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

3D Models: Polygonal Representations - Part 2

60

Alcuni dei formati utilizzati (i primi della lista sono i più comuni)

- 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

61

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)

62

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)

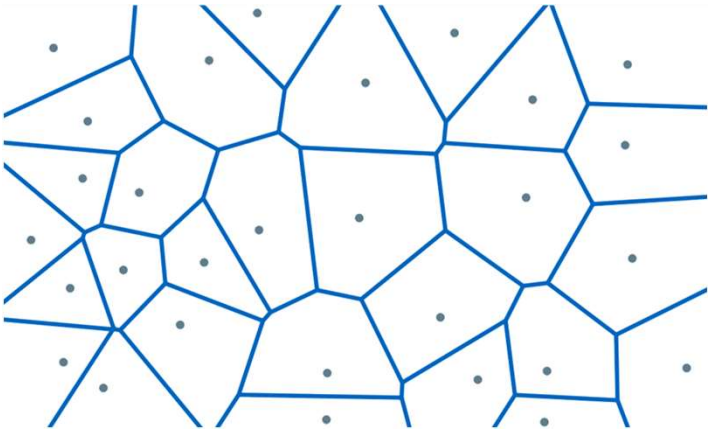


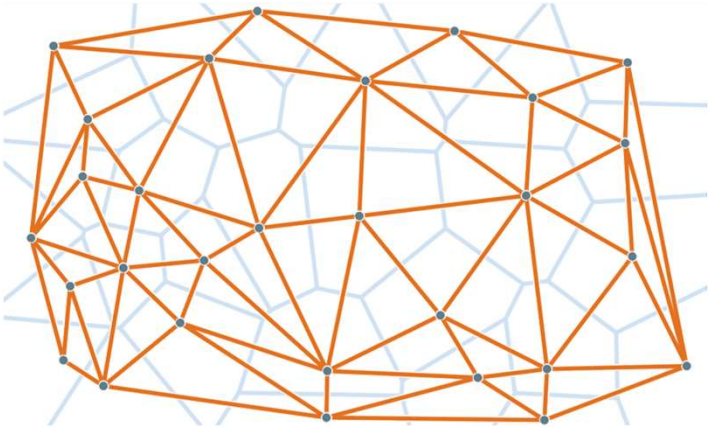
Diagramma di Voronoi

63

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)



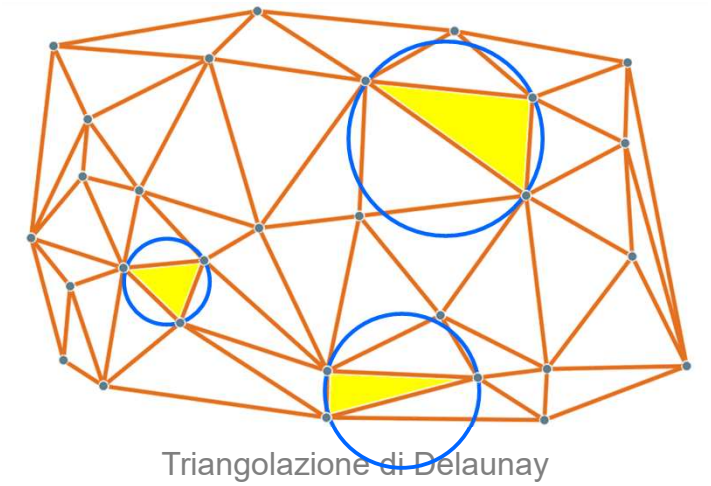
Triangolazione di Delaunay

64

Da Nuvola di punti a Triangle Mesh… in 2D

✓ E', una «buona» triangolazione

⇒Una buona proprietà è garantita



Triangolazione di Delaunay

65

Triangolazione di Delaunay: note

- ✓ Diagramma di Voronoi
  - ⇒ Una scomposizione in regioni di piano indotta da n punti 2D (detti "seed") $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2,$
  - ⇒ Ogni seed produce una regione
  - ⇒ Una 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
  - ⇒ Si ottiene una triangolazione con ottime proprietà, come ad esempio: il circocentro che inscrive ogni triangolo non include nessun altro vertice

66

Da Nuvola di punti a Triangle Mesh passando per…

		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** …	**Point Cloud**		
VOLUMETRICI	(3-manifold)	**Voxels**, **Solid Textures**	**Tetra Mesh**	**Hexa Mesh**	**Implicit model**, **(es. CSG)**

