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

3D Models: meshes (5/5) Textures & UV mapping

201

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)**

202

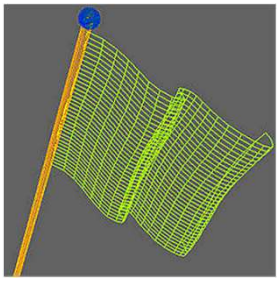
Textures

- ✓ Spesso vorremmo poter campionare un *attributo* (un segnale: come colore, normale, lucentezza, temperatura etc) sulla superficie in modo *molto più denso* di quanto campioniamo la stessa superficie per catturare la sua forma geometrica.

- ✓ Per es:
  - ⇒ forma: ci bastano 10 campioni per cm²
    (sufficiente per rappresentare una statuetta in dettaglio)
  - ⇒ colore: vorremmo 100 campioni per cm²
    (per es, necessario per riprodurre il dettaglio pittorico su un vaso)
- ✓ Memorizzando gli **attributi per vertice** su di una mesh poligonale, ho:
  1 campione di segnale = 1 campione di forma = 1 vertice ☹
  - ⇒ e invece attributi per faccia, guadagno solo un fattore 2 in media
    (se è una mesh di triangoli – fattore 1 se è una mesh di quad)
- ✓ In 2D, le immagini "raster" (griglie 2D regolari di pixel)
  sono un ottimo modo di campionare segnali molto densamente
  - ⇒ es: un'immagine a 100 DPI (DPI = dots per inch²)
    = 40 dpcm (dots per cm²) = 40x40 (1600) campioni (cioè «pixel») per cm²
  - ⇒ es: immagine risoluzione 1000x1000 = 1 Milone di pixel = 1 «MegaPixel»
- ✓ Come ottenere una simile densità di campioni su superfici 3D?

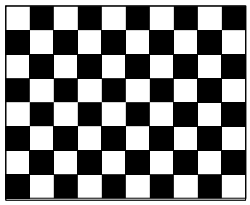
204

Mesh + Texture



Mesh
(pure quad mesh)

+



Texture

(griglia regolare
di "texel" 2D)

=

205

Mesh + Texture

Tri-Mesh

+

Texture
(in questo caso:
un patchwork di foto)

=

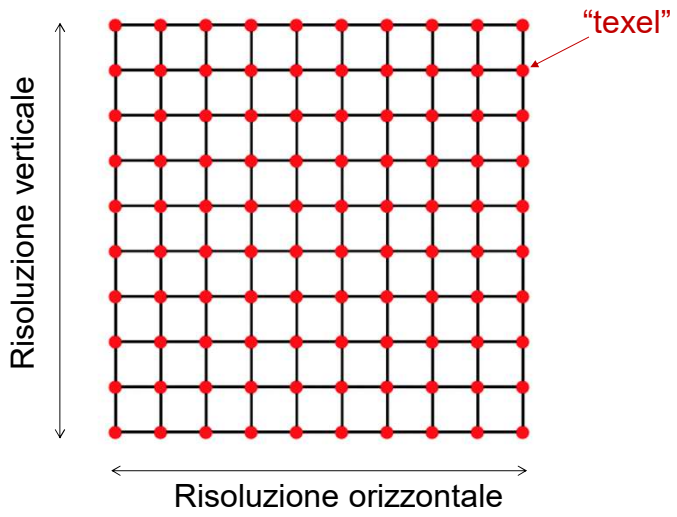
206

Texture Mapping

- ✓ Idea intuitiva: incolliamo un'immagine raster (detta **texture**, o **tessitura**) alla superficie di una mesh
  - ⇒ immagine raster = grigliato regolare di **pixel** (pixel = **pic**ture **el**ement)
  - ⇒ texture = grigliato regolare di **texel** (texel = **tex**ture **el**ement)
- ✓ Ogni punto sulla superficie 3D S ← rappresentata dalla mesh
  viene associato ad un punto su un rettangolo 2D T
  - ⇒ convenzione: le coordinate 2D su T ← anche dette coordinate texture o parametriche
    sono indicate da (u, v) – a volte da (s, t)
    per distinguerle dalle coordinate 3D su S che sono (x, y, z)
  - ⇒ questo mapping da S a T è detto
    UV-map o **UV-mapping** della mesh, ← in contesto industriale
    oppure la sua **parametrizzazione** ← in contesto accademico
- ✓ Domande a cui dobbiamo rispondere:
  - ⇒ come è memorizzata una texture
  - ⇒ come memorizzare lo UV-map sulla mesh?
  - ⇒ come viene costruito un UV-map per una data mesh?
    («how do we **parametrize**» or «how do we **uv-map**» a mesh)

207

Una tessitura (texture)



(di solito, sono potenze di due: 1024, 2048, 4096...)

