Marco Tarini - Computer Graphics 2025/2026
Università degli Studi di Milano

Mesh Poligonali: Textures & UV mapping 1/2

228

Tessiture: intro

- ✓ Attributi come…
  - ⇒ Colore (colore base: caratteristica del «materiale»)
  - ⇒ Normale
  - ⇒ Altre caratteristiche del «materiale» come luncentezza, etc (vedi seconda metà del corso)
  - ⇒ O qualsiasi altro dato numerico che varia sulla superficie
- ✓ …possono essere memorizzati
  - ⇒ Per vertice (ed interpolati sulle facce)
  - ⇒ Per faccia (e costanti per faccia)
  - ⇒ Oppure, **per Texel**, cioè su una griglia 2D di campioni detta **tessitura** - l'argomento di questa lezione -

Questo 3zo caso, che consente di memorizzare attributi con una densità di campionamento molto maggiore

231

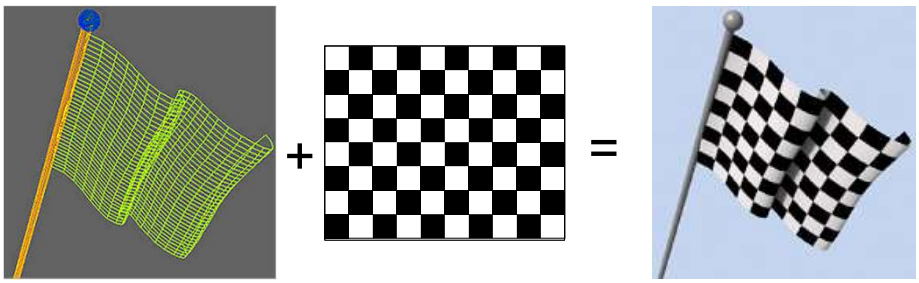
Textures

- ✓ 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?

232

Mesh + Texture



Mesh
(pure quad mesh)

Texture

(griglia regolare
di "texel" 2D)

233

Mesh + Texture

Tri-Mesh

+

Texture
(in questo caso:
un patchwork di foto)

=

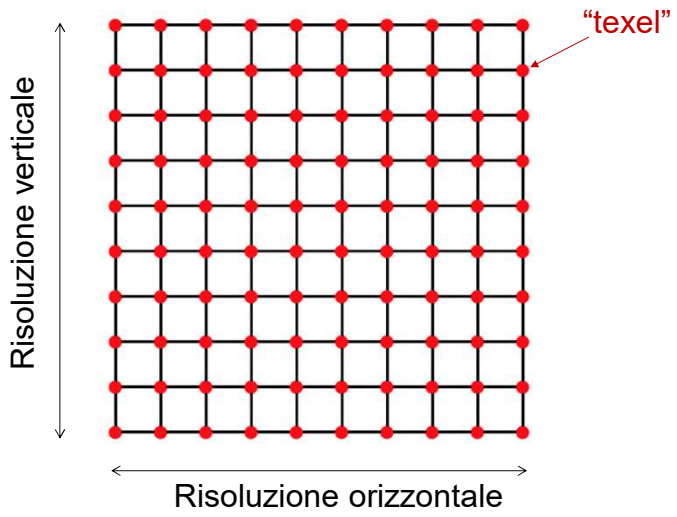
234

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)

235

Una tessitura (texture)



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

236

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

237

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)

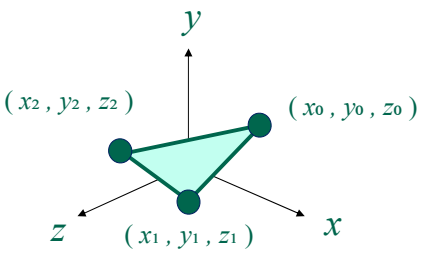
	Geometria	Attributi
$V_0 \rightarrow$	x_0 , y_0 , z_0	u_0 , v_0
$V_1 \rightarrow$	x_1 , y_1 , z_1	u_1 , v_1
$V_2 \rightarrow$	x_2 , y_2 , z_2	u_2 , v_2
$V_3 \rightarrow$	x_3 , y_3 , z_3	u_3 , v_3
$V_4 \rightarrow$	x_4 , y_4 , z_4	u_4 , v_4
$V_5 \rightarrow$	x_5 , y_5 , z_5	u_5 , v_5

lo *uv*-map!