68

Da Nuvola di punti a Triangle Mesh: sommario

- ✓Alcuni metodi aggiungono solo la connettività
  - ⇒la nuvola di punti costituisce già la geometria della mesh
  - ⇒Front Advancing, come Ball Pivoting
  - ⇒In 2D: Delaunay triangulation (da Voronoi diagram)
- ✓Altri metodi ricampionano anche i vertici
  - ⇒Uso di rappresentazioni volumetriche intermedie
  - ⇒vedi lez. dati volumetrici ("Poisson Reconstruction")
  - ⇒Utile, quando la una nuvola di punti è affetta da errori e outlier
  - ⇒In pratica, questi metodi consentono di ottenere denoising e outlier removal come effetto collaterale

69

Alcuni dei formati utilizzati
I primi nella lista sono i 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

71

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)

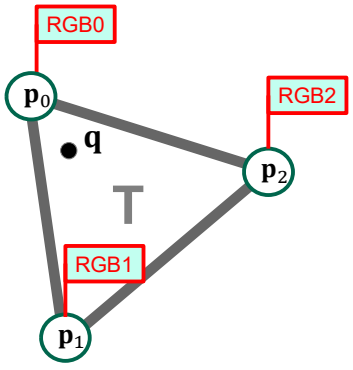
75

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 **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 **q** assegno il colore

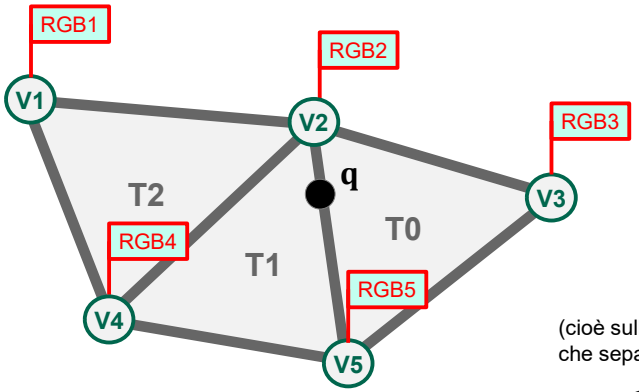
$$k_0\ \boxed{\text{RGB0}} + k_1\ \boxed{\text{RGB1}} + k_2\ \boxed{\text{RGB2}}$$

per vertice

76

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)

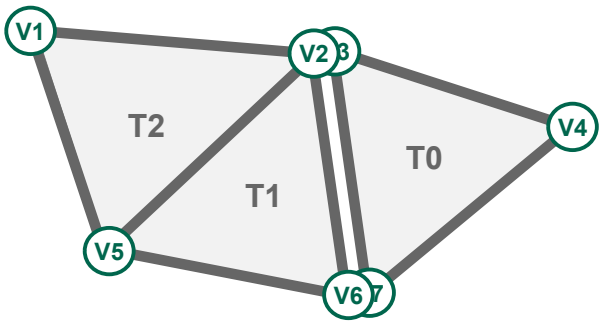


- ✓ Infatti: si consideri un punto **q** a cavallo fra T0 e T1:
- ✓ quale attributo è associato a **q**, se lo si considera parte di T0?
- ✓ quale attributo è associato a **q**, se lo si considera parte di T1? (lo stesso!)

77

Attributi per vertice con discontinuità

- ✓ Se è necessario, è possibile modellare delle discontinuità (C0) di attributo lungo un edge, ma solo 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»

78

Attributi: il caso delle normali

Nel caso in cui l'attributo è la normale
(cioè il vettore unitario ortogonale alla superficie)

- ✓ per faccia:
  - ⇒ la normale costante su ciascuna faccia
  - ⇒ facce (dall'aspetto) piatto
  - ⇒ discontinuità C0 sugli edge:
    edge sono «spigoli taglienti»,
    detti edge di crease, o «hard» edges
- ✓ per vertice
  - ⇒ normale che varia sulla faccia
  - ⇒ facce (dall'aspetto) curvo
  - ⇒ continuità C0: → aspetto «smooth» (liscio),
  - ⇒ (ma non continuità C1)

80

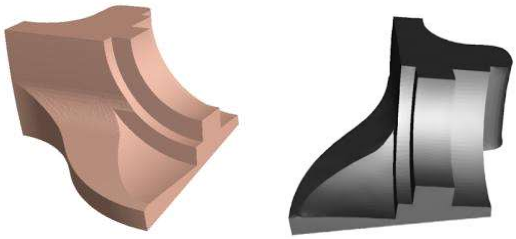
Normali come attributi per vertice di una mesh

- ✓ Normali definite per faccia:
  - ⇒ Le normali sono costanti sulle facce:
  - ⇒ Le facce appaiono piatte
  - ⇒ coerentemente col modello digitale, ma a differenza (spesso) dell'oggetto 3D che si intendeva rappresentare!
  - ⇒ Appaiono crease su ogni edge della mesh
- ✓ Normali definite per vertice:
  - ⇒ Le normali variano sulle facce
    (vengono interpolate dentro le facce, come qualsiasi altro attributo)
  - ⇒ Le facce appaiono in generale curve
  - ⇒ Come ottengo una faccia che appaia piatta?
    Uso una stessa normale come attributo dei tre vertici di un triangolo
  - ⇒ Come ottengo un crease (una discontinuità di normale)? (vedi next)
- ✓ Nota: le normali risultano visibili all'osservatore attraverso l'illuminazione del modello digitale
  - ⇒ Il calcolo dell'illuninazione è un task del rendering (2nda metà del corso)