208

Tessiture e texel

- ✓ Ogni pixel della tessitura, chiamato texel, memorizza valori qualsiasi che variano sulla superficie
  - ⇒ Tipicamente: colori, normali, valori di trasparenza, coefficienti di lucidità – o altri modi di descrivere un «materiale» …
- ✓ La tessitura è del tutto analoga ad un'immagine raster
  - ⇒ Cioè un array 2D (una «matrice») di «texel» (texture elements)
- ✓ Come tale, è caratterizzata da:
  - ⇒ Una risoluzione orizzontale *Rx* (di solito, potenza di due ≤ $2^{13} = 8192$)
  - ⇒ Una risoluzione verticale *Ry* (idem)
  - ⇒ Numero di canali per texel *Ch* (tipicamente, 1,2 o 4)
  - ⇒ Numero di bit per canale *B* (8 (spesso), 16, 24 …)
  - ⇒ Spazio totale occupato in memoria (espressa in *bit*) = $\overbrace{Rx \cdot Ry}^{\text{numero di texel}} \cdot \overbrace{Ch \cdot B}^{\text{«bit-depth»}}$
- ✓ In quanto immagine, può essere, per es, «dipinta» da un artista, catturata da foto, photoshopped, etc
  - ⇒ Esistono anche molte tecniche per sintetizzarle automaticamente
  - ⇒ Vedremo fra poco uno dei metodi più utilizzati

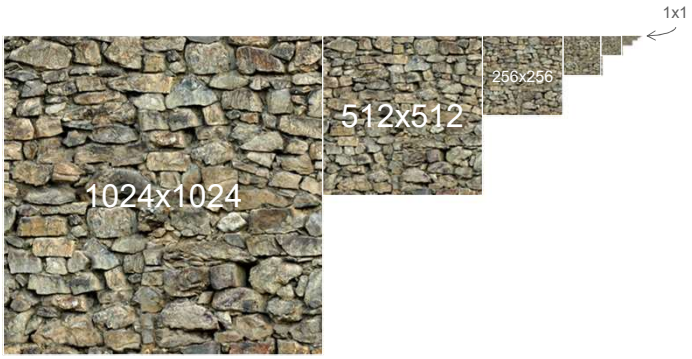
209

Tessiture e MIPmap level

- ✓ In quanto immagine raster, una tessitura si presta a semplici strutture multirisoluzione:
  - ⇒ Basta ridurre la risoluzione dell'immagine per ottenere un «livello» diverso di una piramide di livello di dettaglio
  - ⇒ Come si riduce la risoluzioni dell'immagine? Vedere corso di image processing (hint: interpolando fra i pixel / i texel)
- ✓ Struttura chiamata «**MIP-map**»:
  Piramide di livelli di dettaglio per tessiture:
  - ⇒ Livello 0: la tessitura originale, risoluzione massima
  - ⇒ Livello 1: una tessitura ridotta ad ½ della risoluzione in entrambe le dim
  - ⇒ Livello 2: riduzione del livello 1 (come sopra)
  - ⇒ …
  - ⇒ Livello N: tessitura 1x1 (per tessiture quadrate)
  - ⇒ Nota: in tutto sono $\log_2(RES)$ livelli
  - ⇒ Nota: in tutto, lo spazio tutale occupato da tutti i livelli oltre lo 0 è solo 1/4 + 1/16 +1/64 + 1/256 + … → 1/3 quello del livello 0

210

Texture: multirisoluzione



Piramide di livelli di dettaglio
detta MIP-map

211

Tessiture e supporto Hardware

- ✓ Le tessiture sono estremamente utilizzate nella maggior parte dei contesti applicativi in cui si usino mesh poligonali
  - ⇒ Come: games, VR, movies, commercio elettronico, applicazioni architetturali, sci vis, beni culturali…
    (ma spesso non: CAD, applicaz medicali)
- ✓ Parte della loro diffusione è dovuta al supporto HW:
  - ⇒ le GPU (il processore delle schede video) sono progettate per renderizzare mesh triangolari con tessitura associata
  - ⇒ Livello di MIP-map compresi!
  - ⇒ I limiti e le caratteristiche tipiche di una tessitura (num di canali ≤ 4, risoluzione massima, bit depth, risoluzione come potenza di 2) sono imposti da questo

