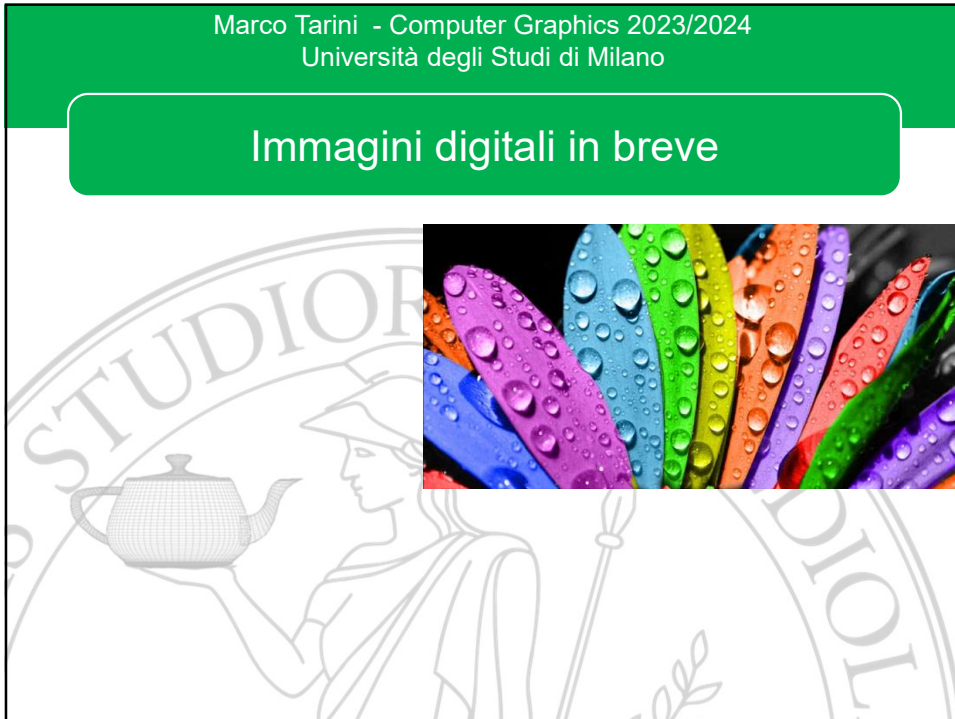
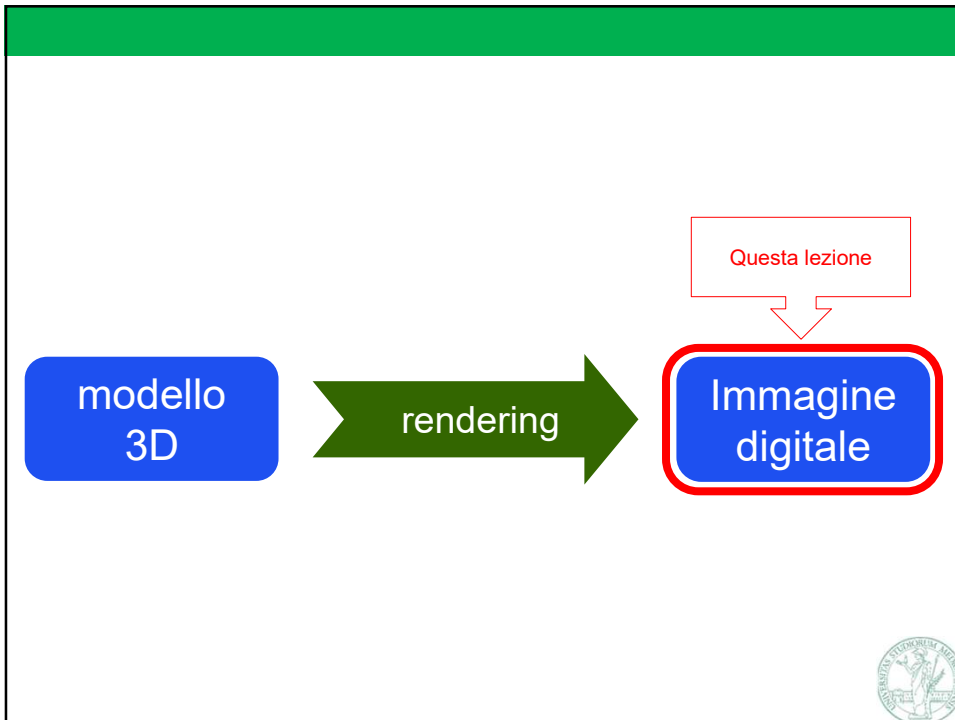


