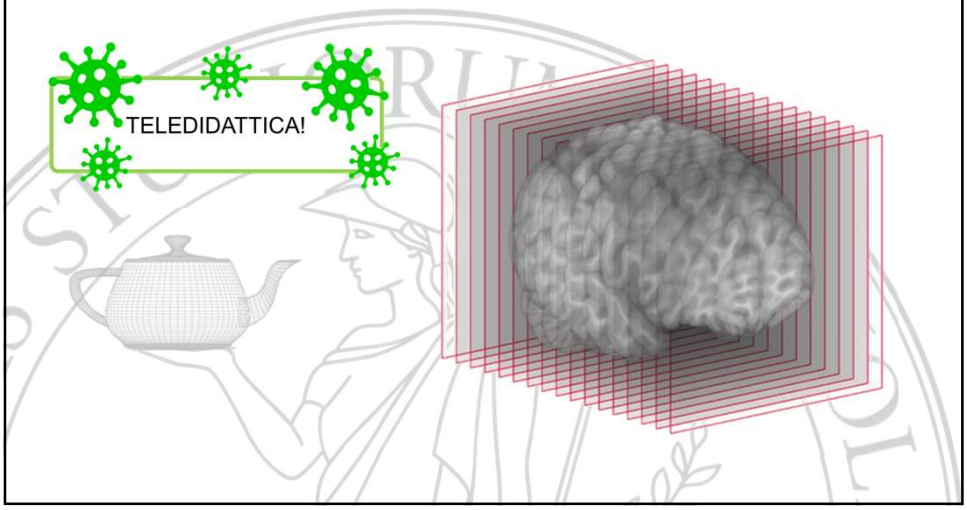
Marco Tarini - Computer Graphics 2019/2020
Università degli Studi di Milano

Rappresentazioni a Voxel



2

Modelli 3D Volumetrici

1. Discreti & regolari: dataset voxellizati
   ⇒analogo di un immagine rasterizzata, ma in 3D
   ⇒una griglia di voxel

2. Discreti & irregolari: mesh poliedrali
   ⇒Tetra-mesh, hexa-mesh
   ⇒insieme di poliedri adiacenti faccia a faccia

3. Continui: modelli impliciti
   ⇒rappresentazione basata su funzioni volumetriche
   ⇒superficie: luogo di zeri di questa funzione

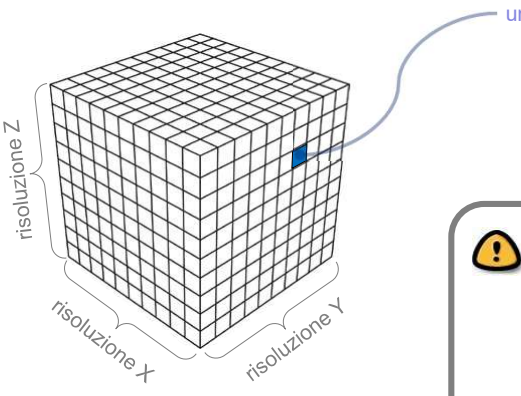
3

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

		ELEMENTI DISCRETI			CONTINUI
		regolari *«a griglia»*	semi-regolari o irregolari		
			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 surfaces** **Parametric Surfaces** **(es. B-splines)**
	non-manifold *«non rappresenta una sup»*	**Set di Range Scan**	**Point Cloud**		
VOLUMETRICI	(3-manifold)	**Voxelized Volume** **Volumetric Textures**	**Tetra Mesh**	**Hexa Mesh**	**Implicit models** **(es. CSG)**

4

Modello 3D a voxel (o voxellizato)



Griglia regolare 3D
(o lattice)

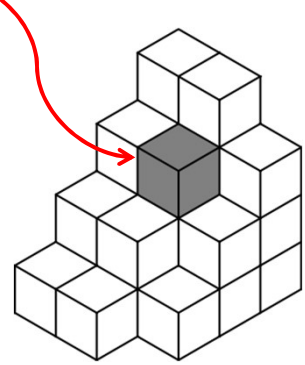
consumo memoria *cubico* con la risoluzione
(diventa facilmente ingestibile)

```
array [RES_X] [RES_Y] [RES_Z] of Voxels
```

5

Modelli Voxellizati

- ✓"Voxel" = Volume element
  - ⇒Così come…
    "Pixel" = Picture Element
    "Texel" = Texture Element
- ✓Elemento di una griglia regolare 3D
  - ⇒che è anche detta un lattice