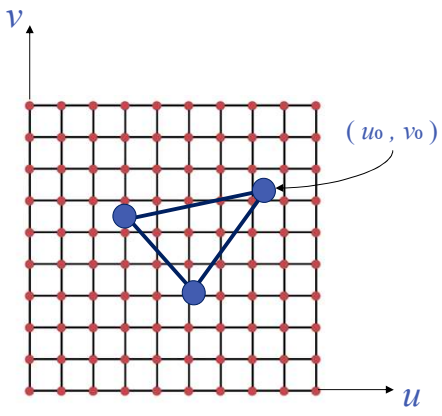
- ✓ Conseguenze:
  - ⇒ Ogni triangolo della mesh corrisponde a due triangoli: il triangolo T_3 nello spazio in 3D «x,y,z», il triangolo T_2 nello spazio 2D «u,v», dentro al rettangolo della texture
  - ⇒ Ogni punto dentro al triangolo in T_3 è mappato nel punto *con le stesse coordinate baricentriche* 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)

238

Mesh con UVmap



Mesh (3D)
spazio x,y,z

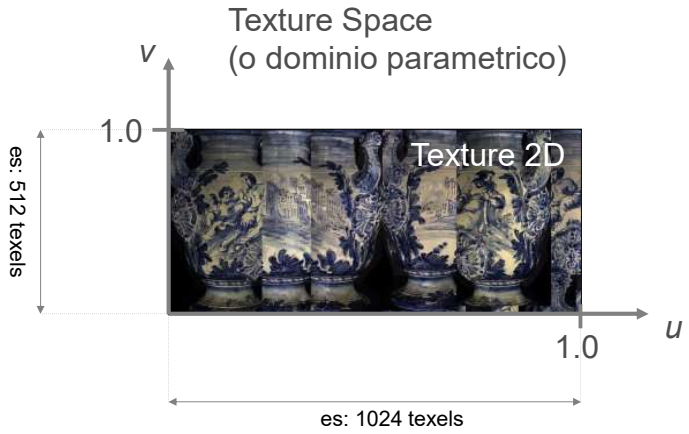


Tessitura (2D)
spazio u,v

239

Le coordinate texture sono “normalizzate” fra 0 e 1

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

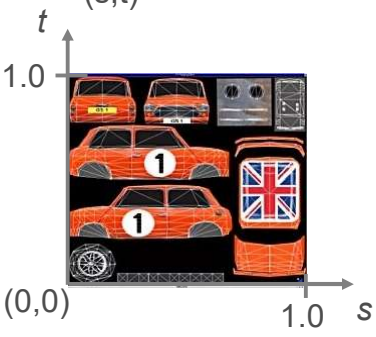


Movito: lo UV-map memorizzato su una mesh è definito indipendentemente dalla *risoluzione* della texture

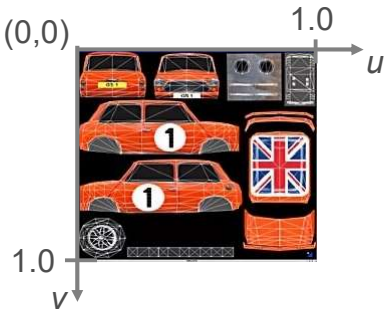
240

Due notazioni molto usate (purtroppo diverse)

Texture Space
in OpenGL
(s,t)

