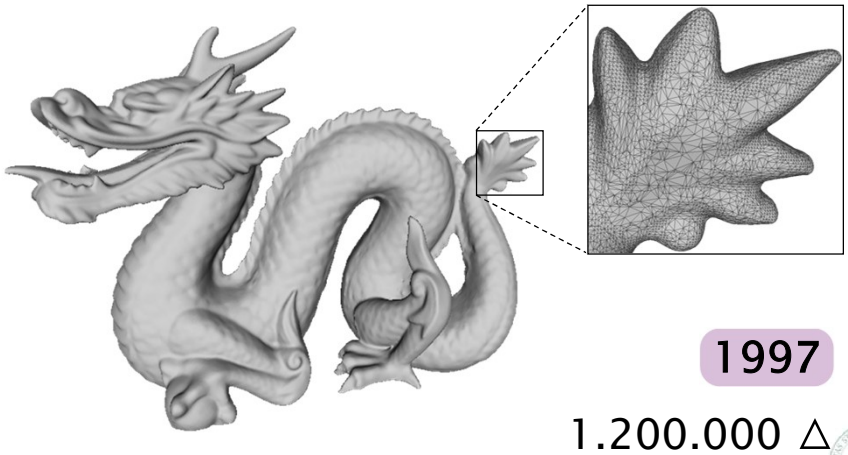


1994

70.000 △

14

Risoluzione tipica: crescente negli anni

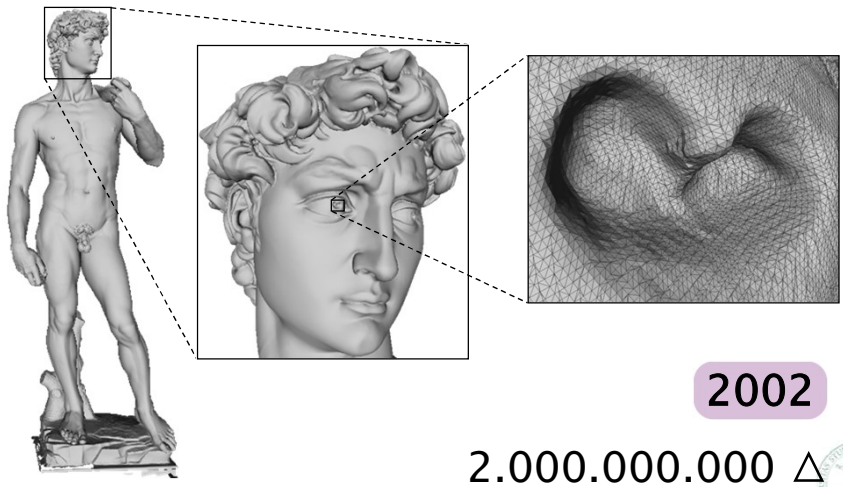


1997

1.200.000 △

15

Risoluzione tipica: crescente negli anni

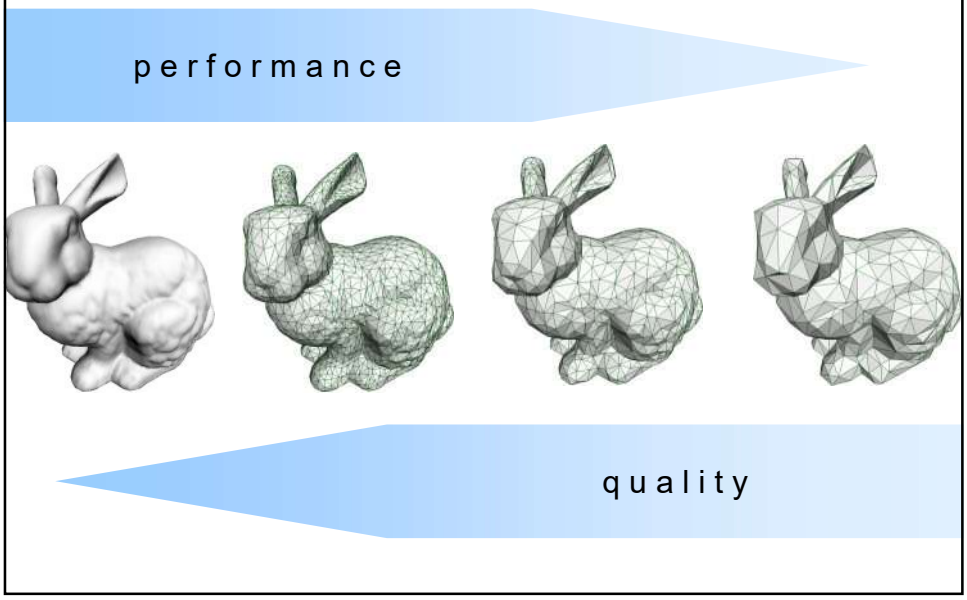


2002

2.000.000.000 △

16

3D Meshes: resolution



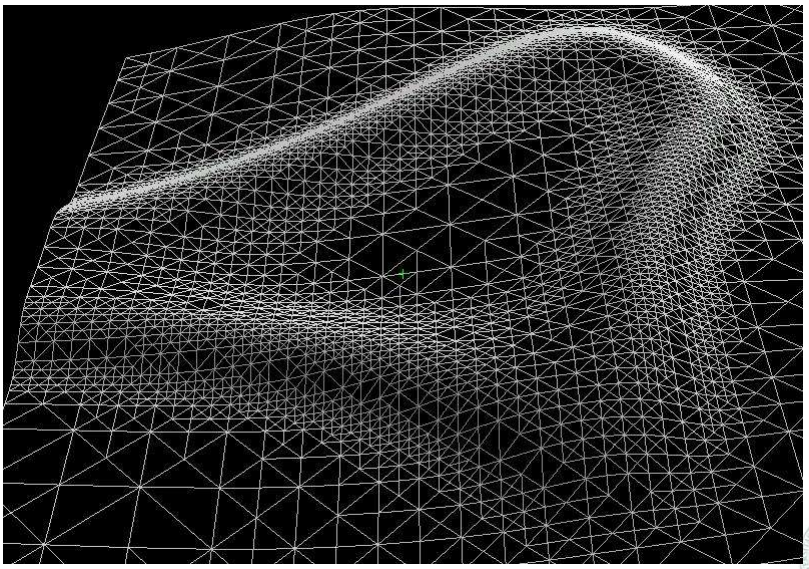
17

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

18

3D Meshes: *adaptive* resolution



19

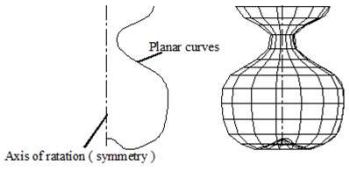
Mesh Triangolare o Tri-meshes

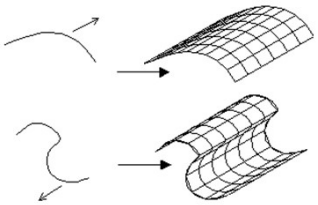
- ✓ Vantaggio: facce sempre planari
  - ⇒ tre punti nello spazio sono sempre co-planari
- ✓ Vantaggio: semplice modo di interpolare attributi (vedi poi)
- ✓ Modo preferito dai matematici
  - ⇒ la chiamano “mesh simpliciale”
  - ⇒ matematicamente: è una una approssimazione “lineare a tratti” della superficie che stiamo rappresentando
- ✓ Modo preferito dalle GPU
  - ⇒ vedremo, l’unico tipo di mesh che possono essere renderizzate direttamente dall’Hardware specializzato per la CG (schede video)
  - ⇒ altri poligoni vengono scomposti in triangoli!
- ✓ Di gran lunga il tipo di mesh più usato in praticamente tutte le applicazioni di CG