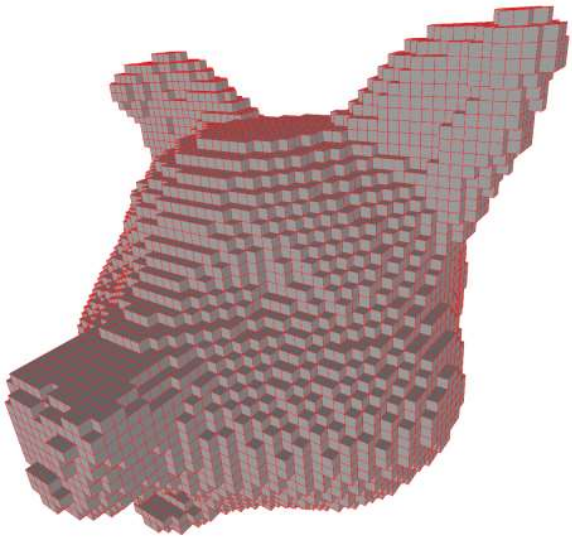
- ✓In codice:

```
Voxel[][][] data = new Voxel[resX][resY][resZ];
```

Esempio in Java

6

In questo caso, 1 Voxel = 1 Boolean (1 bit)

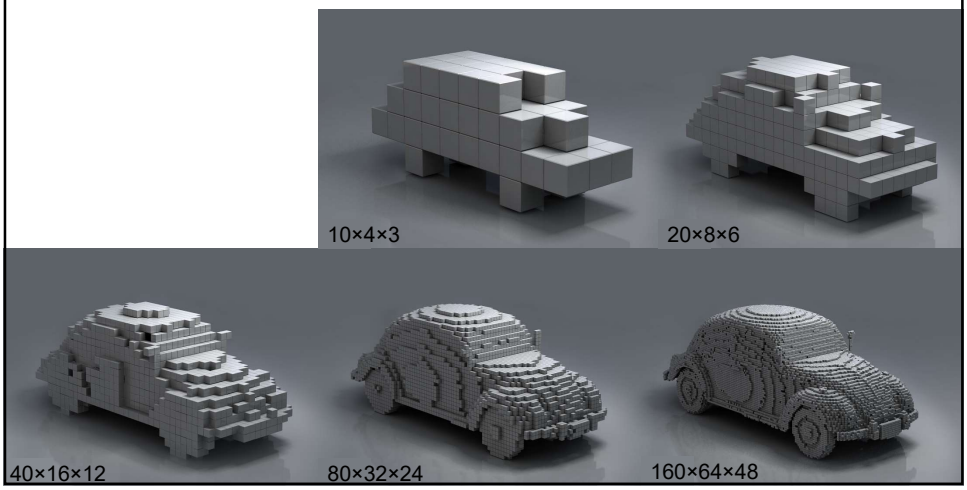


ogni *voxel* è pieno (1) o vuoto (0)

7

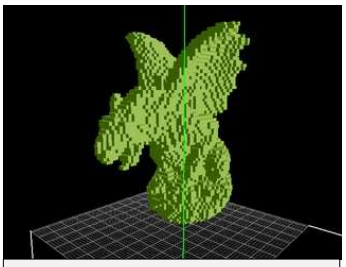
Risoluzione e costo in memoria

✓risoluzione: un intero per lato res = (X,Y,Z)



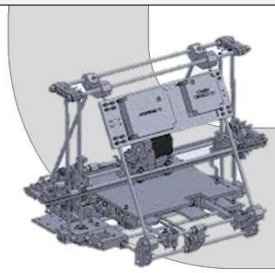
8

Voxel = 1 bit (pieno / vuoto)

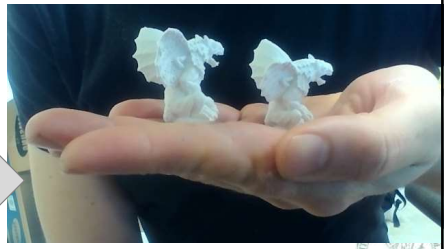


Voxelized dataset
Voxel: pieno/vuoto

✓Input naturale di (alcuni)
3D printing devices
⇒Rapid prototyping devices per stampa “additiva”