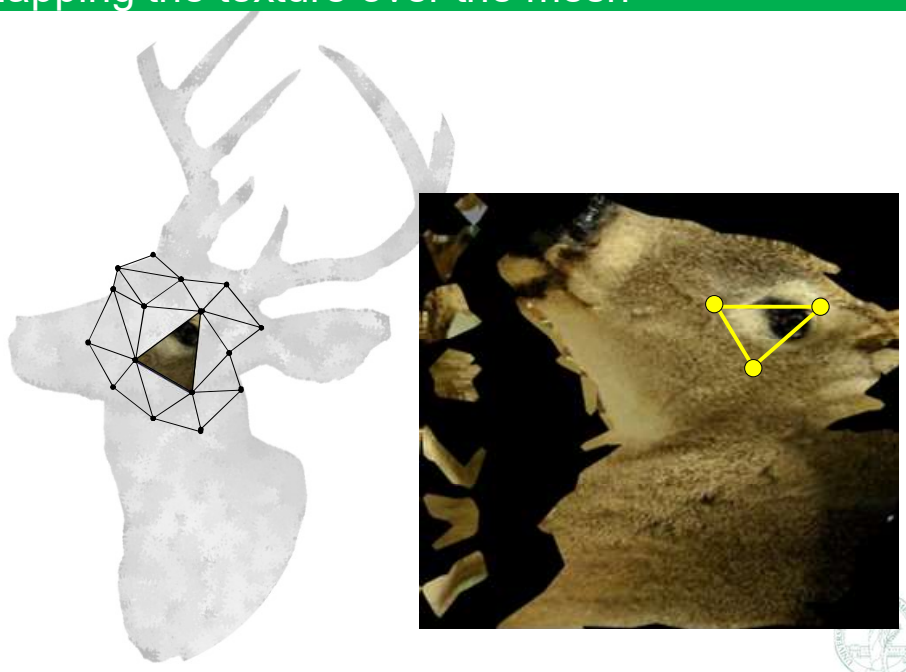


Texture Space
in DirectX
(u,v)



241

Mapping the texture over the mesh



242

Rimappare la tessitura sul dominio parametrico

243

Mesh (con UV map) + Texture: il dataset più comune




245

Tessiture e supporto Hardware

- ✓ Le tessiture sono estremamente utilizzate nella maggior parte dei contesti applicativi in cui si usino mesh poligonali
  - ⇒ Come: videogames, 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 grafico) sono progettate proprio per renderizzare mesh triangolari con tessitura associata
  - ⇒ I limiti e le caratteristiche tipiche di una tessitura (num di canali ≤ 4, risoluzione massima, bit depth, risoluzione come potenza di 2) sono imposti dalle caratteristiche dell'HW

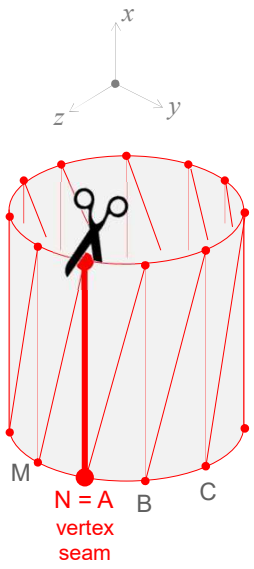
246

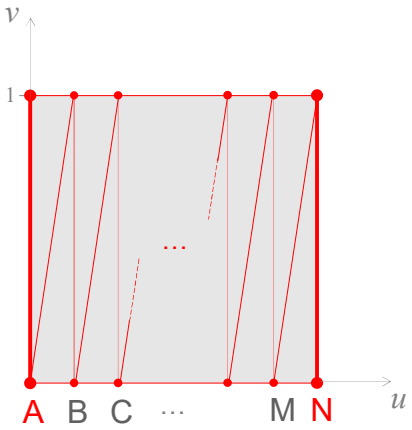
Mesh parametrization (task di geometry processing)

- ✓ **UV-mapping di una mesh**, o **mesh parametrization**: il task di creare texture seams e assegnare coordinate UV ad ogni vertice
  - ⇒ Di solito le mesh è 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
  - ⇒ Limitato numero di *texture seams*

247

Gli UV mapping richiedono tagli («texture seam»)





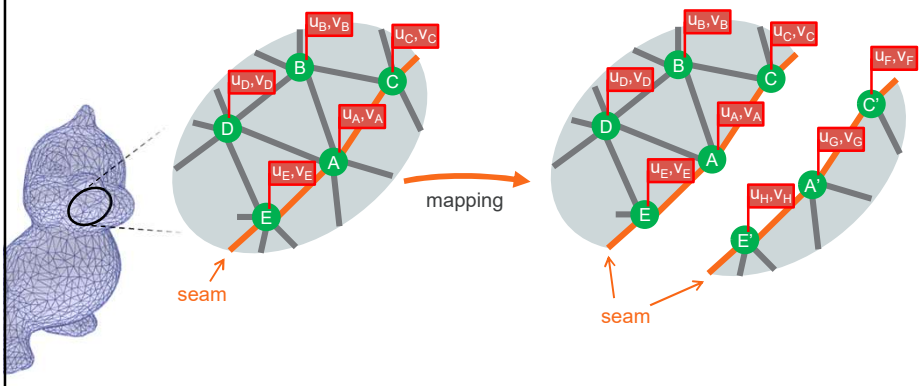
MESH TEXTURE

248

Rappresentazione dei texture seams (o texture cut)

