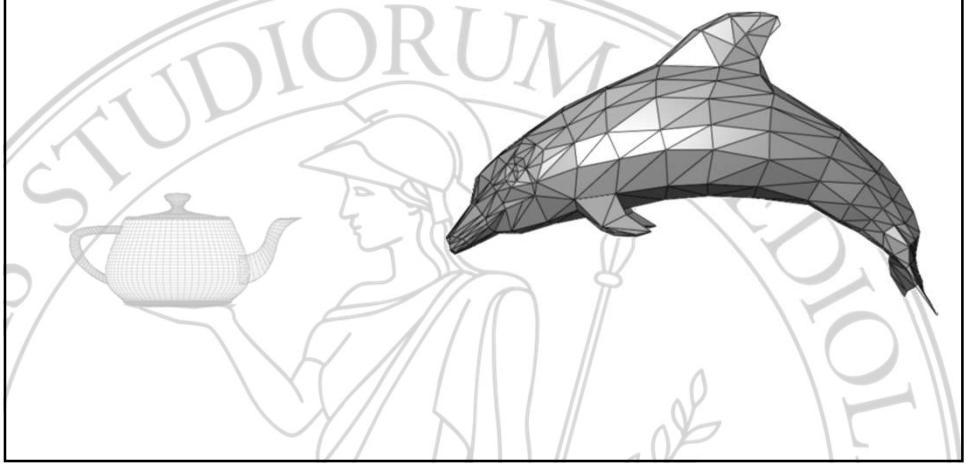
Marco Tarini - Computer Graphics 2024/2025
Università degli Studi di Milano

3D Models: Polygonal Representations



1

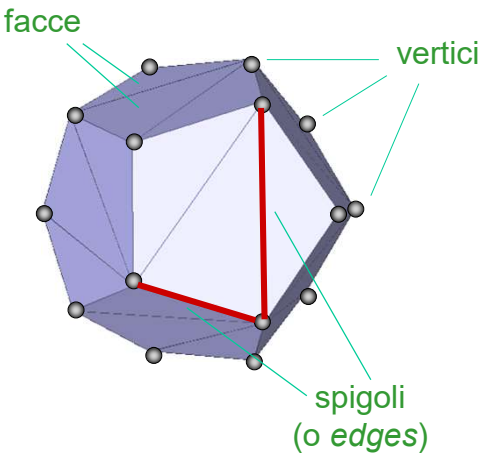
Una (imperfetta) categorizzazione dei tipi di modelli digitali 3D

		ELEMENTI DISCRETI			CONTINUI
		regolari	semi-regolari o irregolari		
		«a griglia»	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 (cioè congruenti) lato a lato

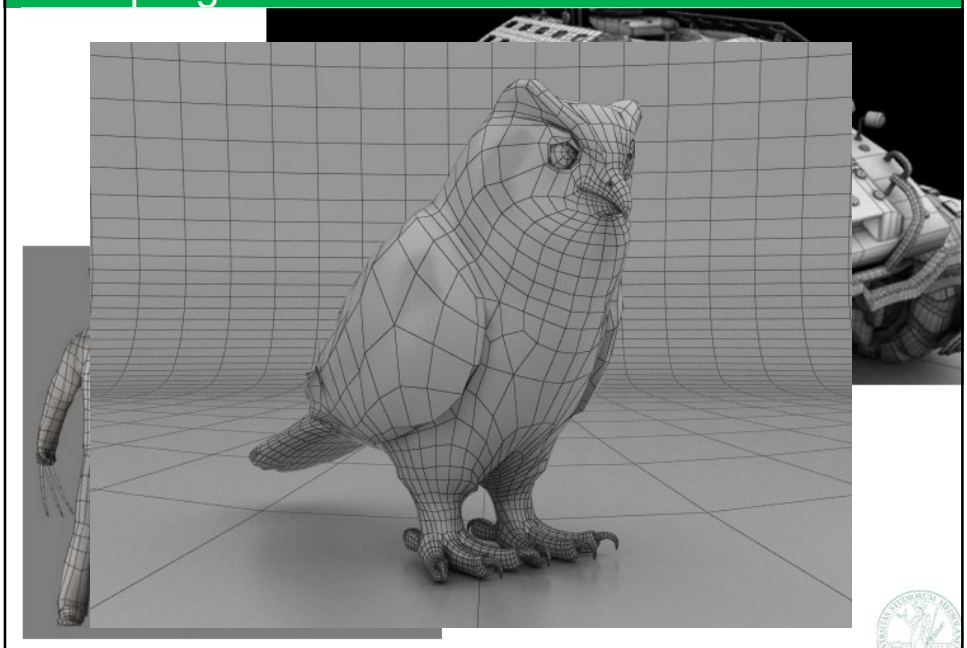
✓Il modello 3D digitale più diffuso!