3D printer

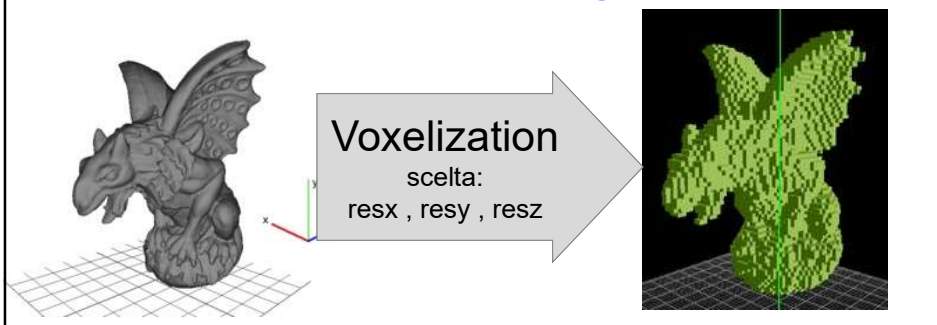


oggetti stampati

9

Voxelized Volumes:

- ✓Input naturale dei **3D printing devices**



Tri Mesh

ben orientata
e chiusa!

Voxelized volume

– 1bit × voxel

to printer

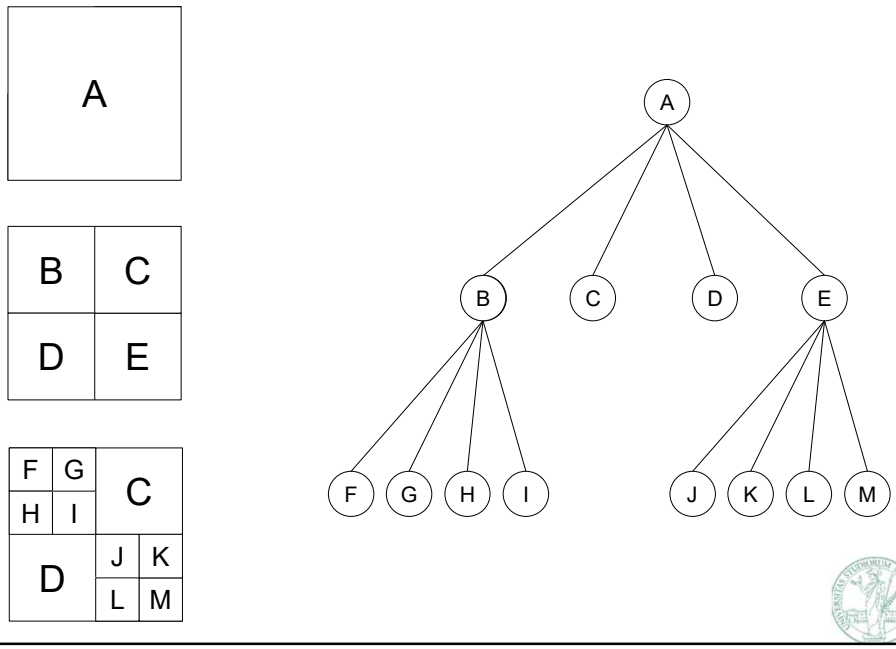
10

Occupazione spaziale dei dataset voxelizzati

- ✓La «curse of dimensionality» colpisce:
- ✓Lo spazio cubico con la risoluzione (linare) è di solito un prezzo troppo alto
  - ⇒Es: 1024^3 voxel = 1 gigavoxel
  - ⇒Persino nel caso, come abbiamo visto fin'ora, di 1 solo bit per voxel (pieno/vuoto)
- ✓Esistono molte strutture dati più compatte
- ✓Una delle più diffuse è l'oct-tree
  - ⇒Vediamo prima una sua versione 2D: il quad-tree