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

⇒ Dunque, rappresentabile attraverso duplicazione di vertici



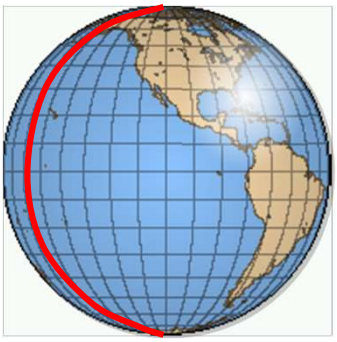
MESH (3D)

TEXTURE (2D)

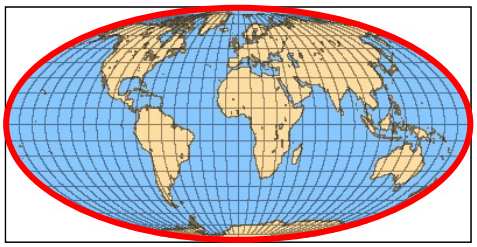
249

Tagli (cuts) o texture "seams"

✓ Sempre necessari se la mesh è chiusa



two-manifold chiuso in 3D

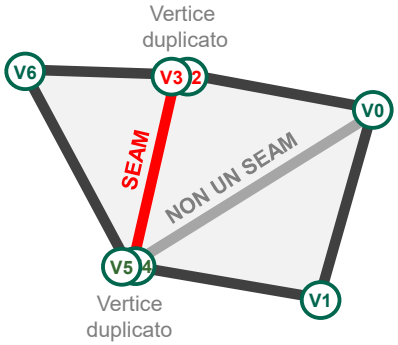


spazio parametrico 2D

250

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

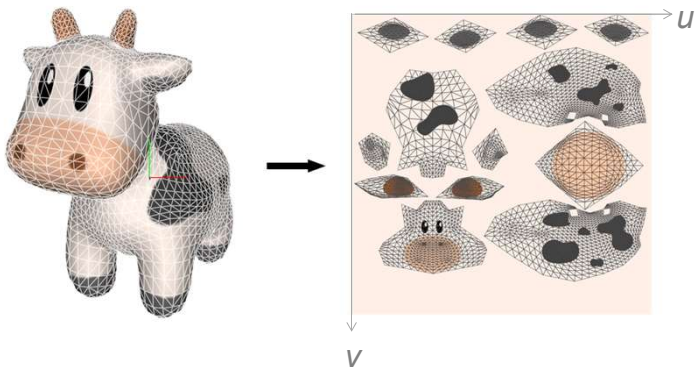
Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T0	0	1	4
T1	4	2	0
T2	5	3	6

CONNETTIVITA’

251

Texture Atlas

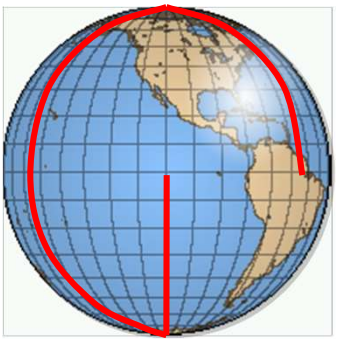
- Tipo di UV map in cui mesh viene divisa in zone (patches), ciascuna rimappata in un “isola” della tessiura.
- (il nome proviene dall’analogia con un atlante di mappe)



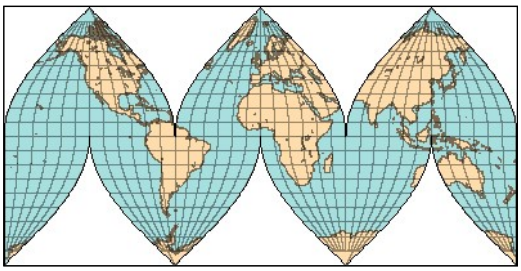
252

Tagli (cuts) o texture “seams”

✓Più tagli ⇒ meno distorsione (di solito)