212

Texture = immagine rasterizzata? Le differenze

- ✓ Una texture 2D è del tutto analoga ad un'immagine rasterizzata.
  Perché usare un termine separato per indentificare questo tipo di immagine?
- ✓ Diversi in…
  - ⇒ **uso atteso:** una texture è pensata esplicitamente per essere rimappata su una mesh (attraverso un UV-map)
  - ⇒ **contenuto:** i texel di una texture campionano qualsiasi segnale sulla superficie (normale, colore, proprietà del materiale (per es, coefficiente di lucidità…) etc), laddove i pixel di un'immagine sono campioni di *colore*, di solito rappresentati da una tripla di valori R,G,B
- ✓ Esistono formati file specifici per texture (come dds),
  diversi da quelli per immagini generiche (come png o jpeg), in termini di…
  - ⇒ **numero di canali:** una texture standard ha 4 canali (a volte: 1 o 2), un'immagine tipicamente ne ha 3
  - ⇒ **risoluzione:** per motivi storici inerenti al rendering, una texture ha risoluzione per lato come potenza di 2 (es: 512, 1024, 2048, 4096)
  - ⇒ **profondità di texel:** ogni canale di una texture ha un numero di bit per canale variabile da 4 a 16, molti formati per immagine prevedono solo 8bit per canale
  - ⇒ **compressione:** una tessitura deve essere random-accessible anche da compressa. Esistono *schemi di compressione per texture*, diversi dagli schemi usati per le immagini (che non hanno bisogno di essere random-accessible)
  - ⇒ **multi-risoluzione:** una tessitura spesso comprende un tipo di piramide di livelli di dettaglio detta «MIP-map»: ogni livello successivo dimezza la risoluzione del precendete (es: 1024x1024, 512x512, 256x256, … , 2x2, 1x1)
- ✓ Un formato per immagine generica (come png) può essere comunque usato come formato di interscambio per texture

213

Mesh con UVmap

- ✓ **Le posizioni UV sono memorizzate come attributo per vertice**
  - ⇒ Come ogni attributo, lo si considera interpolato dentro alle facce della mesh (attraverso le coordinate baricentriche)

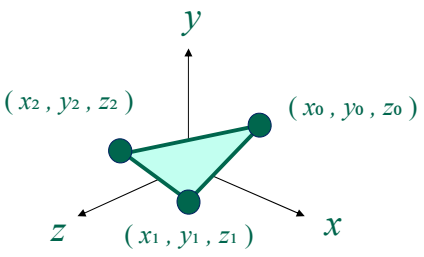
	Geometria	Attributi
$V_0 \rightarrow$	x_0 , y_0 , z_0	u_0 , v_0
$V_1 \rightarrow$	x_1 , y_1 , z_1	u_1 , v_1
$V_2 \rightarrow$	x_2 , y_2 , z_2	u_2 , v_2
$V_3 \rightarrow$	x_3 , y_3 , z_3	u_3 , v_3
$V_4 \rightarrow$	x_4 , y_4 , z_4	u_4 , v_4
$V_5 \rightarrow$	x_5 , y_5 , z_5	u_5 , v_5

lo *uv*-map!

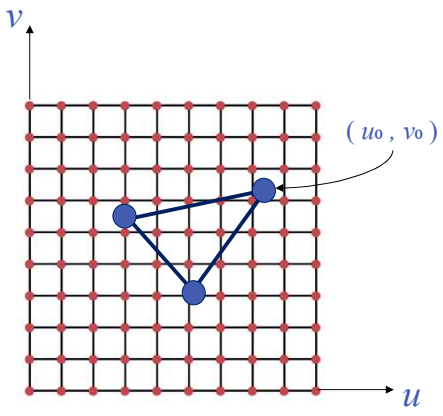
- ✓ Conseguenze:
  - ⇒ Ogni triangolo della mesh corrisponde a due triangoli: il triangolo T_3 nello spazio in 3D «x,y,z», il triangolo T_2 nello spazio 2D «u,v», dentro al rettangolo della texture
  - ⇒ Ogni punto dentro al triangolo in T_3 è mappato nel punto *con le stesse coordinate baricentriche* nel triangolo T_2 (come tutti gli altri attributi per vertice, «u,v» è interpolato nelle facce)
  - ⇒ L'intera mesh è embedded sul «dominio parametrico» 2D)