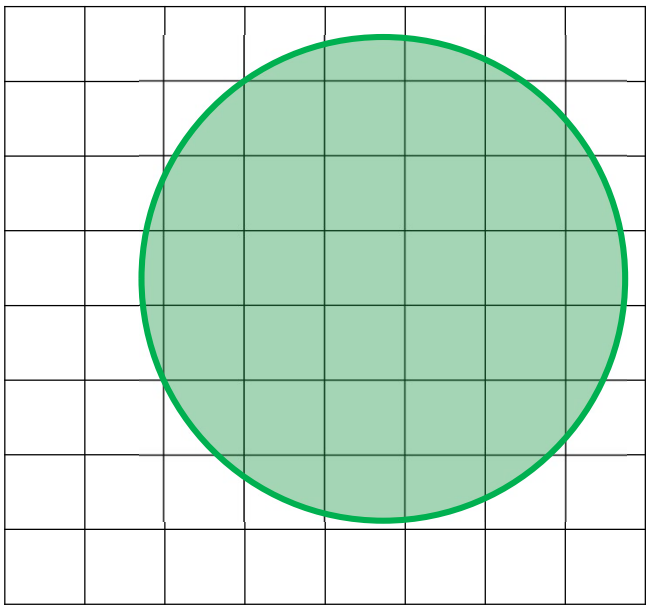
12

Quad Tree



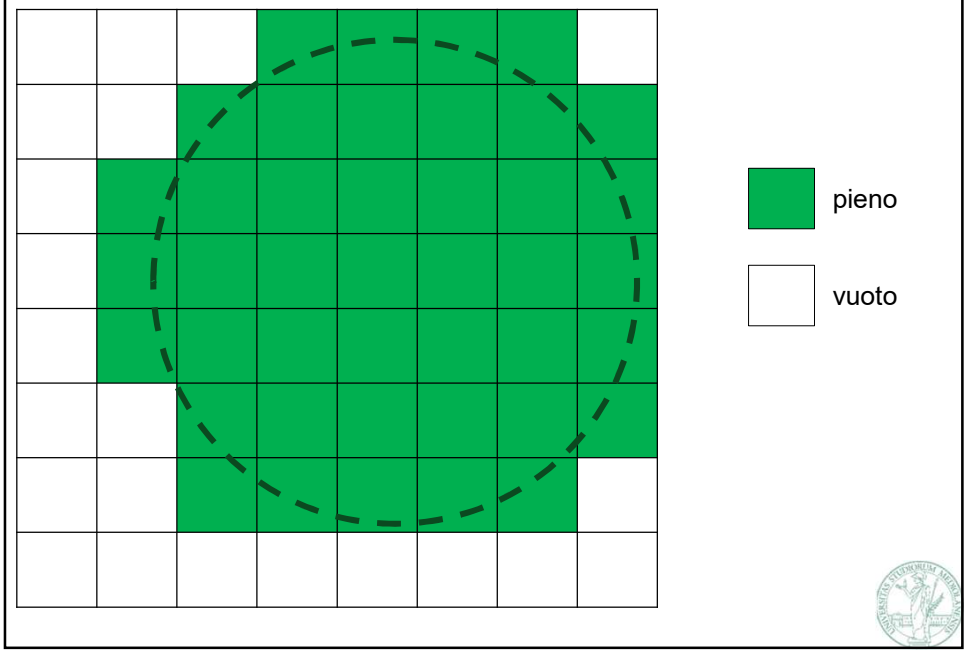
13

Esempio. 1/4: la forma da rappresentare



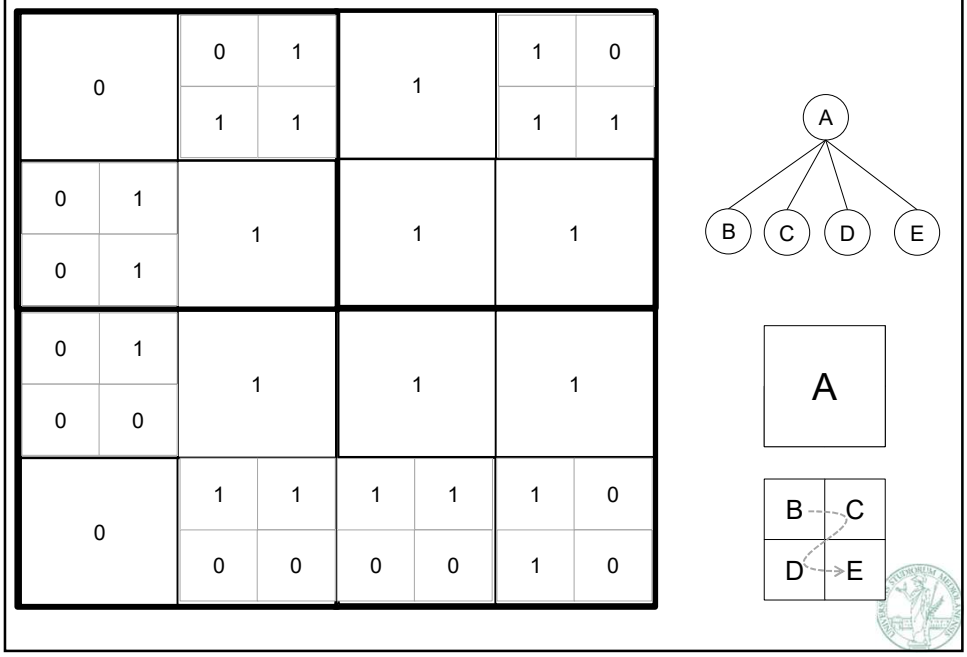
15

Esempio. 2/4: “voxellizzazione” (ma in 2D)