20

Pure quad-mesh

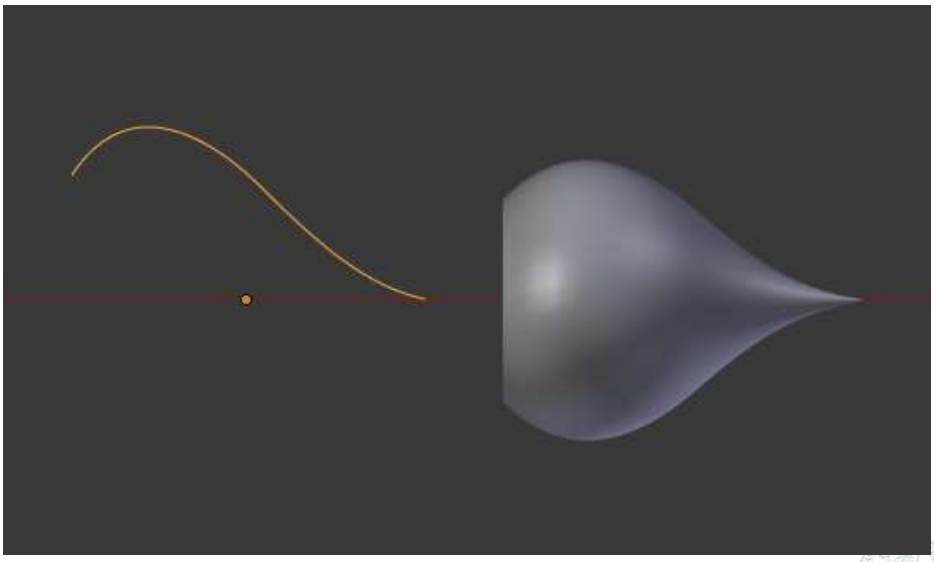
- ✓ Generalmente, più difficili da costruire
  - ⇒ Esistono un numero minore di tecniche per produrle
  - ⇒ Alcune eccezioni:
  - ⇒ superfici «di rivoluzione»
  - ⇒ per es, superfici «di spazzata»





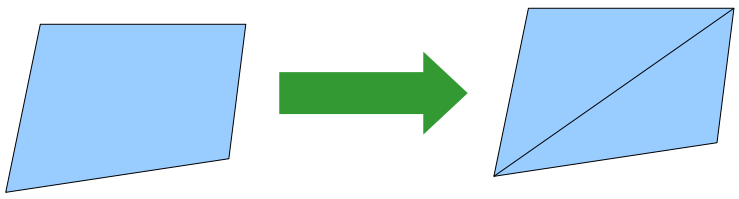
- ✓ Richieste da alcuni algoritmi
  - ⇒ es, di suddivisione (vedi poi)

21



22

Da *Quad meshes* a *Triangle meshes*



"un quad"

due triangoli!

"diagonal split"

23

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...)

24

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**

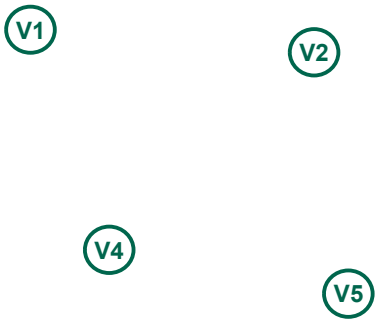
- Definiti sulla superficie
- es: colore, normali, UV, (indice di) materiali, …

25

Mesh: geometria

✓Insieme di posizioni dei vertici

⇒Un vettore posizione (x,y,z) per ogni vertice



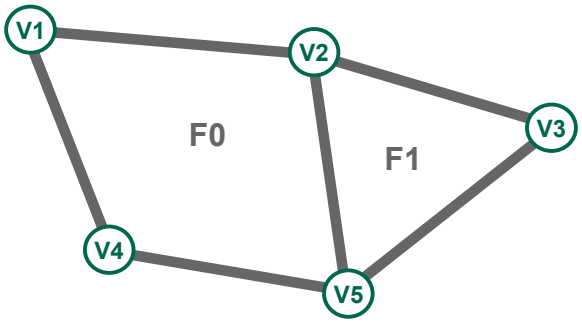
26

Mesh: connettività (o topologia)