214

Mesh con UVmap



Mesh (3D)
spazio x,y,z

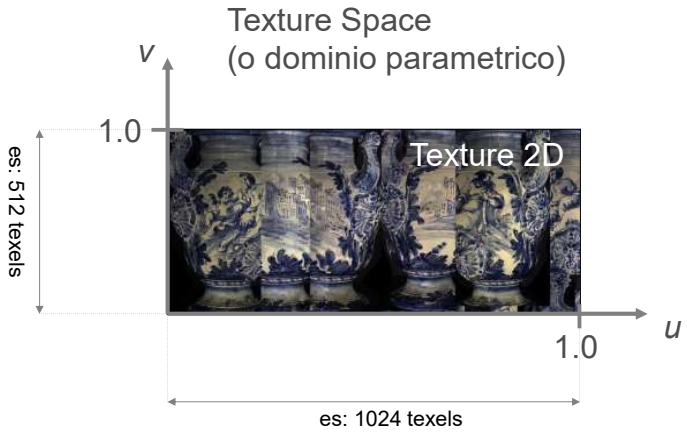


Tessitura (2D)
spazio u,v

215

Le coordinate texture sono “normalizzate” fra 0 e 1

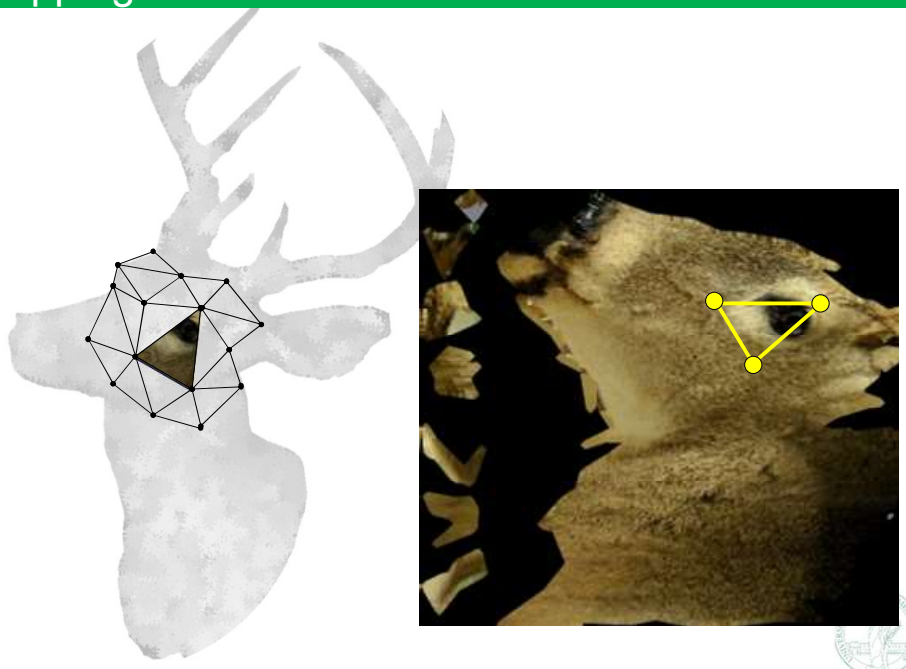
La texture è definita
nella regione [0,1] x [0,1]



Movito: lo UV-map memorizzato su una mesh è definito indipendentemente dalla *risoluzione* della texture

216

Mapping the texture over the mesh



217

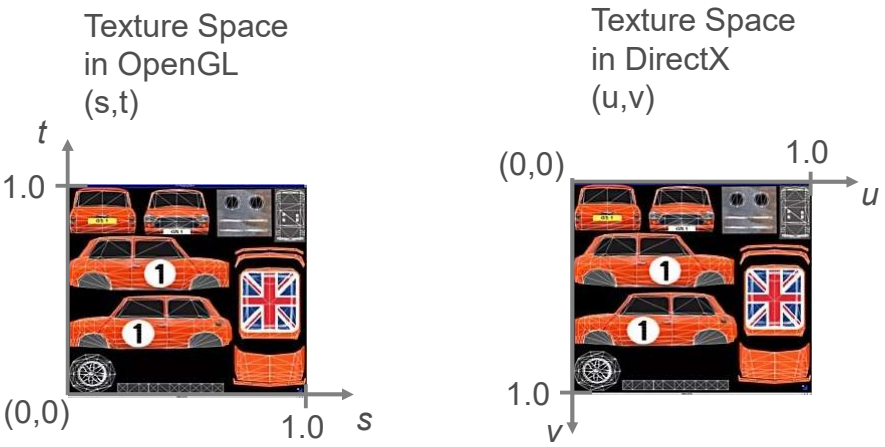
Mesh + Texture

218

Rimappare la tessitura sul dominio parametrico

219

Due notazioni



220

Mesh + Texture

221

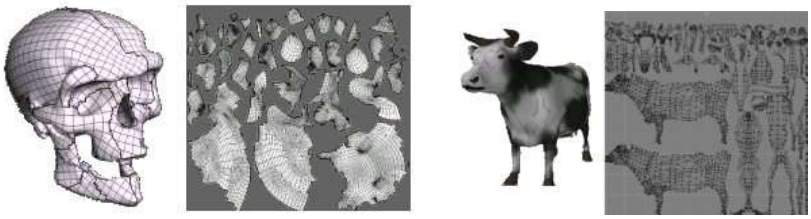
Costruire un UV-map per una mesh

- ✓ **UV-mapping di una mesh**, o **mesh parametrization**: il task di creare texture seams e assegnare coordinate UV ad ogni vertice
  - ⇒ A volte, (raro) la mesh viene creata congiuntamente al suo UV map
  - ⇒ Più spesso, la mesh creata (es. acquisita, o modellata, o estratta da un altro tipo di dato 3D) sprovvista di un UV-map
- ✓ Task non semplice
  - ⇒ Automatizzazione: tema molto studiato nel Geometry Processing
  - ⇒ Artisti digitali intervengono manualmente con software di 3D modelling (es Blender, Maya…)
- ✓ Un «buon» UV map deve rispondere a molti criteri:
  - ⇒ Iniettività: ogni punto della tessitura può essere mappato in al più un punto della mesh (cioè: nessun triangolo è sovrapposto in spazio UV)
  - ⇒ Bassa distorsione: ogni triangolo di mesh T_3 deve essere mappato in un triangolo di tessitura T_2 di *forma simile* e di *area proporzionale*
  - ⇒ Buon coverage della tessitura: le parti di tessitura non coperte da nessun triangolo rappresentano uno spreco di memoria e vanno minimizzate