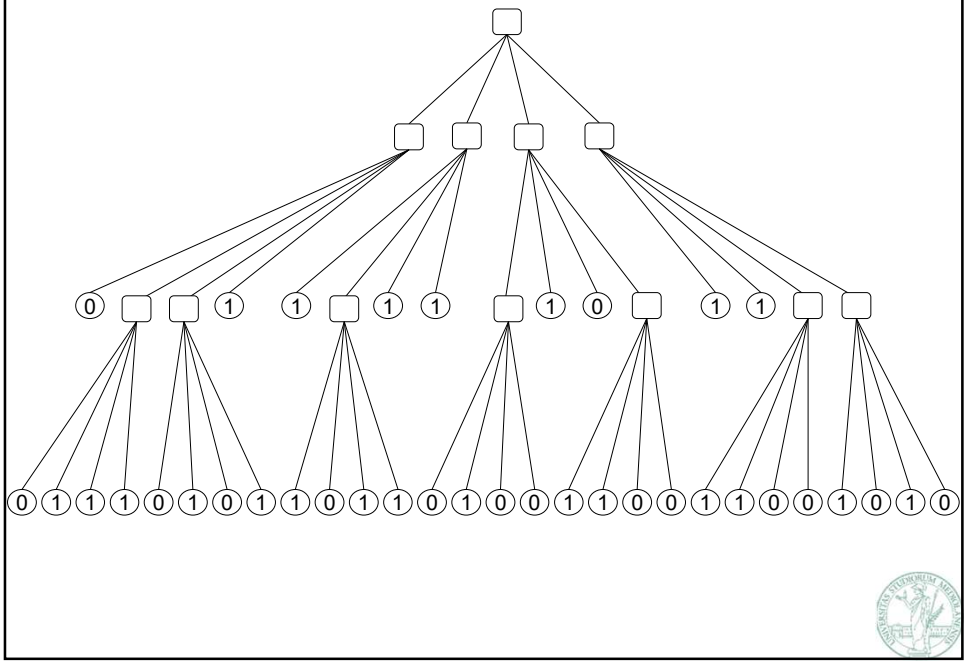
16

Esempio. 3/4: quad-tree (scomposizione)



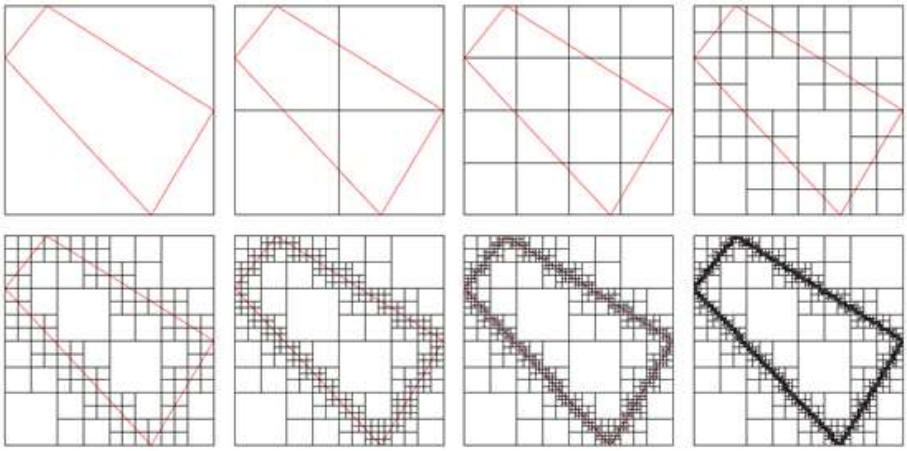
17

Esempio. 4/4: quad-tree (l'albero)