⇒In molti contesti (per es, nei games) mesh poligonale 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: raterization-based)

5

Mesh poligonale



6

Mesh di poligonale: struttura dati

✓ Componenti:

⇒**geometria**
- i vertici, ciascuno con pos (x,y,z)
- è 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, …
- Campionano una quantità che varia sulla superficie

9

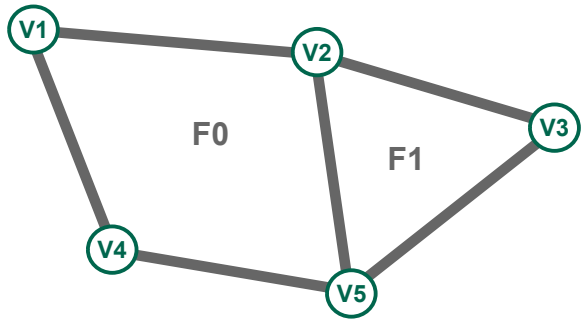
Mesh: geometria

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

10

Mesh: connettività (o topologia)

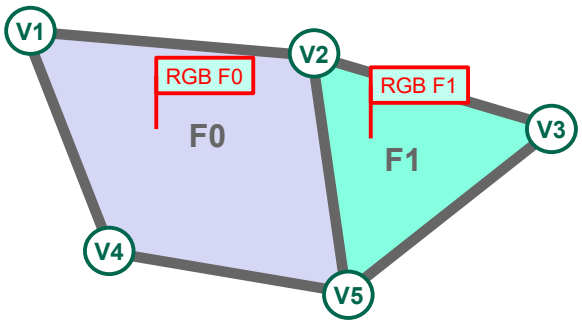
- ✓ Facce
  - ⇒ poligoni che connettono fra loro i vertici
  - ⇒ simile a: nodi connessi da archi in un grafo



11

Mesh: attributi

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

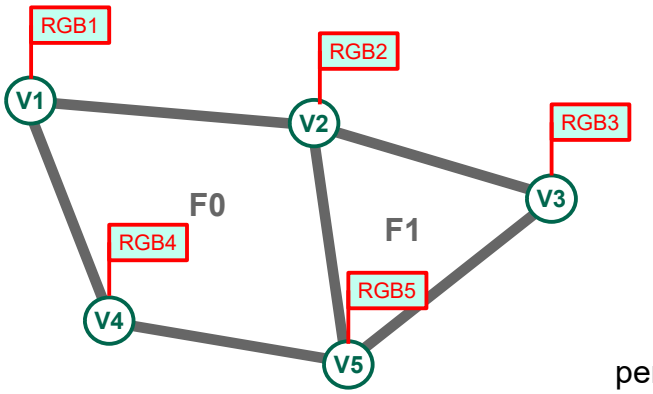


per faccia

12

Mesh: attributi

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



per vertice

13

Mesh poligonali: tipologie

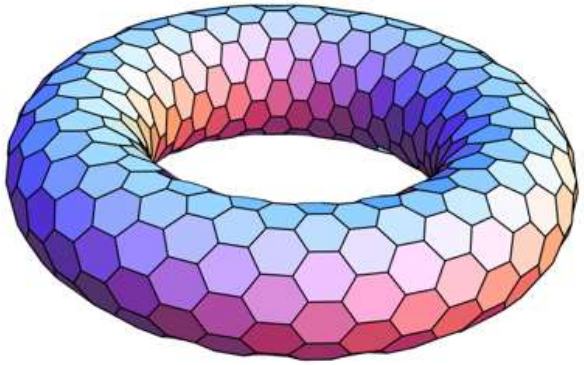
I poligoni possono essere:

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

14

Mesh poligonali

- ✓ volendo, esistono anche le hesagonal-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

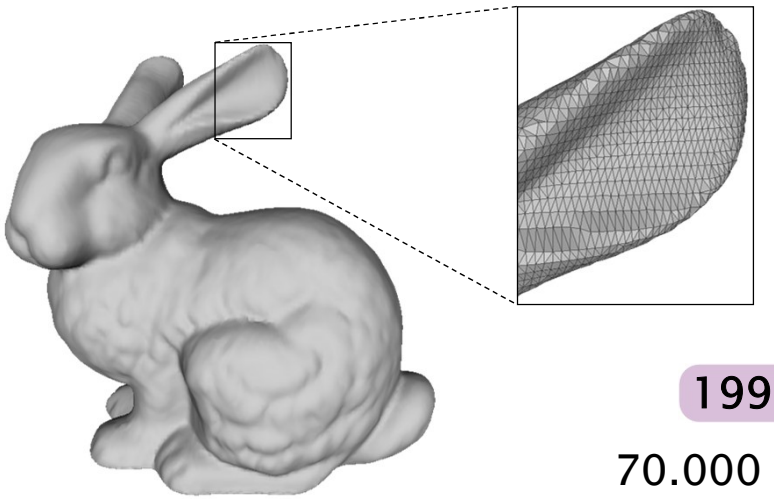
num facce lineare con num vertici.
Statisticamente:
per tri-mesh:
num facce ≅ 2 × num vertici
per quad-mesh:
num facce ≅ num vertici

⇒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)