Marco Tarini - Computer Graphics 2023/2024
Università degli Studi di Milano

Immagini digitali in breve



1


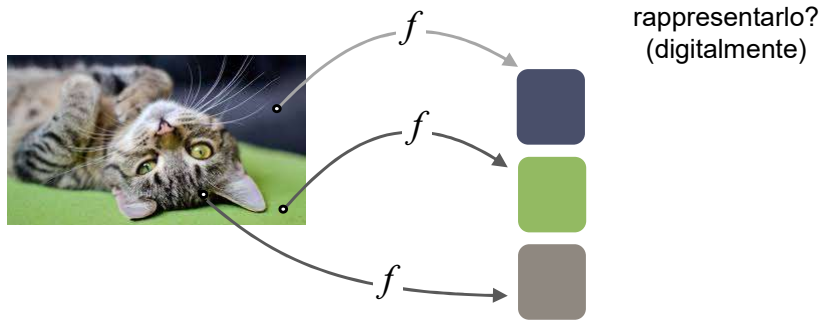


5

Immagine

Immagine = funzione f da (x,y) a (colore)


Come rappresentarlo? (digitalmente)



7

Immagini digitali

- ✓ **Rappresentazioni vettoriali:**
un set di primitive 2D sovrapposte, quali:
 - ⇒ curve parametriche
 - ⇒ triangoli , poligoni (2D), cerchi, ...
 - ⇒ zone contornate da curve di Bèzier (2D)
 - ⇒ testo (in ascii, ad una pos, associato ad un font)
 - ⇒ *etc.*
 - ⇒ ciascuna primitiva associata ad un colore
- ✓ **Rappresentazioni “rasterizzate”:**
una griglia regolare 2D di campioni di **colore** detti “pixel” (da “picture element”)



8

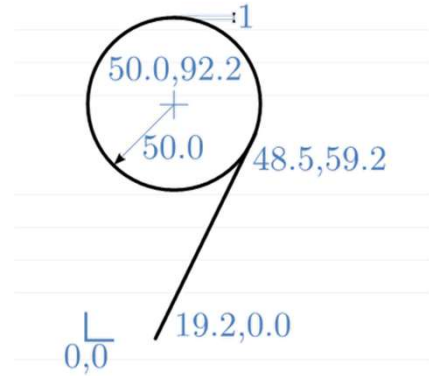
Immagine vettoriale: esempio (inventato)

```

NUMBER_OF_PRIMITIVES 2

CIRCLE
center 50.0, 92.2
radius 50.0
fill_color 1.0, 1.0, 1.0
line_color 0.0, 0.0, 0.0
line_thickness 1pt

SEGMENT
endpoint_one 19.2, 0.0
endpoint_two 48.5, 59.2
line_color 0.0, 0.0, 0.0
line_thickness 1pt
  
```

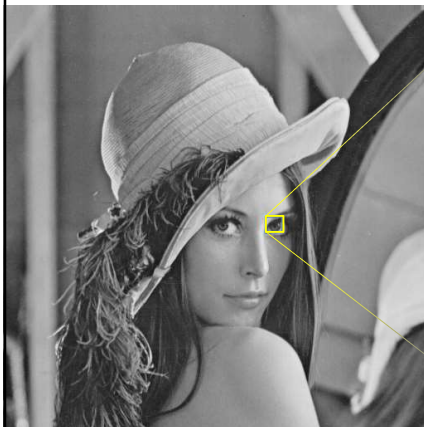


Per un esempio reale: usare il software opensource Inkscape (link sul sito del corso), produrre un immagine qualsiasi, salvare in formato SVG, trovare il file, aprirlo con un editore di testo,