222

Geometry processing: mesh parametrization

- ✓ Task: costruire automaticamente un UV-map per una mesh data
  - ⇒ problema difficile da risolvere in modo soddisfacente (soprattutto la scelta sul posizionamento/numero dei tagli)
  - ⇒ un gran numero di approcci algoritmici diversi
  - ⇒ rimane problema aperto: gli UV-map prodotti automaticamente sono spesso di scarsa qualità



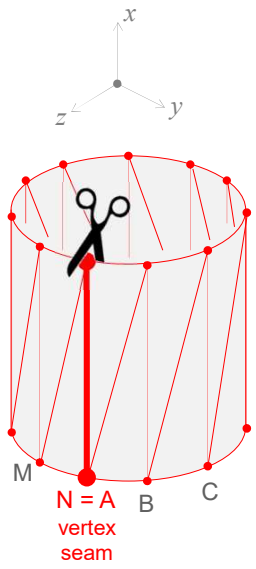
223

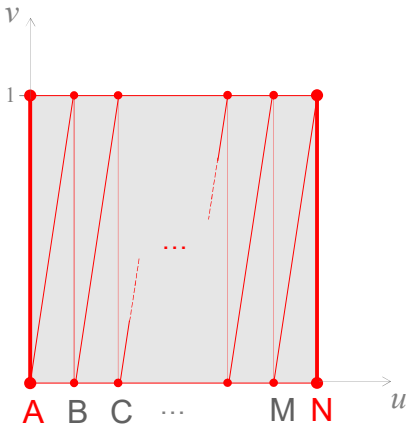
Texture seams (o cut)

- ✓ Nozione intuitiva:
  - ⇒ il task di mesh parametrization ricorda quello di sbucciare una mela e disporre la buccia dentro un rettangolo
  - ⇒ Simile: come costruire un planisfero, cioè la mappa (2D) della superficie del pianeta (sfera in 3D) (che è un problema storicamente molto studiato: es. vedi: http://vcg.isti.cnr.it/~tarini/spinnableworldmaps/)
  - ⇒ La differenza è che la mesh ha una forma e una topologia arbitraria, piuttosto che sferica
- ✓ Come questa intuizione suggerisce, un UV-map richiede di introdurre dei tagli
  - ⇒ detti «texture seams» o cuts
  - ⇒ questi tagli consentono di aprire la mesh per stenderla (unwrap) su un piano

224

Gli UV mapping richiedono tagli («texture seam»)





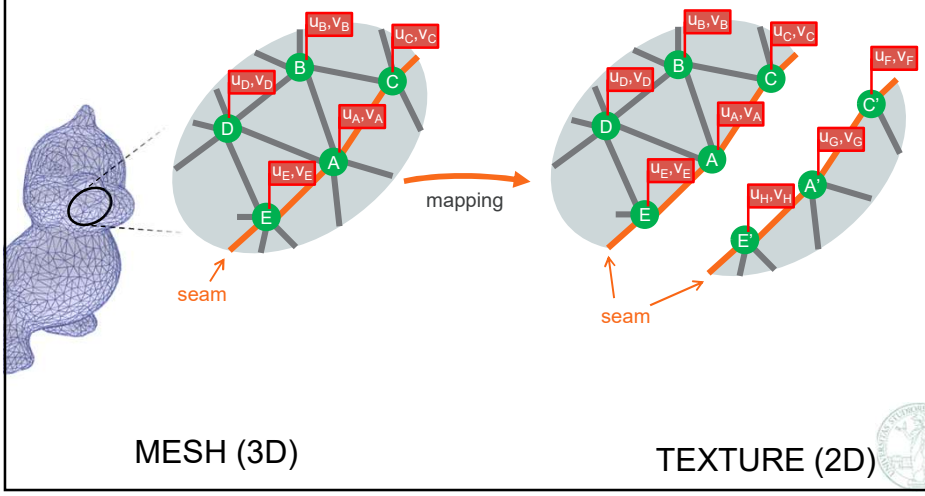
MESH TEXTURE

225

Rappresentazione interna dei vertex seams

✓ E' un altro esempio di *discontinuità di attributo*

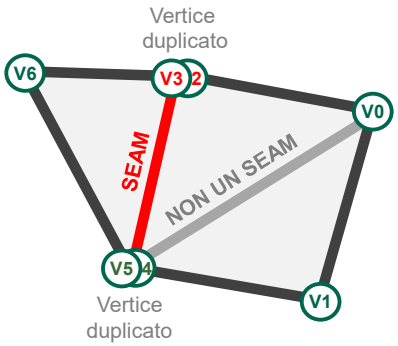
⇒ Dunque, rappresentabile attraverso duplicazione di vertici



226

Texture seams (o anche texture "cuts")

✓ I seams sono scomodi ma necessari!