✓Facce

⇒poligoni che connettono fra loro i vertici

⇒simile a: nodi connessi da archi, in un grafo



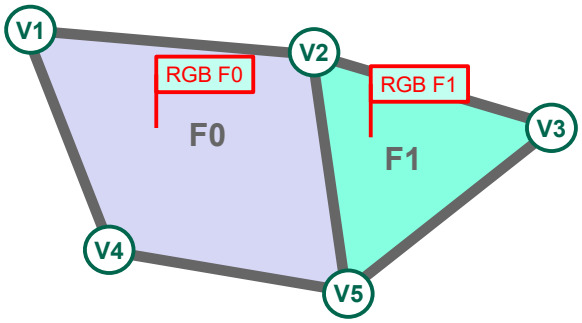
27

Mesh: attributi

✓Quantità che variano sulla superficie

⇒es: colore RGB

⇒definiti per faccia, costanti su ogni faccia?



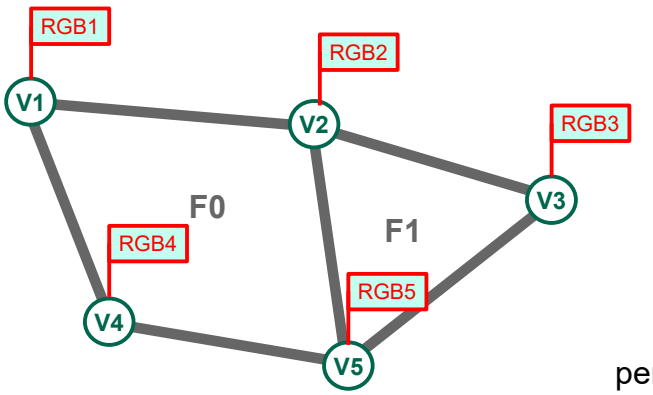
per faccia

28

Mesh: attributi

✓Quantità che variano sulla superficie

⇒Campionati per vertice, interpolati nelle facce



per vertice

29

Come rappresento internamente una mesh?

Modo diretto: (anche detto: zuppa di triangoli)

✓un lungo vettore di poligoni

⇒per ogni poligono (di n lati):
è 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

31

Come rappresento internamente una mesh?

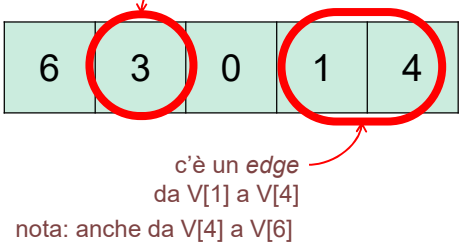
✓ Modo **indexed mesh**:

⇒**Geometria**: array di vertici V[0] … V[N]
- 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»)

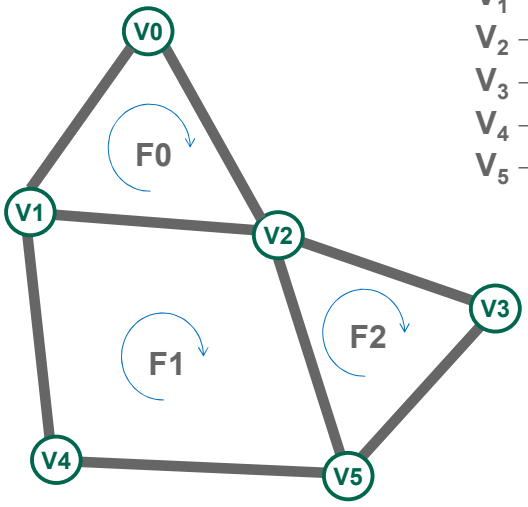
ordinata e ciclica

es.: una faccia pentagonale: (5 corners)



32

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

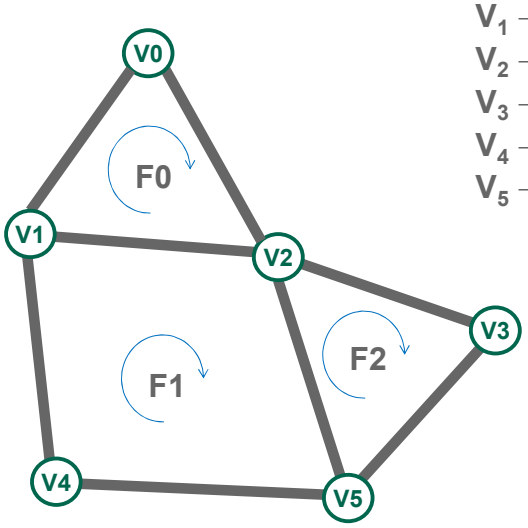
GEOMETRIA + ATTRIBUTI

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

CONNETTIVITA'
(lista facce)