9

Raster Image: gray-scale image



190	187	189	192	192	189	183	172	164	154	139	124	122	122	129	122	111	110	118	112
191	189	191	189	191	190	183	169	158	144	138	136	129	126	136	120	90	81	88	73
194	194	193	190	192	187	180	162	153	138	137	132	103	110	97	71	59	57	55	51
198	198	198	193	193	183	170	151	138	115	100	97	68	70	57	49	50	55	53	56
203	199	197	190	185	173	159	132	118	87	58	62	53	51	54	51	50	52	51	51
203	202	194	188	178	158	128	91	78	59	56	60	49	55	58	55	49	57	62	58
207	203	199	186	156	118	75	61	56	53	59	53	51	52	62	51	52	71	86	91
208	202	189	159	101	56	54	58	51	47	50	55	54	50	57	63	51	86	127	124
207	197	167	112	65	49	50	51	45	46	42	49	59	58	81	113	92	75	157	150
204	181	122	81	54	50	49	44	43	49	44	44	55	56	91	148	128	69	161	164
193	143	92	67	50	50	53	60	52	48	43	45	61	57	77	143	137	76	150	176
158	111	86	69	56	50	65	67	63	54	45	40	60	60	68	99	106	70	143	179
129	101	78	79	78	51	60	84	80	63	42	46	72	85	71	83	76	69	149	178
124	90	85	103	97	61	61	94	100	89	75	67	87	105	85	75	58	87	162	176
110	93	105	120	120	93	58	85	97	100	93	86	88	89	87	70	64	123	174	173
104	105	108	128	134	118	81	65	85	103	100	96	98	82	69	71	111	161	179	175
99	111	114	129	144	148	111	90	77	75	89	88	86	75	76	115	158	180	182	177
100	97	107	125	137	150	139	114	105	88	79	79	86	100	123	156	183	181	177	172


Ogni pixel un livello di grigio
 (memorizzato con un numero da 0: nero a 255: bianco)



10

Raster Images: resolution

10x19 20x37 38x70 158x300




11

Raster Image: RGB image

Red: 150
Green: 77
Blue: 103

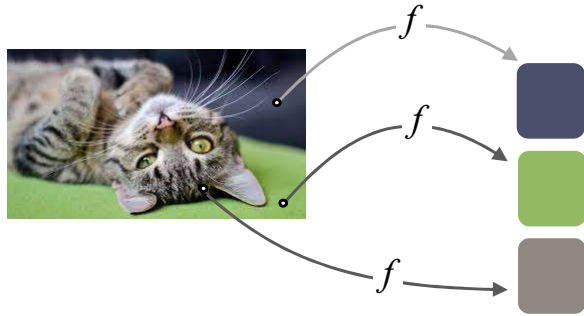
Ogni pixel, un "colore"
tre valori di intensità per rosso, verde, blu




12

Immagine

- ✓ Un assegnamento di un colore ad ogni punto di una regione piana rettangolare
- ✓ Quindi una funzione f da (x,y) a (colore)




- ✓ Come rappresento digitalmente un colore?



13

Rappresentazione digitale del colore

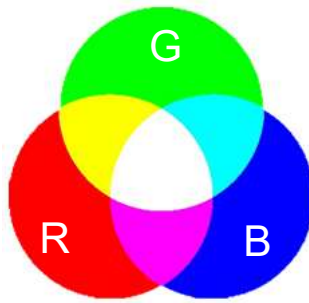
- ✓ Lo studio dei colori come fenomeno fisico, della sua misurazione e rappresentazione è il soggetto di alcune discipline
 - ⇒ **Colorimetria**: studia la misurazione e rappresentazione del colore
 - ⇒ **Radiometria / Fotometria** : ramo della fisica che studia della *radiazione elettromagnetica* (che comprende la luce)
- ✓ Alcune domande interessanti:
 - ⇒ quale caratteristiche fisiche di una luce determinano «di che colore è»?
 - ⇒ quale caratteristiche fisiche di un oggetto fisico (un «materiale») determinano «di che colore è»?
 - ⇒ quale fenomeno fisico determina la percezione del «colore» da parte di un essere umano?
 - ⇒ quale fenomeno fisico determina la misurazione del «colore» da parte di un dispositivo di cattura, come una macchina fotografica digitale (o analogica), o di una videocamera?
- ✓ In questo corso, ci accontentiamo di rispondere ad una quesito più semplice:
 - ⇒ Cosa devo mandare ad un monitor affinché sia riprodotto un dato colore in uno dei suoi pixel?