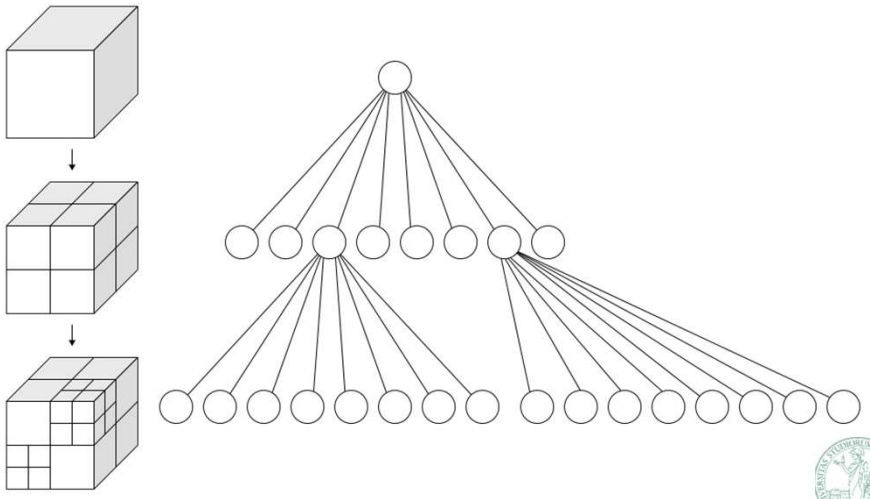
18

Quad tree: esempio



19

Oct Tree: stessa cosa, per 3D



20

Oct tree: sommario

- ✓ Struttura ricorsiva, ad albero
- ✓ Qui usata per memorizzare efficientemente un volume voxellizzato a 1 bit
  - ⇒ Ci sono molti altri usi in computer graphics
- ✓ Ogni nodo è associato ad un cobo del volume
  - ⇒ Radice: associato all'intero dataset
  - ⇒ Nodo interni: blocchi che contengono sia 1 che 0
    Hanno 8 figli, uno per ciascun ottavo del padre
  - ⇒ Foglie: rappresentano aree di tutti 1 o 0
- ✓ Numero totale nodi: circa
  proporzionale al bordo dell'oggetto rappresentato
  - ⇒ è quindi quadratico, non cubico, con la risoluzione!

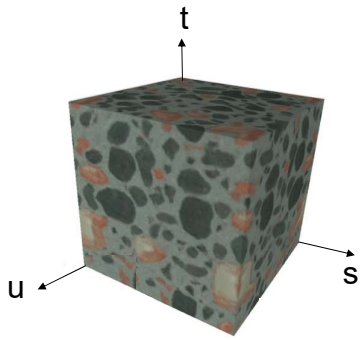
21

Quando 1 Voxel = un *indice* (al tipo di terreno)

Esatto, proprio i terreni di minecraft ™

23

Se Voxel = 1 colore RGB: “texture volumetriche”



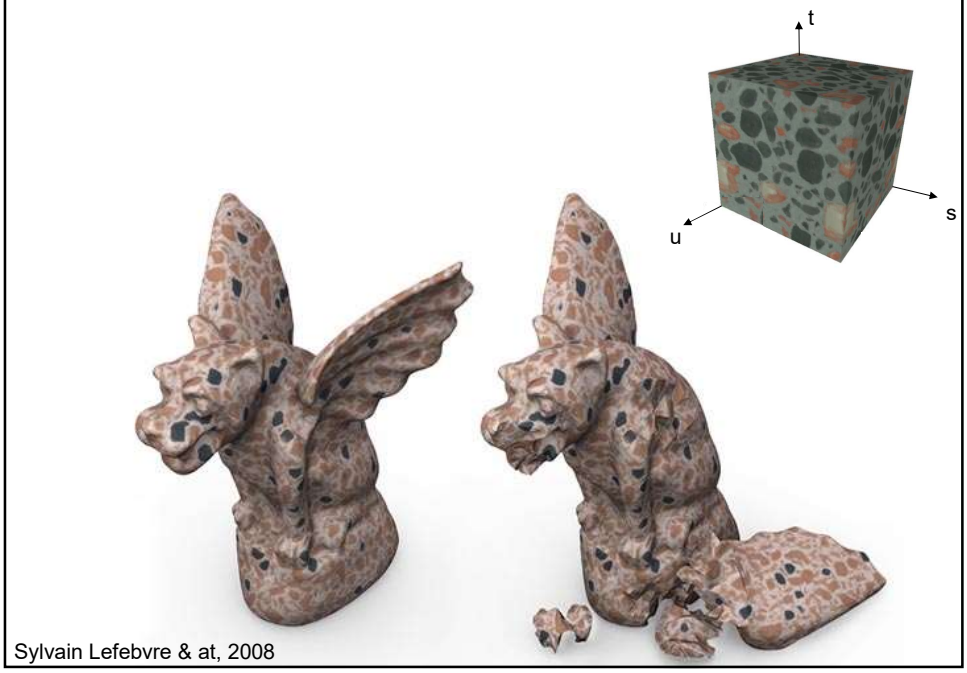
24

Volumetric Textures (o “solid Textures”)