	X	Y	Z	U	V	…
V0	p_x0	p_y0	p_z0	$u0$	$v0$	…
V1	p_x1	p_y1	p_z1	$u1$	$v1$	…
V2	p_x2	p_y2	p_z2	$u2$	$v2$	…
V3	p_x2	p_y2	p_z2	$u3$	$v3$	…
V4	p_x3	p_y3	p_z3	$u4$	$v4$	…
V5	p_x3	p_y3	p_z3	$u5$	$v5$	…
V6	p_x4	p_y4	p_z4	$u6$	$v6$	…

GEOMETRIA + ATTRIBUTI

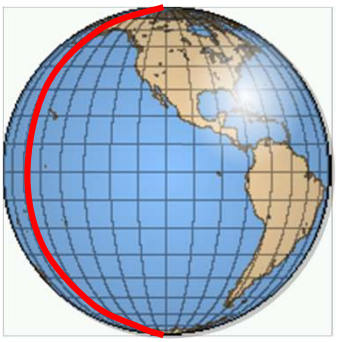
Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T0	0	1	4
T1	4	2	0
T2	5	3	6

CONNETTIVITA'

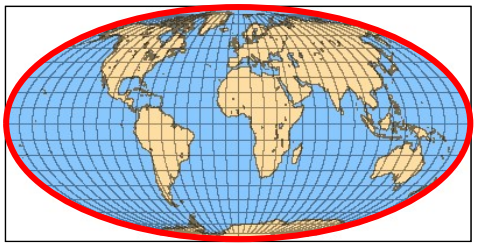
227

Tagli (cuts) o texture “seams”

✓Necessari se la mesh è chiusa



two-manifold chiuso
in 3D

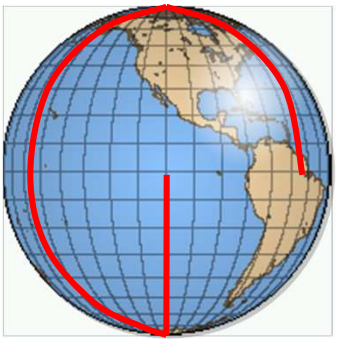


spazio parametrico 2D

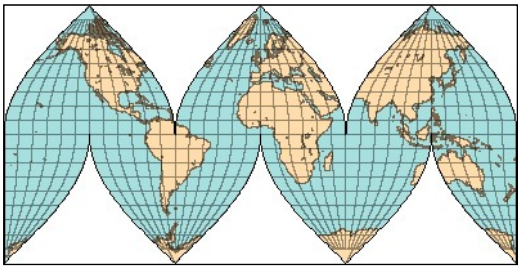
228

Tagli (cuts) o texture “seams”

✓Più tagli ⇒ meno distorsione (di solito)



two-manifold chiuso
in 3D

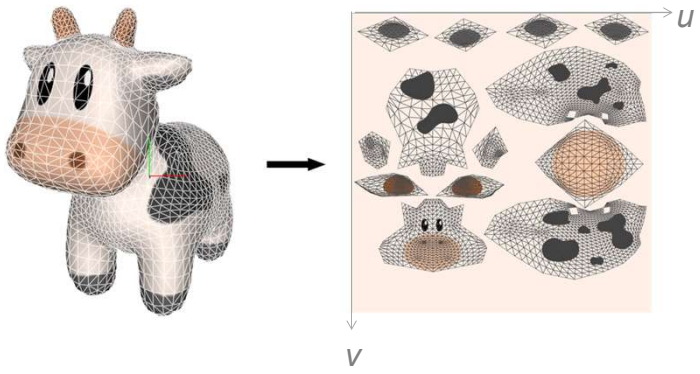


spazio parametrico 2D

229

Texture Atlas

- Approccio in cui mesh viene divisa in zone (patches), ciascuna parametrizzata in un “isola” di tessitura.
- Lo UV-map di questo tipo si dice «texture atlas» (analogia con un atlante di mappe)



230

Esempio di mapping generato automaticamente

231

Esempio di mapping generato automaticamente

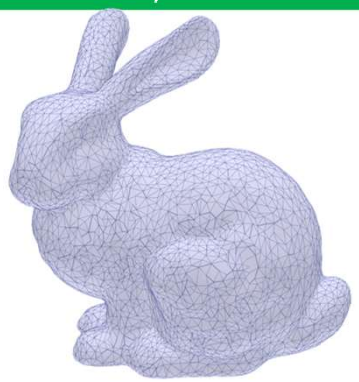
232

Una normale per Texel: le Normal maps

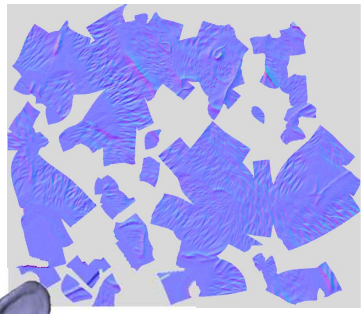
- ✓ Una normal-map è una texture che campiona le normali della superficie
  - ⇒ in questo modo, contribuisce a specificare la forma geometrica dell'oggetto
  - ⇒ (introducendo però una approssimazione)
  - ⇒ tipicamente, la normal map riproduce dettagli geometrici minuti, ad alta frequenza (come gli avvallamenti su una buccia di arancia), mentre la mesh riproduce la forma generale dell'oggetto (come la forma sferica dell'arancia)
- ✓ Uno stesso oggetto può essere rappresentato, con simile accuratezza e una resa simile, attraverso …
  - ⇒ una mesh ad alta risoluzione, oppure
  - ⇒ una mesh a bassa risoluzione ma provvista tessitura per dettagli «ad alta frequenza» (come una normal-map e/o una tessitura colore)
- ✓ Quali vantaggi e svantaggi?