15

Sintesi additiva del colore

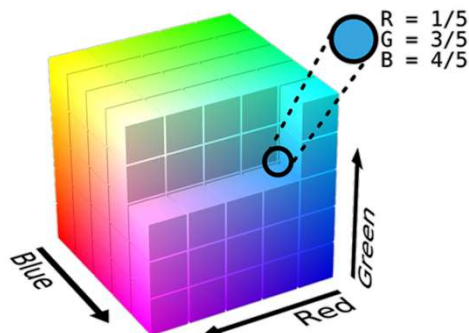
- ✓ Il pixel di un monitor riproduce un dato colore sovrapponendo 3 luci, una rossa (R), una verde (G), una blu (B), a 3 intensità diverse
 - ⇒ Questa soluzione è giustificata da considerazioni sulla percezione umana (che ricadono nel dominio della colorimetria)
vedi: la **teoria del tristimolo**
- ✓ Per determinare quale colore riprodurre dobbiamo scegliere tre valori di intensità (uno per R, uno per G, e uno per B)
 - ⇒ ciascuno scelto fra un massimo e un minimo (es fra 0.0 e 1.0, o fra 0 e 255)
 - ⇒ i valori min e max corrispondono rispettivamente alla massima e minima intensità della luce producibile dai tre emettitori (che congiuntamente costituiscono un pixel sul monitor)



16

Spazio colore

- ✓ Un colore (da riprodursi su un monitor) è dunque rappresentabile da una tripletta di valori scalari
 - ⇒ Questa struttura dati somiglia a quella per un punto 3D / vettore 3D: tre componenti scalari (seppur chiamati R,G,B invece che X,Y,Z)
 - ⇒ I valori di R,G,B sono interpretabili come le coordinate di un punto di uno «spazio colore» 3D: in cui ogni punto è un colore diverso
 - ⇒ questo spazio colore «RGB» è solo un esempio di «Spazio» Colore


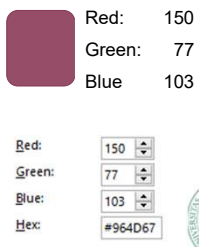


17

Rappresentazioni di un colore in spazio RGB

✓ Ogni «canale» (R, G, B) è rappresentato da...
...un numero intero a n bits

- ⇒ n è detta la profondità del canale
- ⇒ Quindi, R G e B sono numeri da 0 a $2^n - 1$
- ⇒ caso più diffuso: $n = 8$ (1 byte per canale)
- ⇒ Conseguenze:
 - Livello massimo esprimibile: = 255
 - In decimale, un colore può essere espresso come, ad es (150,77,103)
 - In esadecimale, il colore è espresso come, ad es, $0\times$ **964D67**
 $\underbrace{\quad\quad\quad}_{150} \underbrace{\quad\quad\quad}_{77} \underbrace{\quad\quad\quad}_{103}$
 - per es, in HTML o CSS

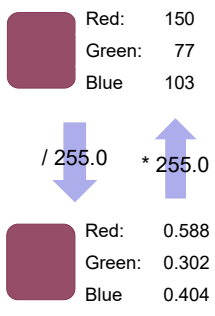


18

Rappresentazioni di un colore in spazio RGB

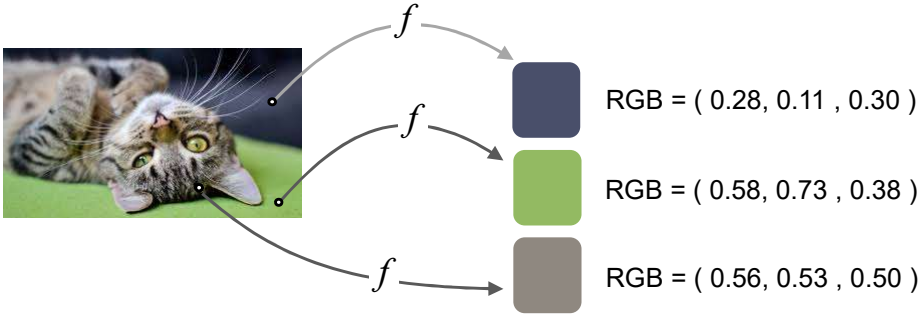
✓ Ogni «canale» (R, G, B) è rappresentato da...
...uno scalare da 0 a 1

- ⇒ Si può usare la stessa struttura dati utilizzata per punti e vettori
- ⇒ Per es, nel linguaggio GLSL il tipo `vec3` si usa per rappresentare punti, vettori, o colori e le sue tre componenti scalari possono essere chiamate intercambiabilmente
 - x, y, z (coordinate del punto)
 - r, g, b (canali del colore)



19







Come rappresento digitalmente un colore?