16

Risoluzione tipica: crescente negli anni

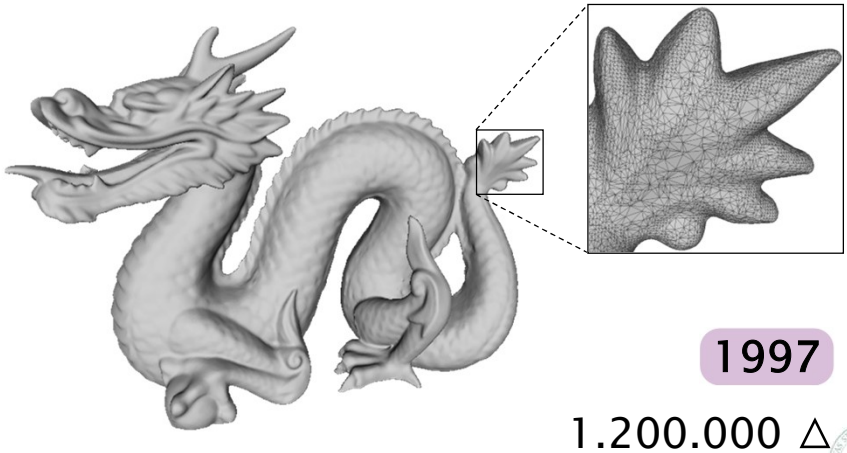


1994

70.000 △

17

Risoluzione tipica: crescente negli anni

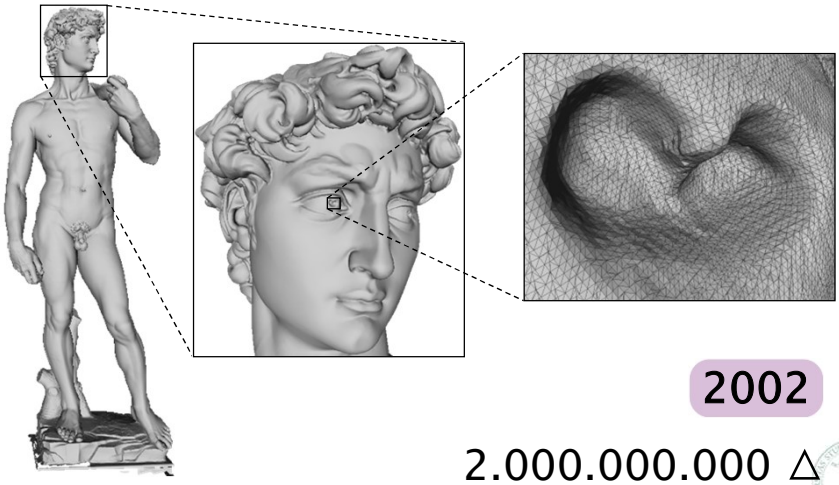


1997

1.200.000 △

18

Risoluzione tipica: crescente negli anni

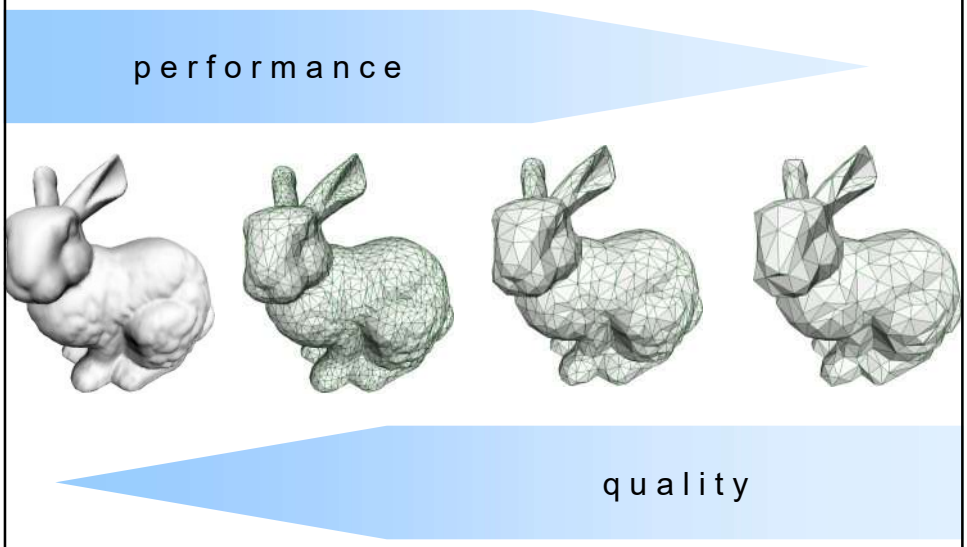


2002

2.000.000.000 △

19

3D Meshes: resolution



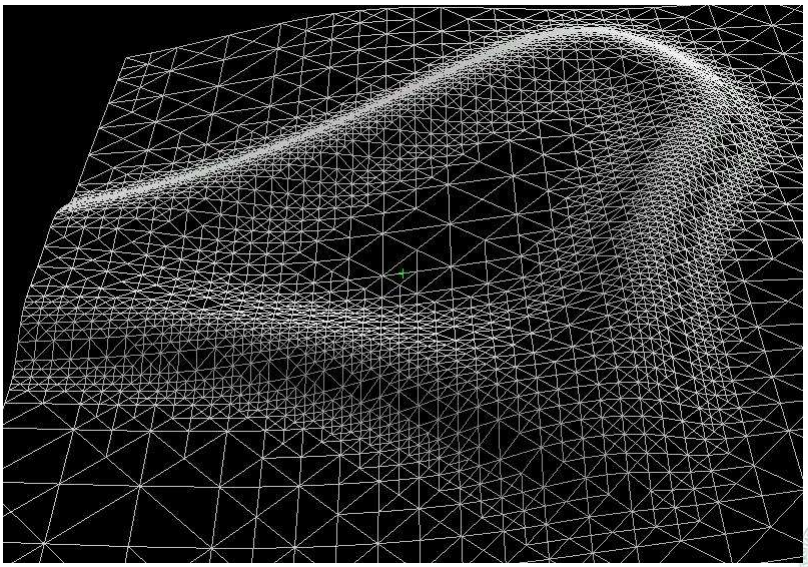
20

Mesh Polionali: risoluzione

- ✓ La risoluzione di una mesh può essere adattiva: tassellamento più fine (campionamento più fitto da un numero maggiore di triangoli di dimensione minore) 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