233

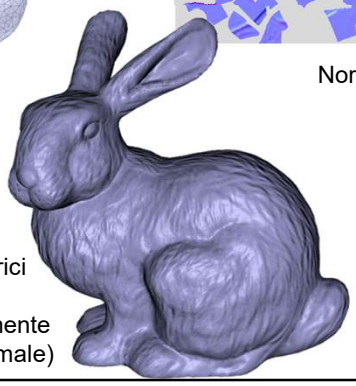
Normal map: tessitura che codifica una normale in ogni texel



Mesh Low res
(10K triangoli)



Normal Map



Risultato:
molti dettagli geometrici
sulla superficie
(modellati escluisvamente
attraverso la loro normale)

234

Textured Mesh VS Hi-resolution meshes

Hi-res mesh (con attributi)

- ✓ densità di campionamento segnale = densità di campionamento geometria
- ✓ incrementare il numero di vertici (cioè la risoluzione della mesh) ha un costo elevato: coinvolge anche la connettività, richiede di processare più triangoli
- ✓ non richiede *vertex seams* (se il segnale è continuo)
- ✓ la forma della superficie è modellata esplicitamente, fin nei suoi dettagli più minuti (questo permette di tenerne conto in tutti i tipi di processing)

Low-res mesh con texture

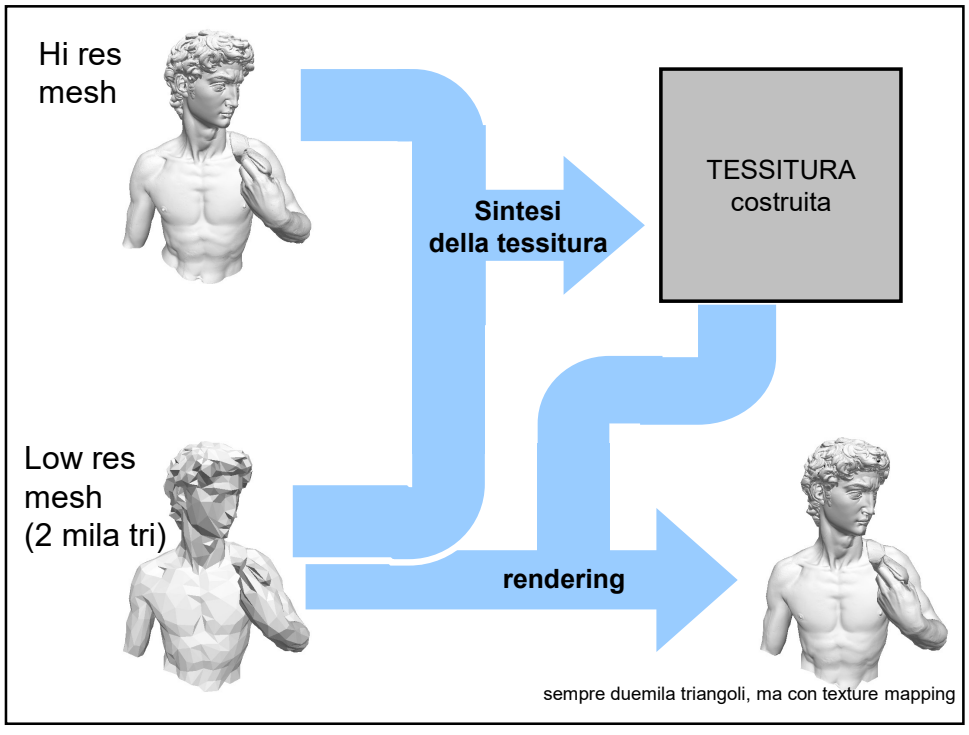
- ✓ risoluzione texture (colore, etc) indipendete da risoluzione mesh (geometria)
- ✓ incrementare il numero di texel (cioè la risoluzione della tessitura) ha un costo piccolo
- ✓ costo extra: UV-map sulla mesh. Va costruito (difficile); va mantenuto in memoria; richiede l'introduzione di vertex seams.
- ✓ ma, in totale, meno onerosa in termini di spazio (memoria), e tempo (di rendering)
- ✓ normal-map: i dettagli di forma sono riprodotti solo visualmente attraverso il lighting

235

Texture «baking» (traduzione letterale: cotta al forno)