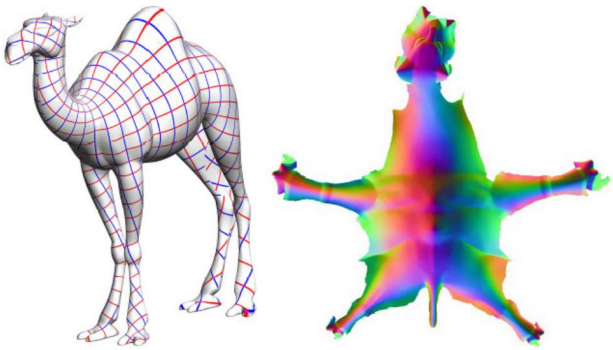
two-manifold chiuso in 3D



spazio parametrico 2D

253

Mesh parametrization (task di geometry processing)



Img by Sébastien Loriot, Olga Sorkine-Hornung

254

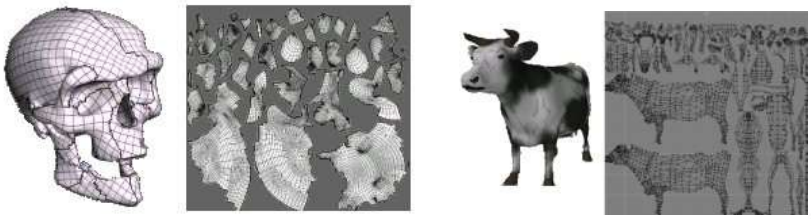
Mesh parametrization (task di geometry processing)

- ✓ Il task:
  - ⇒ Data: una mesh (two-manifold) di input
  - ⇒ Produrre: il suo UV-map (cioè un assegnamento delle texture coordinates u-v per ogni vertice)
- ✓ Richiede di risolvere sottoproblemi come:
  - ⇒ Dove introdurre i tagli (i texture seams, o texture cuts)
  - ⇒ Come stendere («unwrap») la superficie sul piano
  - ⇒ Nelle tessiture ad atlante (atlas): come comporre le varie isole nel rettangolo della tessitura («packing»)
- ✓ Task chiamato anche:
  - ⇒ **UV-mapping** di una mesh (specie in ambiente industriale – che sia fatto ad opera di un algoritmo automatico O di un artista digitale umano)
  - ⇒ Automatic Mesh **parametrization** (in ambiente accademico)

255

Mesh parametrization (task di geometry processing)

- ✓ 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 da artisti digitali sono spesso di qualità migliore



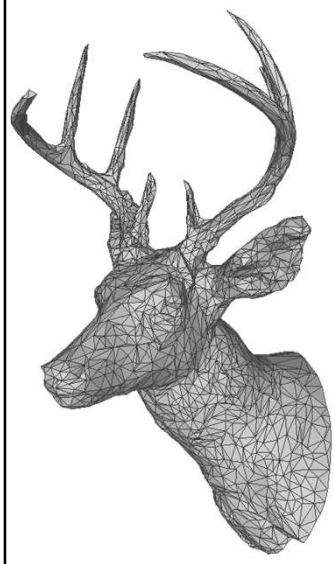
256

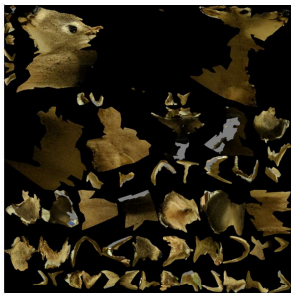
Texture seams (o cut)

- ✓ 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, defnire un UV-map richiede di introdurre i tagli adatti
  - ⇒ questi tagli consentono di aprire la mesh per stenderla (unwrap) su un piano

257

Esempio di mapping generato automaticamente





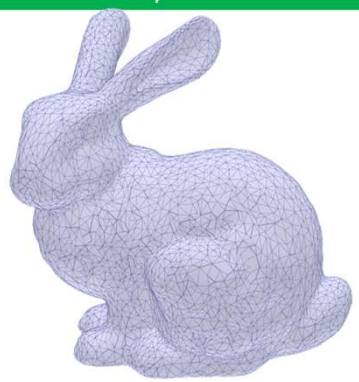
258

Normale memorizzate per Texel: le Normal maps

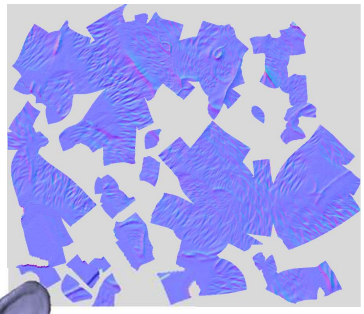
- ✓ Una **normal-map** è una **texture** che campiona in ogni suo **texel** la normale della superficie
  - ⇒ in questo modo, contribuisce a specificare la forma geometrica dell'oggetto a piccole scale
  - ⇒ (introducendo però una approssimazione)
  - ⇒ tipicamente, la normal map riproduce dettagli geometrici 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?