3D Meshes: *risoluzione adattiva*



22

Mesh Triangolare o Tri-meshes

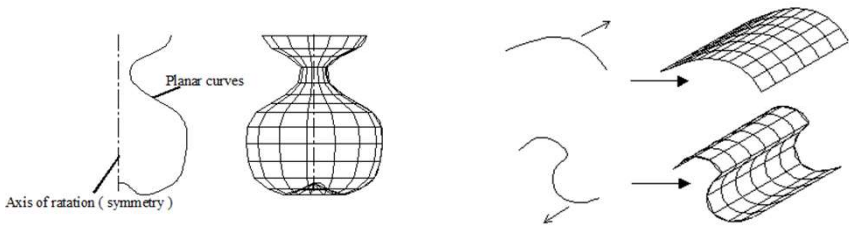
- ✓ Vantaggio: le facce sono sempre planari
  - ⇒ Perché tre punti nello spazio sono sempre co-planari
  - ⇒ matematicamente: la mesh è una approssimazione “lineare a tratti” della superficie che stiamo rappresentando
  - ⇒ Così come una linea curva può essere approssimata da *segmenti dritti*, una superficie curva può essere approssimata da *triangoli piatti*
- ✓ Per questo, è 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

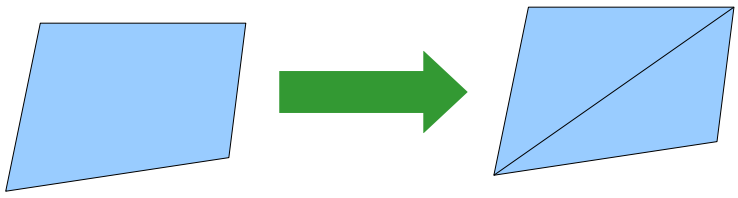
- ✓ Generalmente, più difficili da generare automaticamente
- ✓ Ma molto usate da modellatori umani (per esempio, nel CAD, o da artisti digitali)

⇒Esempio, superfici «di spazzata»:



24

Da *Quad meshes* a *Triangle meshes*



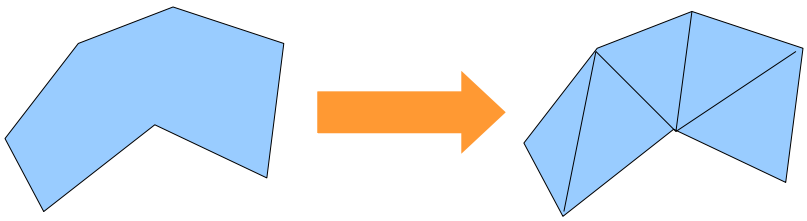
"un quad"

due triangoli!

"diagonal split"

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 **indexed mesh**:

⇒**Geometria**: array di N vertici V[0] … V[N-1]

- per ogni vertice: posizione e eventualm. 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»)

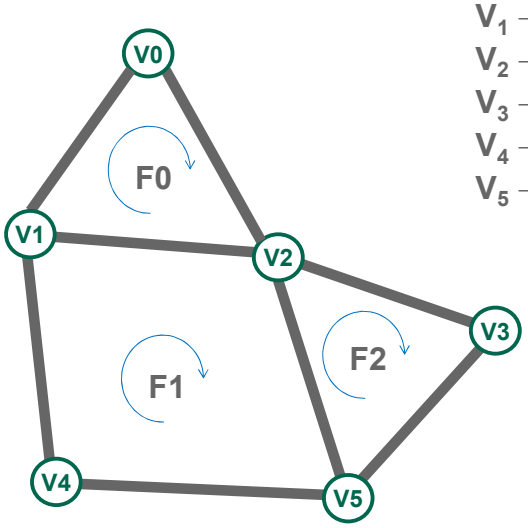
ordinata e ciclica

es.: una faccia pentagonale: (5 corners)