81

Normali di una superficie: osservazioni

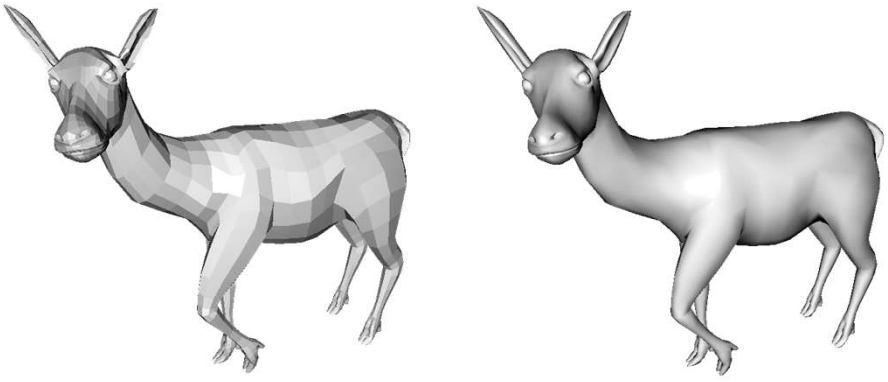
- ✓ Normali costanti => superficie piatta
- ✓ Normali che variano con continuità => superficie curva
- ✓ Normali che variano con discontinuità
  => superficie con un *crease*
  - ⇒ un *crease* è costituito dai punti dove la normale è discontinua



82

Attributi per faccia: normali

Normali per faccia | Normali per vertice



Nota: le normali sono rivelate all'occhio dall'*illuminazione* (computata durante del rendering)

83

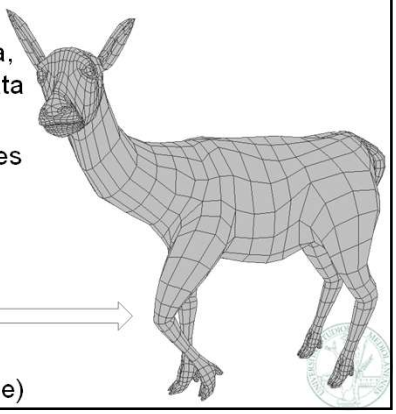
Attributi per faccia: normali

- ✓ Come abbiamo visto, interpolando vettori unitari si ottengono vettori non unitari
  - ⇒ Che vanno quindi ri-normalizzati
- ✓ Questo significa che gli attributi di tipo *vettore unitario*, come le normali, devono essere rinormalizzati dopo ogni l'interpolazione
- ✓ Esercizio: determinare con una stima se questo comporta un onere di calcolo eccessivo, per una GPU, durante un rendering in real-time
- ✓ Traccia
  - ⇒ stimare quante operazioni in virgola mobile (Floating Point Operation) sono necessarie per normalizzare un vettore (soluzione: circa 10)
  - ⇒ stimare quante volte sarà necessario ri-normalizzare un attributo «normale» in un rendering (stimiamo, a braccio: 1 volta in ogni pixel, quindi circa 1000x1000 = 10^6)
  - ⇒ frame per secondo, in un rendering in tempo reale: stimiamo 60
  - ⇒ totale **Fl**oating Point **Op**eration al **S**econdo (FLOPS) necessari a questo passaggio: 10 x 10^6 x 60 = 6x10^8 = 0.6 Giga-FLOPS
  - ⇒ Una GPU in commercio è capace di erogare… Tera-FLOPS = migliaia di Giga-FLOPS

84

Note su rendering delle mesh

- ✓ L'uso di normali come attributo *per faccia* produce un tipo di rendering detto **flat shading**
  - ⇒ Perché le facce appaiono piatte (flat)
  - ⇒ Questo può essere utile per rivelare la forma poligonale delle mesh (ad esempio per poter giudicare la qualità della triangolazione)
- ✓ L'uso di normali come attributo *per vertice* produce un tipo di rendering detto **smooth shading**
  - ⇒ Perché la superficie appare liscia e curva, nascondendo la sua natura poligonalizzata
  - ⇒ Questo è di gran lunga in metodo più utilizzato, per esempio nei videogames
- ✓ Un tipo di rendering che rivela ancora più chiaramente la poligonalizzazione della mesh è detto **wireframe**
  - ⇒ Nel quale vengono disegnati esplicitamente anche gli edge (come linee)



85