260

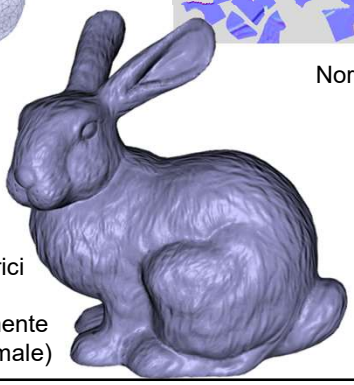
Normal map: tessitura che codifica una normale in ogni texel



Mesh Low res
(10K triangoli)



Normal Map



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

261

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

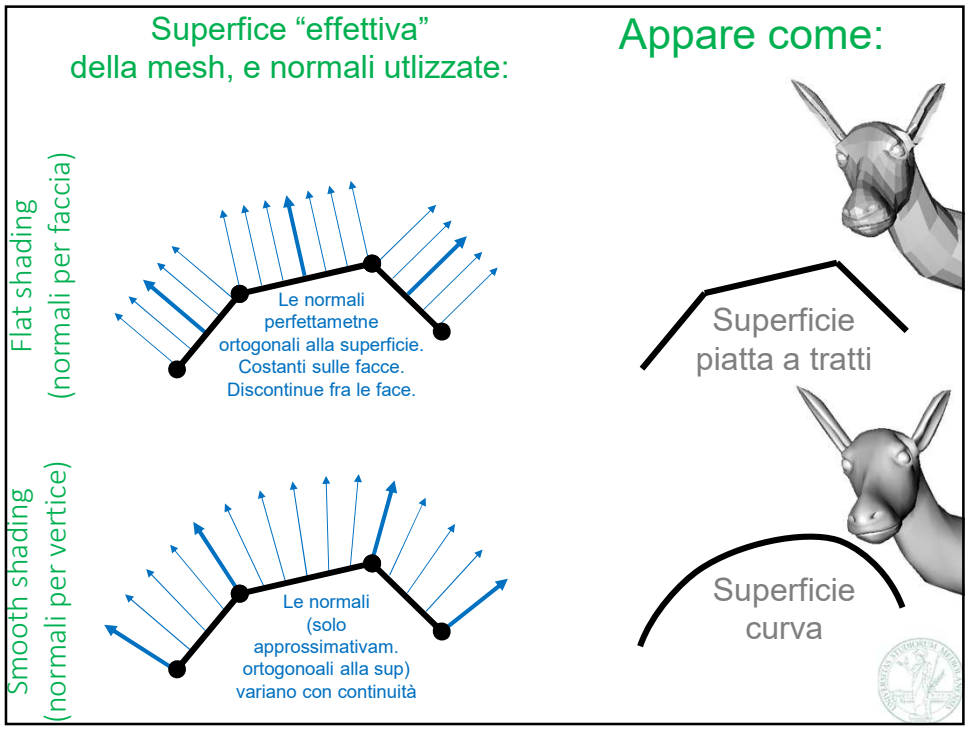
- ✓ risoluzione texture (colore, etc) indipendente da risoluzione mesh (geometria)
- ✓ incrementare il numero di texel (cioè la risoluzione della tessitura) ha un costo piccolo
- ✓ costo extra: UV-map sulla mesh. Va costruito (difficile); va mantenuto in memoria; richiede l'introduzione di vertex seams.
- ✓ 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