The diagram illustrates the process of digitizing color. On the left, a photograph of a cat is shown. Three arrows, each labeled with the function f , point from different regions of the cat's fur to three distinct color swatches on the right. Each swatch is accompanied by its RGB value:

- Dark blue swatch: $RGB = (0.28, 0.11, 0.30)$
- Light green swatch: $RGB = (0.58, 0.73, 0.38)$
- Gray swatch: $RGB = (0.56, 0.53, 0.50)$

Below the main diagram, a grid of color swatches is provided with their RGB values:


 $RGB = (0, 0, 0)$ [nero]	 $RGB = (0.00, 0.69, 0.98)$
 $RGB = (1, 1, 1)$ [bianco]	 $RGB = (1.00, 0.75, 0.00)$
 $RGB = (1, 0, 0)$	 $RGB = (0.50, 0.50, 0.50)$



20

Caratteristiche delle immagini Raster

- ✓ **Resolution** = numero di linee e colonne
 - ⇒ Per esempio: 640×480 pixel
 - ⇒ Esprimibile anche in MegaPixel (es. $1000 \times 1000 = 1\text{MPixel}$)
- ✓ Ogni pixel ha k **canali**
 - ⇒ 1 canale = 1 valore scalare
 - ⇒ Immagine RGB = 3 canali: Red, Green, Blue
 - ⇒ Immagine toni di grigi (gray-scale) = 1 canale (gray level)
- ✓ Ogni canale è memorizzato con un dato numero di bit
 - ⇒ se 8: ho $2^8 = 256$ valori possibili, min = 0 and max = 255
 - ⇒ Immagine B & W = il suo canale ha un 1 bit (0 = B e 1 = W)
- ✓ **image depth** (profondità): bit totali per un pixel
 - ⇒ per es: RGB con 8 bits per channel («true color») = 24 bits
 - ⇒ per es: B & W image (una «bitmap»): = 1 bit



21

Immagini: raggio dinamico

- ✓ Rapporto fra luminosità del punto più luminoso e quello più buio
- ✓ HDRI – High Dynamic Range Images:
immagini con raggio dinamico elevato
 - ⇒ Possono richiedere più bit per canale che non i soliti 8