- ✓ Sintesi automatica di una tessitura, per riprodurre un attributo definito su una mesh hi-res
- ✓ Input:
  - ⇒ Mesh Hi-res M0, con attributi (per vertice, per faccia…) che rappresentano un segnale che ci interessa riprodurre nella tessitura (colore, normale, materiale…)
  - ⇒ Mesh Low-res M1 che approssima la stessa forma, dotata di UV-map (cioè «parametrizzata») ma non di tessitura
- ✓ Output
  - ⇒ una tessitura sintetizzata (baked texture) per M1 che riproduce su di essa l'attributo definito su M0
- ✓ Un passaggio comune nel pipeline di creazione dei modelli 3D
  - ⇒ Supportato per es da tutte le suite di modellazione 3D

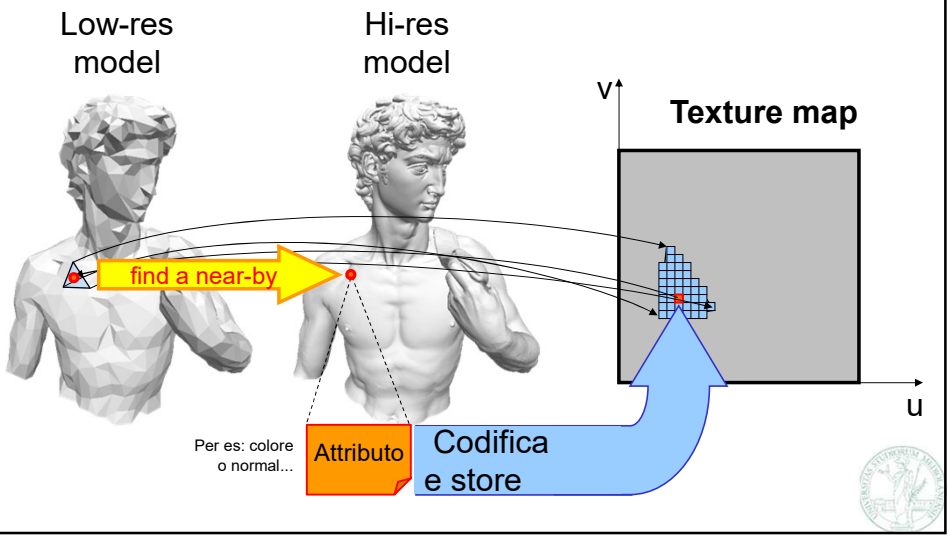
236



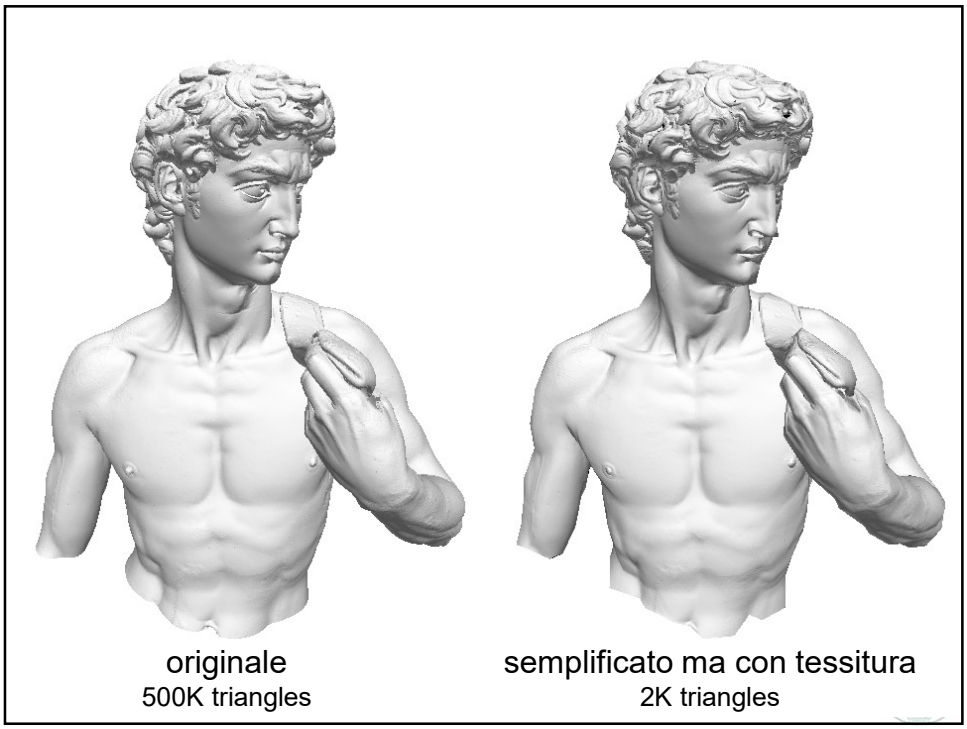
237

Sintesi della tessitura: schema dell'algoritmo

L'attributo definito per vertice nel modello Hi-Res (interpolato per triangolo) viene memorizzato sui texel della tessitura,
secondo l'UV-mapping della mesh Low res.



238



239