262

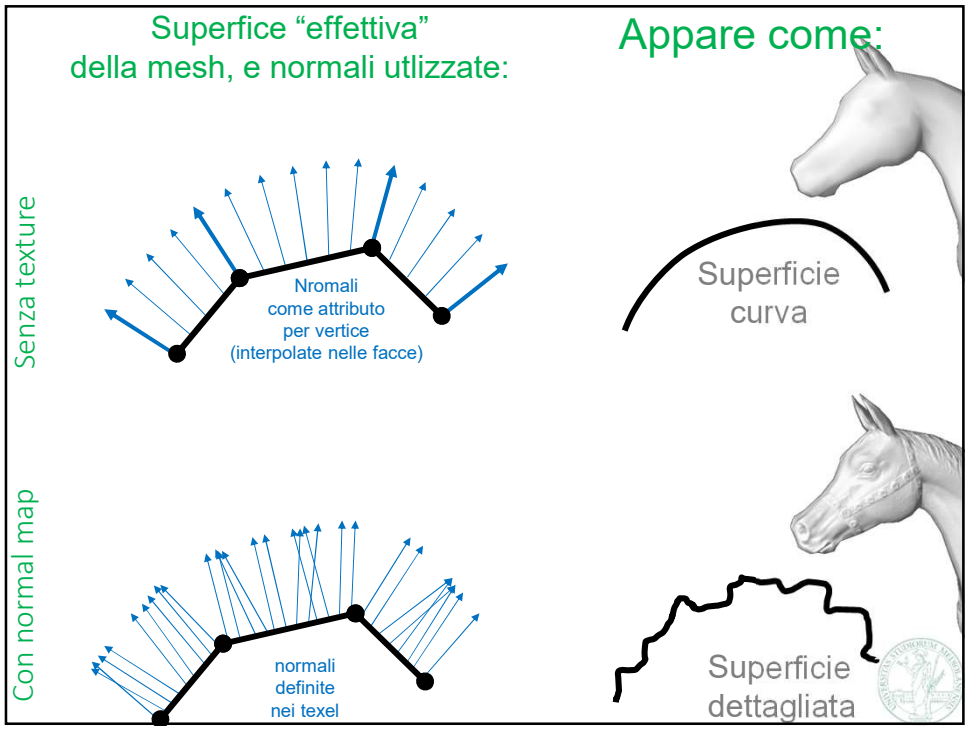
Osservazione: geometria e illuminazione

- ✓ Come ci eravamo già resi conto nel caso dello smooth shading, nel rendering l'aspetto percepito delle superfici è dominato dall'illuminazione (lighting, il chiaroscuro cioè), che è determinata dalle normali, maggiormente che non dall'effettiva forma geometrica rappresentata dai nostri modelli
- ✓ Ad esempio:
  - ⇒ Un oggetto renderizzato con flat-shading (cioè usando normali definite per faccia) rivela la sua reale natura **piatta**, dato che le normali sono costanti, ma…
  - ⇒ visualizzato con smooth shading (cioè normali definite per vertice ed interpolate nelle facce) ci appare **curvo**, dato che le normali variano con continuità
  - ⇒ Questo ci consente di visualizzare superfici (che appaiono) curve anche usando mesh triangolari, che sono per definizione piatte «a tratti» (in ogni faccia)
- ✓ Anche nel caso nelle normal-map, il lighting è molto efficace nel suggerire una forma complessa codificata delle normali memorizzate nella tessitura
  - ⇒ Piuttosto che la forma effettiva, ben più povera di dettaglio!
  - ⇒ Ottenere lo stesso dettaglio geometrico attraverso un incremento della risoluzione della mesh sarebbe molto più oneroso (memoria, tempo di rendering)
- ✓ Si tratta, in entrambi i casi, di un'illusione dovuta al lighting
  - ⇒ rivelata, per es, dalla sagoma, o silhouette
  - ⇒ (vedremo nelle prossime lezioni strutture dati superficiali capaci di rappresentare superfici *effettivamente* curve!)

263


Superfice “effettiva”
della mesh, e normali utlizzate:
Appare come:
Flat shading
(normali per faccia)
Le normali
perfettametne
ortogonali alla superficie.
Costanti sulle facce.
Discontinue fra le face.
Superficie
piatta a tratti
Smooth shading
(normali per vertice)
Le normali
(solo
approssimativam.
ortogonoali alla sup)
variano con continuità
Superficie
curva

264


Superfice “effettiva”
della mesh, e normali utlizzate:
Appare come:
Senza texture
Nromali
come attributo
per vertice
(interpolate nelle facce)
Superficie
curva
Con normal map
normali
definite
nei texel
Superficie
dettagliata

265