33

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)

34

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

```
class Vertex {
  vec3 pos;
};

class Face{
   vector<int> vertexIndex;
   rgb color;    /* attribute 1 */
   vec3 normal; /* attribute 2 */
};

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

35

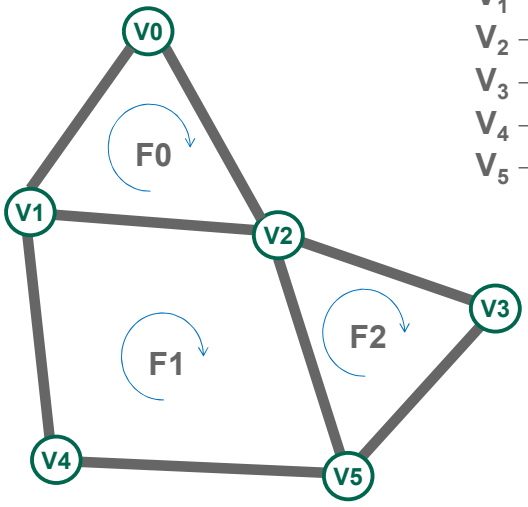
Indexed mesh: as a class (here: in C++)

```
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: normali memorizzate per faccia, e la mesh è pure-quad (4 indici di vertice per faccia)

36

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

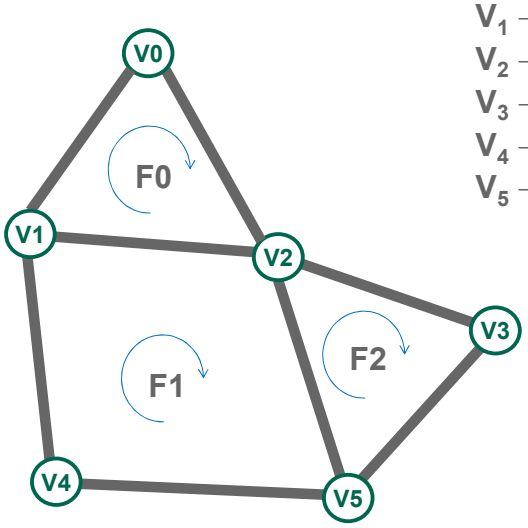
GEOMETRIA + ATTRIBUTI

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

CONNETTIVITA'
(lista facce)

37

Struttura dati: indexed mesh (piccola variante)



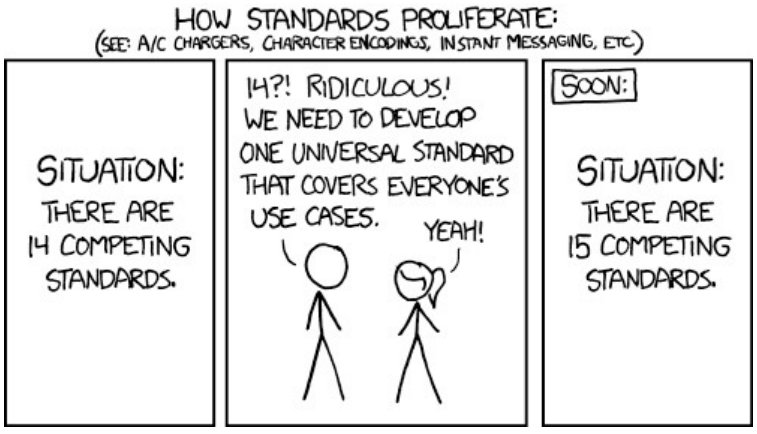
	GEOMETRIA	ATTRIBUTI
V_0 →	x_0, y_0, z_0	r_0, g_0, b_0
V_1 →	x_1, y_1, z_1	r_1, g_1, b_1
V_2 →	x_2, y_2, z_2	r_2, g_2, b_2
V_3 →	x_3, y_3, z_3	r_3, g_3, b_3
V_4 →	x_4, y_4, z_4	r_4, g_4, b_4
V_5 →	x_5, y_5, z_5	r_5, g_5, b_5