- ✓ 1 texel = 1 voxel
  - ⇒esempio, solid RGB textures: 1 texel = 1 RGB color
- ✓ E’ supportata dall’Hardware, come ogni altra tessitura:
  - ⇒occupa la RAM della scheda video
  - ⇒accesso HW accelerato durante il rendering
  - ⇒interpolazione **tri**-lineare durante l’accesso …
- ✓ Modella il segnale (es. il colore) *dentro* al volume
  - ⇒per es: come gli oggetti sono colorati all’interno
  - ⇒utile per modelli che si possono rompere
  - ⇒utile per pattern come legno, marmo…
- ✓ Non richiede alcuna parametrizzazione della superficie!
  - ⇒La tessitura viene indicizzata dalle posizioni dei vertici
- ⚠ Solito problema, occupazione di memoria
  - ⇒es: quanto per 1 tessitura 1024^3 8-bits-per-channel RGBA?
  - ⇒es: quanto per 1 tessitura 265^3 8-bits-per-channel RGBA?

25

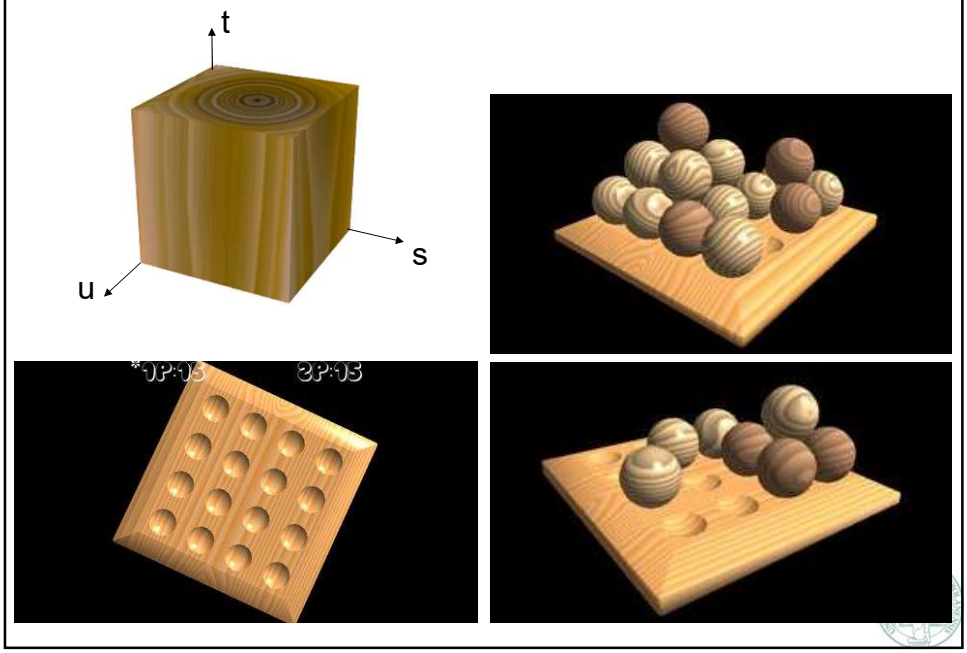
Un *colore* per voxel: “texture volumetriche”



Sylvain Lefebvre & at, 2008

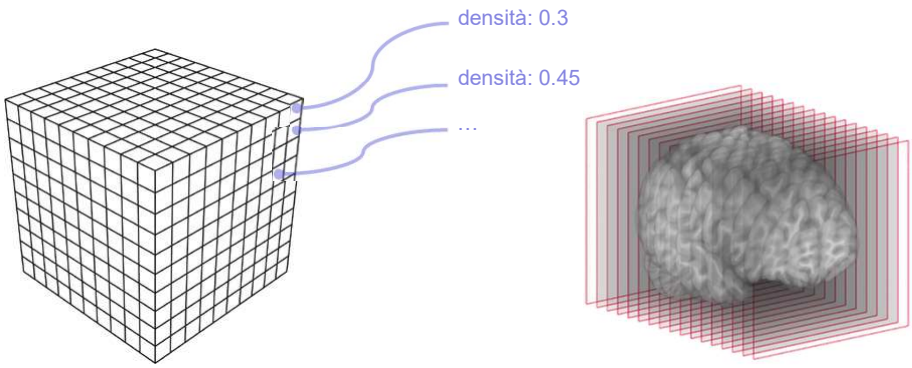
26

Volumetric Textures (o “solid Textures”)