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

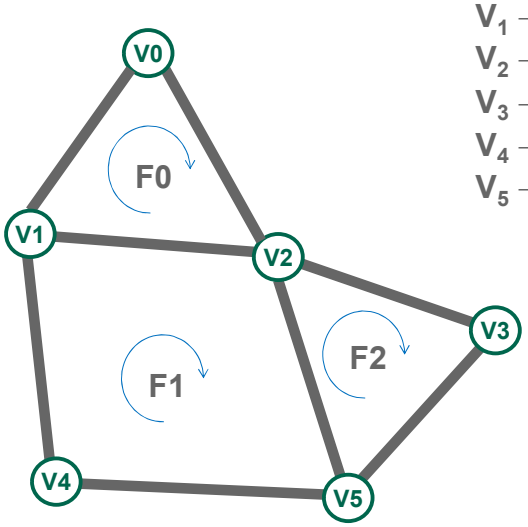
GEOMETRIA + ATTRIBUTI

$F_0 \rightarrow$	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)

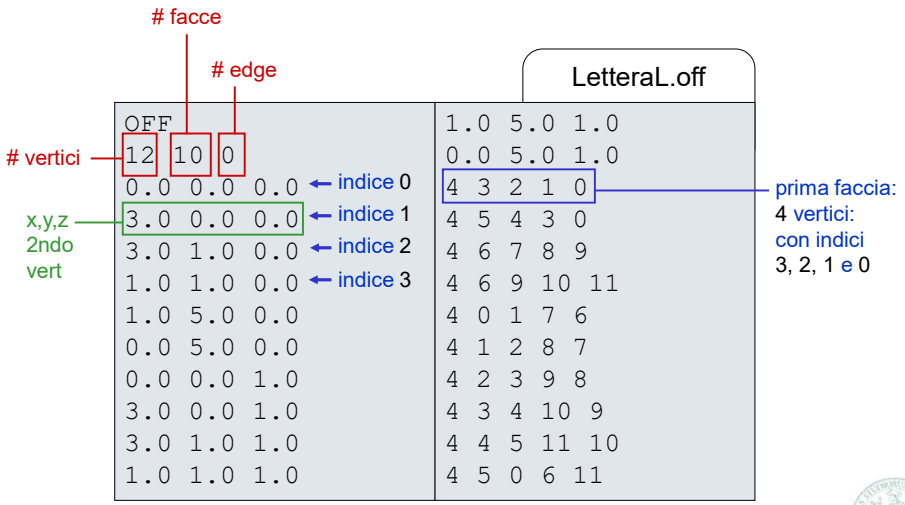
$F_0 \rightarrow$	0, 2, 1
$F_1 \rightarrow$	1, 2, 5, 4
$F_2 \rightarrow$	2, 3, 5

CONNETTIVITA'
(lista facce)

32

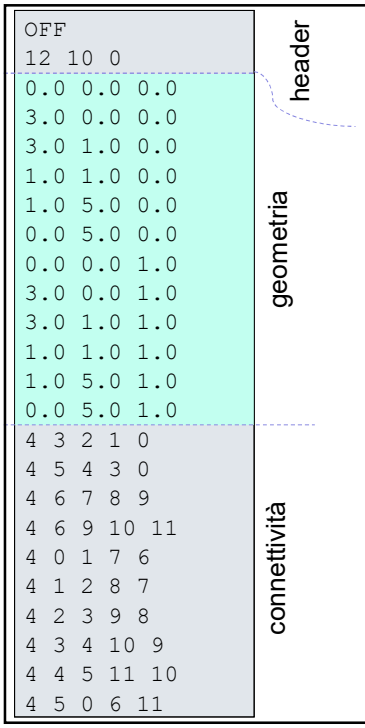
File formats per indexed mesh:

Per esempio: OFF (“Object File Format”)



33

File formats per indexed mesh



Mesh in formato OFF
(“Object File Format”)

Quale oggetto rappresenta?