(due vettori paralleli)

	CONNETTIVITA' (lista facce)
F_0 →	0, 2, 1
F_1 →	1, 2, 5, 4
F_2 →	2, 3, 5

38

Formati di file per mesh



(from **xkcd.com**)

39

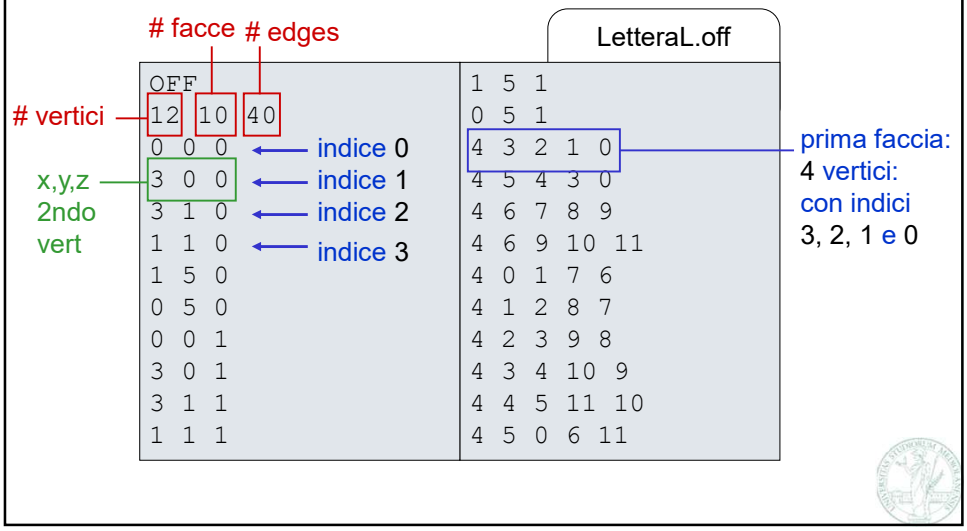
Solo alcuni dei formati utilizzati
I primi nella lista sono forse più diffusi

- 3DS - 3D Studio Max
- OBJ – Uno dei più diffuse. Molto semplice
- MA, MB - di Maya
- OFF – ancora più emplice
- 3DX - Rinoceros file format
- BLEND - Blender file format
- DAE - **COLLADA** (by Khornos, "universale")
- FBX - Autodesk interchange file format
- X - Direct X object
- SMD - (by Valve) Mesh animate
- BVH – Behavioral file (mesh animate)
- MD3 - quake 3 vertex animations
- DEM - Digital Elevation Models
- DXF - (Autodesk's AutoCAD, for CAD)
- FIG - Used by REND386/AVRIL
- FLT - MulitGen Inc.'s OpenFlight format
- IGES - Graphics Exchange Specification
- IV - Open Inventor File Format Info
- LWO, LWB & LWS - Lightwave 3D file formats
- MAZ - Used by Division's dVS/dVISE
- MGF - Materials and Geometry Format
- MSDL - Manchester Scene Description Language
- 3DML - by Flatland inc.
- C4D – Cinema 4D file format
- SLDPTR - SolidWork "part"
- WINGS - Wings3D object
- NFF - Used by Sense8's WorldToolKit
- SKP - Google sketch up
- KMZ - Google Earth model
- OOGL - Object Oriented Graphics Library
- PLG - Used by REND386/AVRIL
- POV - "persistence of vision" per ray-tracer
- QD3D - by apple, pple's QuickDraw 3D Metafile format
- TDDD - for Imagine & Turbo Silver r
- NFF & ENFF - (Extended) Neutral File Format
- VIZ - Used by Division's dVS/dVISE
- VRML, VRML97 Virtual Reality Modeling Language (RIP)
- X3D - tentato successore di VRML
- PLY - introdotto by Cyberware – tipic. dati range scan
- DICOM - Dalla casa omonima – tipic. dati CAT scan
- Renderman - per l'omonimo renderer
- RWX - RenderWare Object
- Z3D - ZModeler File format
- HDF - Hierarchical Data Format
- etc