27

Voxel = 1 scalare (es fra 0.0 e 1.0)



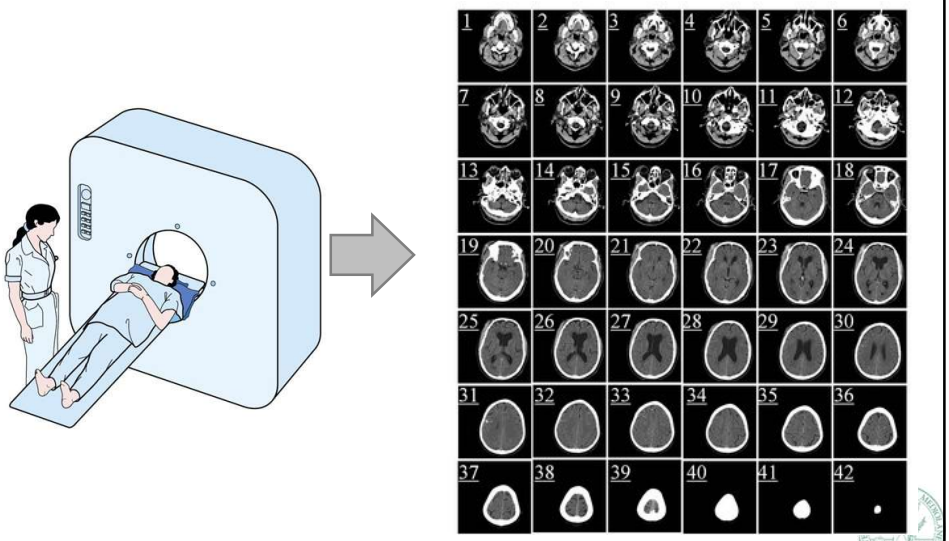
Esempio di File format: **DICOM**
(medicina)

```
Volume float [RES_X] [RES_Y] [RES_Z]
```

28

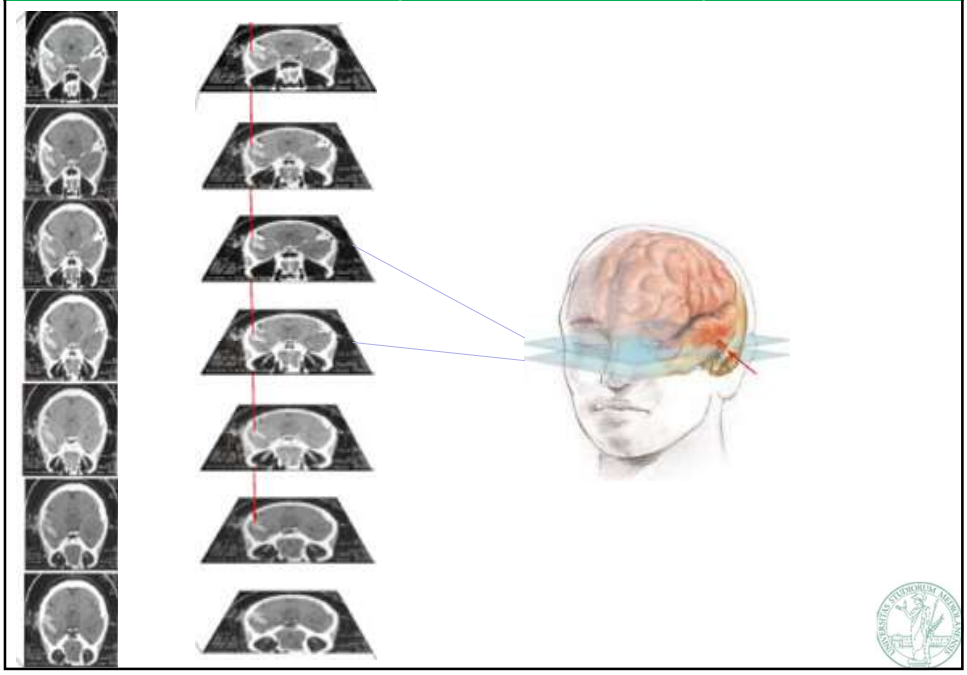
Se 1 voxel = 1 float (valori di densità)

✓Output naturale di **CT scans**



30

Se 1 voxel = 1 float (valori di densità)



32