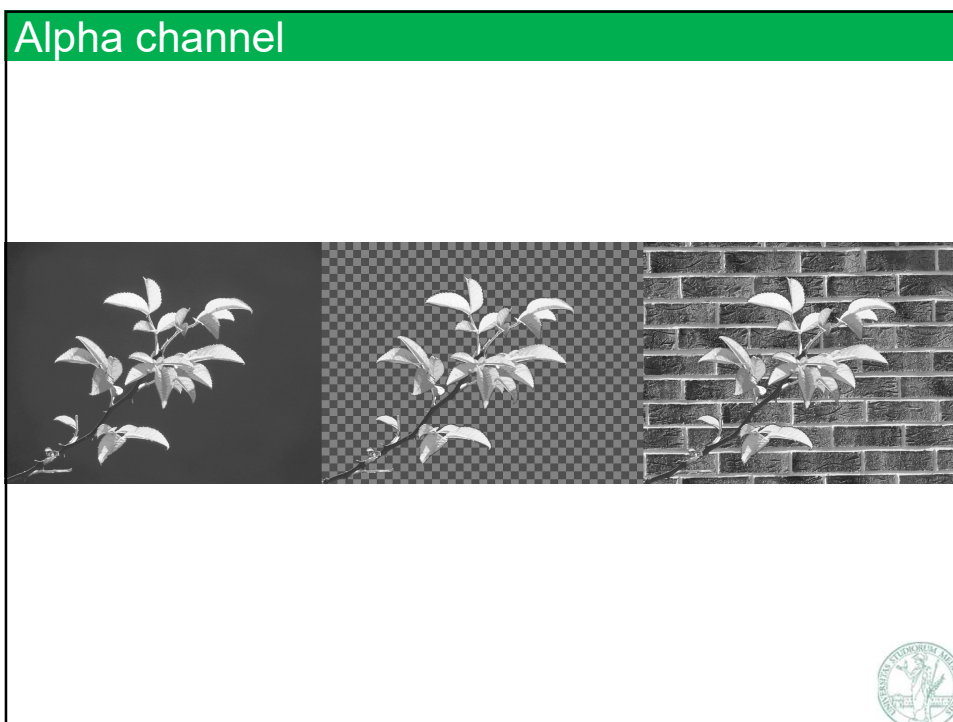
22

Immagini Raster e RAM

- ✓ Totale memoria (in bit):
 $resX \cdot resY \cdot imageDepth$
 - ⇒ pes es: una immagine 240×480 "true color"
memoria = $240 \cdot 480 \cdot 3 \text{ bytes} = 345.600 \text{ bytes}$
cioè $240 \cdot 480 \cdot 24 \text{ bits} = 2.764.800 \text{ bits}$
 - ⇒ per es: immagine 4000×2000 (8 MegaPixel) "true color":
 $4000 \cdot 2000 \cdot 24 \text{ bits} =$
 $4000 \cdot 2000 \cdot 3 \text{ bytes} = 24 \text{ MegaByte}$
- ✓ Compressione spesso necessaria
 - ⇒ Molti schemi di compressione sono usati per le immagini
- ✓ Caratteristiche degli schemi di compressione:
 - ⇒ Lossless : se non deteriorano il dato
 - ⇒ Lossy : più aggressivi (es 1:10), ma deteriorano il dato
 - ⇒ Texture compression: una compressione lossy di tipo speciale utilizzabile dalle immagini di texture



23



24


Alpha Channel e Alpha blending

- ✓ Nelle immagini semitrasparenti, ai tre canali R,G,B si affianca un quarto canale, α , il cui valore rappresenta il *livello di opacità* di quel pixel
 - ⇒ da $\alpha = 0.0$ (0, come intero a 8 bit) → pixel completamente trasparente
 - a $\alpha = 1.0$ (255, come intero a 8 bit) → pixel completamente opaco
- ✓ L'immagine può così essere visualizzata (ad esempio, in una pagina web) come «foreground» sullo sfondo di un'altra immagine (di «background»)
- ✓ **Alpha blending:**

un pixel di «foreground» con colore $\mathbf{f} = \begin{pmatrix} f_R \\ f_G \\ f_B \end{pmatrix}$, e valore di trasparenza α ,

sovrapposto ad un pixel di «background» di colore $\mathbf{b} = \begin{pmatrix} b_R \\ b_G \\ b_B \end{pmatrix}$

risulterà in un colore interpolato (blended) col parametro α :

$$(1 - \alpha) \begin{pmatrix} b_R \\ b_G \\ b_B \end{pmatrix} + \alpha \begin{pmatrix} f_R \\ f_G \\ f_B \end{pmatrix}$$


25

Vector Images: alcuni formati file comuni

- ✓ **SVG:**
 - ⇒ sviluppato da W3C
 - ⇒ molto completo
 - ⇒ consiste in lista di primitive descritte in un formato basato su XML
 - ⇒ largo uso di curve di Bèzier (per contorni, eventualmente riempiti)
 - ⇒ in aggiunta a: forme base, testo (con font), pattern di riempimento etc...
 - ⇒ formato nativo del Web! (si può utilizzare come elemento delle pagine web, è interpretato correttamente da tutti i browser)
 - ⇒ Es. di strumento di editig: inkscape (link dalla pagina del corso)
- ✓ **PostScript (PS):**
 - ⇒ pensato per stampanti
 - ⇒ descrive un insieme di comandi da mandare alla stampante per comporre l'immagine (come controllo per quale inchiostro usare, etc)
- ✓ **Portable Document Format (PDF)**
 - ⇒ proprietario (by Adobe)
 - ⇒ include sottoinsieme di PS



27

Rarter Images: alcuni formati file comuni


- ✓ **PNG (Portable Network Graphics):**
 - ⇒ compressione lossless (stesso algoritmo dei file zip)
 - ⇒ molti sottoformati, a 1, 3 o 4 canali (canale alpha per trasparenza)
 - ⇒ ottime compressioni per immagini artificiali
- ✓ **JPEG (Joint Photographic Experts Group):**
 - ⇒ (tipicamente) compressione lossy (introduce "artefatti di compressione") (basata su "DCT": discrete cosine transform)
 - ⇒ 3 canali, sempre di 8 bit ciascuno
 - ⇒ ottime compressioni su immagini naturali (per es, fotografia digitale)
 - ⇒ non dotato di canale alpha
- ✓ **JPEG 2000**
 - ⇒ Uno sviluppo su JPEG (per ora un po' meno diffuso), con meno limiti
- ✓ **GIF (compuserve)**
 - ⇒ un capriccio della storia dei formati di immagini raster (uso: in declino)
 - ⇒ usato per piccole animazioni per tre decenni (dai '90, ai 2010)
 - ⇒ Alpha channel di un bit (ogni pixel: fully transparent o fully opaque)