40

File formats per indexed mesh:

Per esempio: OFF ("Object File Format")



41

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

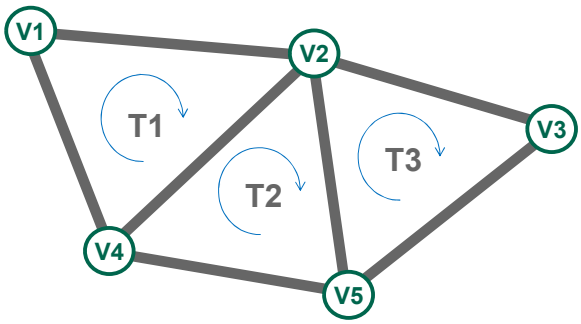
Mesh in formato OFF
("Object File Format")

Quale oggetto rappresenta?

42

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'

43

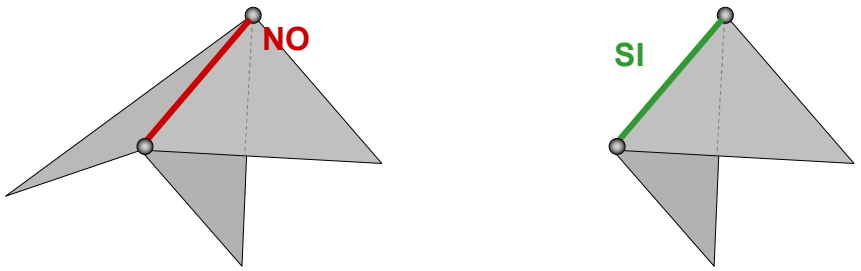
Mesh two-manifold e non

- ✓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

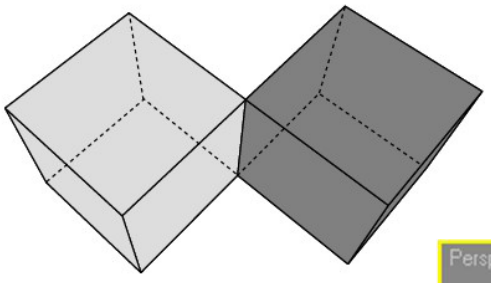
Edge two manifold

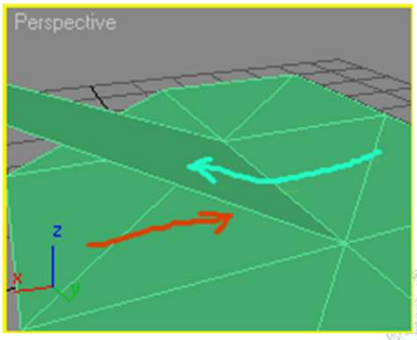
- ✓un edge interno è two-manifold se è condiviso da al massimo due facce



46

Esempi di edge non 2-manifolds



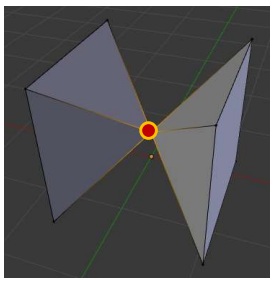


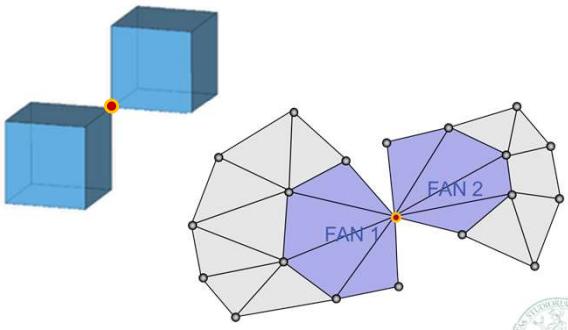
47

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 fan (lett.: un ventaglio)
- ✓ un vertice è two manifold se ha un solo fan

Esempi di vertici non 2-Manifold:

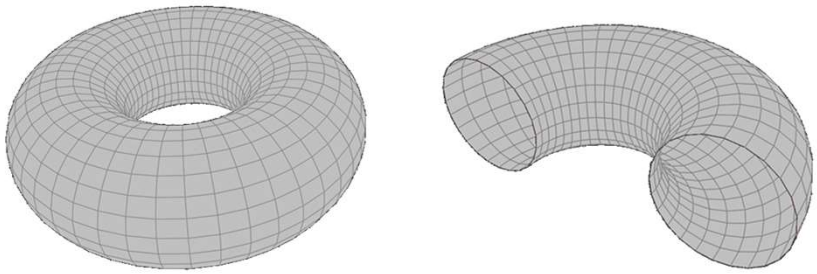




50

Mesh chiuse e aperte

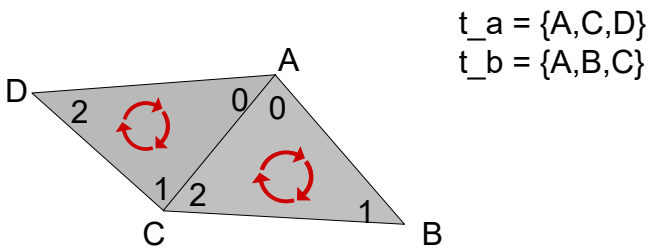
- ✓ un edge condiviso da solo 1 faccia è di bordo;
- ✓ un edge condiviso da 2 faccie è interno;
- ✓ se una mesh non ha edge di bordo, è chiusa (altrimenti, è aperta)



52

Orientamento delle facce

- ✓ Una mesh two manifold può essere ben orientata oppure no



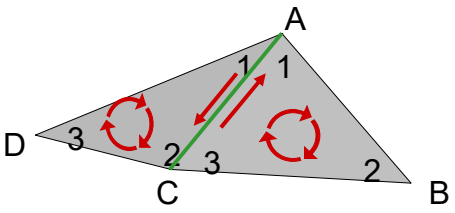
54

Mesh

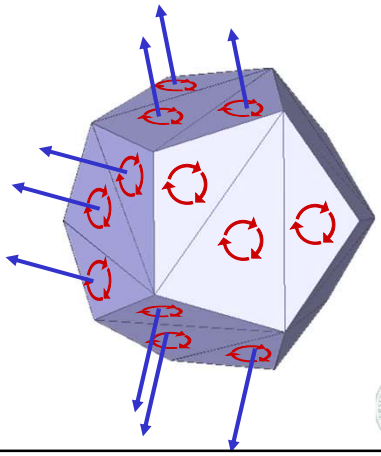
✓Orientabile, non orientabile

⇒è possibile assegnare un orientamento ad ogni faccia coerentemente?

⇒orientabile → normali coerenti



senso opposto, edge coerente



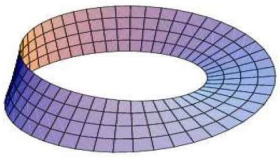
55

Circa la connettività di una mesh

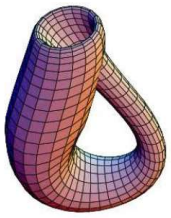
✓Orientabile, non orientabile

⇒esempi di mesh non orientabili:

- mesh non two-manifold
- e...



Nastro di Moebius
(non orientabile, aperta)

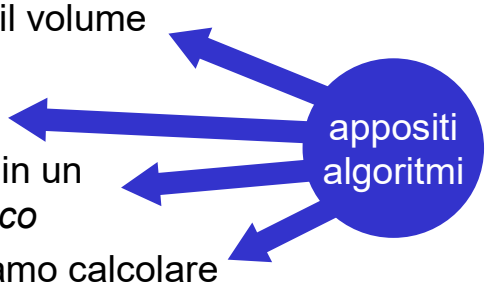


Bottiglia di Klein
(non orientabile, chiusa)

56

Osservazione

- ✓Una mesh two-manifold, ben-orientata e chiusa separa uno spazio *interno* da quello *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!



57

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")

Es 0: costruisci un file di testo con
←questo contenuto (cut and paste),
rinominalo «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?

58