34

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
    - è 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

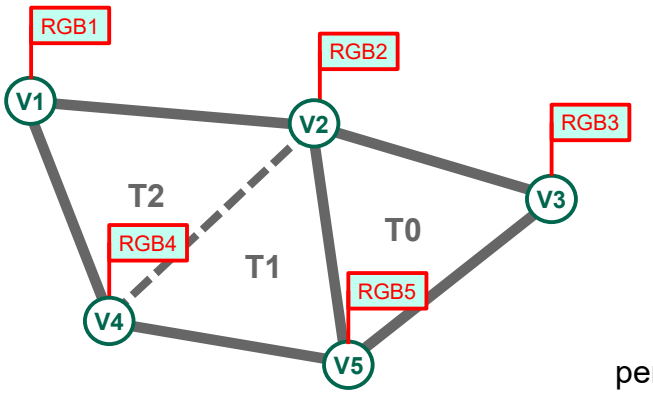
Mesh: attributi

- ✓ 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 constanti 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)

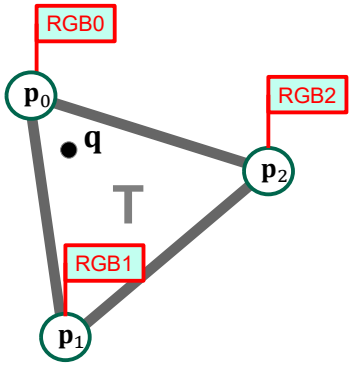
41

Interpolazione degli attributi per vertice dentro ai triangoli

Esempio:

- ✓ Un triangolo **T** ha questi tre attributi colore assegnati ai suoi vertici $\mathbf{p}_0,\ \mathbf{p}_1,\ \mathbf{p}_2$
- ✓ Prendiamo un punto $\mathbf{q}$ su **T** di coordinate baricentriche k_0, k_1, k_2 in **T** , cioè con

$$\mathbf{q} = k_0\ \mathbf{p}_0 + k_1\ \mathbf{p}_1 + k_2\ \mathbf{p}_2$$



- ✓ Allora a $\mathbf{q}$ assegno il colore

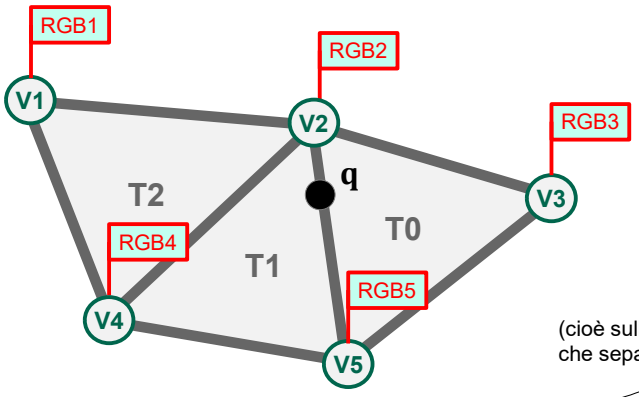
$$k_0\ \boxed{\text{RGB0}} + k_1\ \boxed{\text{RGB1}} + k_2\ \boxed{\text{RGB2}}$$

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 (C0)



(cioè sull'edge che separa T0 e T1)

- ✓ Infatti: si consideri un punto $\mathbf{q}$ a cavallo fra T0 e T1:
- ✓ quale attributo è associato a $\mathbf{q}$, se lo si considera parte di T0?
- ✓ quale attributo è associato a $\mathbf{q}$, se lo si considera parte di T1? (lo stesso!)

43