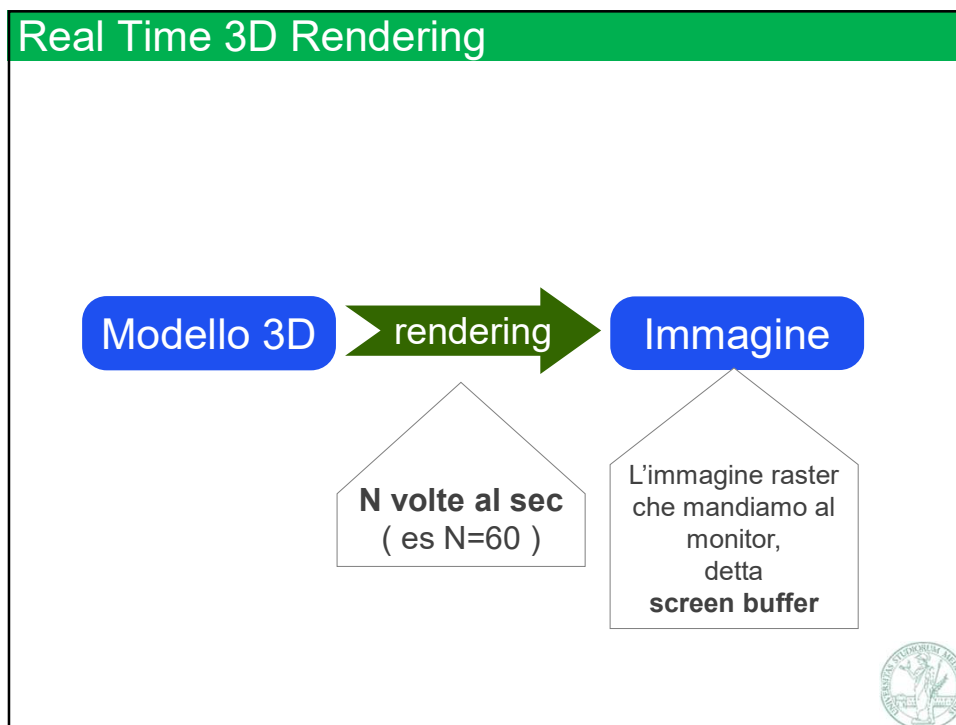
28

Rarter Images: formati meno comuni ma utili

- ✓ TIFF
 - ⇒ usabile con compressione lossy oppure lossless
 - ⇒ specializzato per profondità di pixel maggiori di 24 (hi-dynamic range data – più di 8 bit per canale)
 - ⇒ diffuso per fotografia digitale di alta qualità
- ✓ RAW
 - ⇒ una cattura diretta dell'output di un sensore (CCD di una fotocamera digitale)
 - ⇒ non compresso
 - ⇒ non processato
- ✓ PNM ("portable any map")
 - ⇒ nessuna compressione
 - ⇒ valori dei pixel values memorizzati come ASCII (o in binary)
 - ⇒ non molto diffuso
 - ⇒ ma, utile: formato semplice, "human readable" (ASCII numbers)
 - ⇒ e, banale da leggere (è possibile scrivere un importer / exporter con poche righe di codice, in qualsiasi linguaggio, senza liberie)



29



35

Real Time 3D Rendering: fill rate (esempio)

- ✓ 1 pixel = 32 bit = 4 bytes "pixel depth"
- ✓ screen buffer res = 1024 x 768 pixels "screen resolution"
- ✓ frame al secondo (fps) = 60 Hz "frame rate"
- ✓ total = 4 x 1024 x 768 x 60 byte al sec "fill rate"

Fill Rate:
188 MegaBytes / sec

anche ignorando molti fattori, come l'overdraw
(necessità di sovrascrivere lo stesso pixel del buffer
più volte durante il rendering)



36

GPU per Real-Time 3D Rendering

- ✓ Problema difficile, come si evince dai fill rate
⇒ fortunatamente,
è anche massicciamente parallelizzabile
⇒ "embarrassingly parallel"
- ✓ Ingrediente *base* della soluzione:
hardware specializzato: la GPU



37