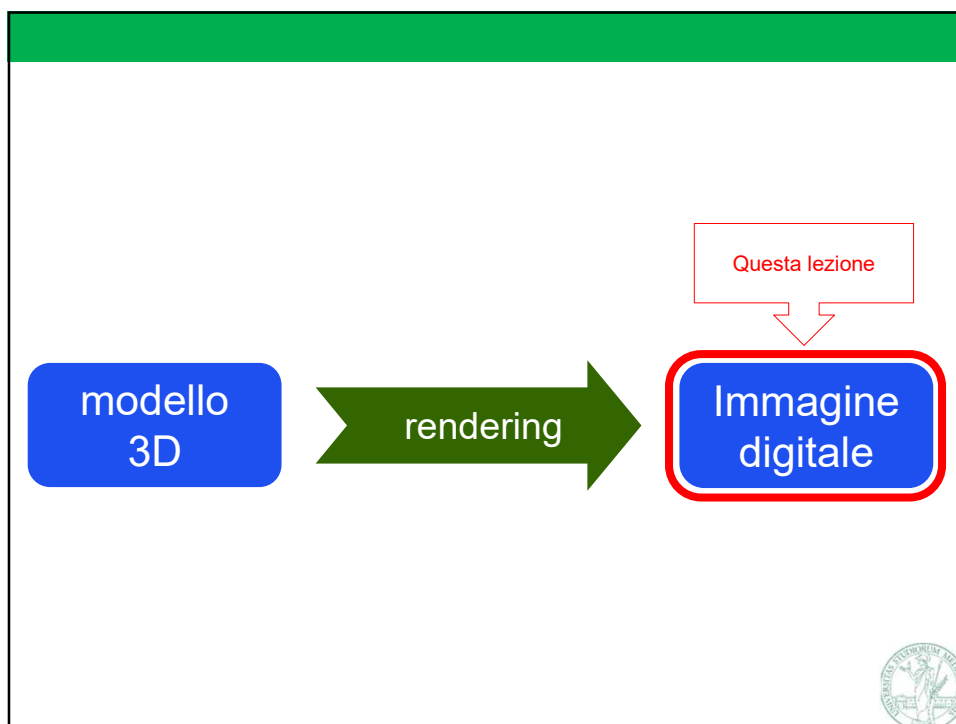




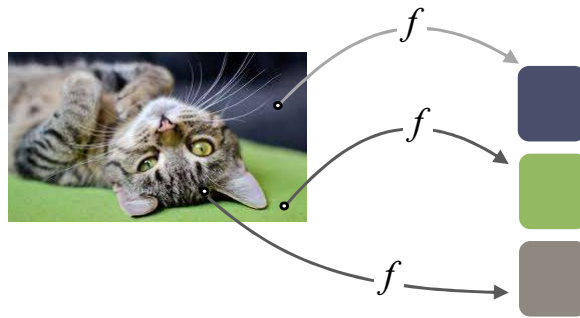
5



6

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?



7

Rappresentazione digitale del colore

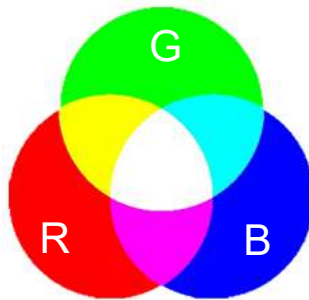
- ✓ Lo studio dei colori come fenomeno fisico, della sua misurazione e rappresentazione è il soggetto di alcune discipline
 - ⇒ **Colorimetria**: studia la misurazione e rappresentazione del colore
 - ⇒ **Radiometria / Fotometria** :
ramo della fisica che studia della *radiazione elettromagnetica* (che comprende la luce)
- ✓ Alcune domande interessanti:
 - ⇒ quali caratteristiche fisiche di una luce determinano «di che colore è»?
 - ⇒ quali 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 per riprodurre un dato colore in uno dei suoi pixel?



8

Sintesi additiva del colore

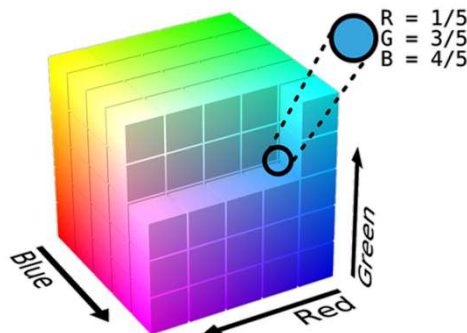
- ✓ Il pixel di un monitor riproduce un dato colore sovrapponendo 3 luci ad 3 date intensità: una rossa (R), una verde (G), una blu (B)
 - ⇒ Questa soluzione è giustificata da considerazioni sulla percezione umana (che ricadono nel dominio della colorimetria) la **teoria del tristimolo**
- ✓ Per determinare quale colore riprodurre dobbiamo scegliere tre valori di intensità (per R, G ,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 prodotta da tre emettitori (che insieme costituiscono un pixel sul monitor)



9

Spazio colore

- ✓ Un colore (che vada riprodotto 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
 - ⇒ lo spazio colore «RGB» è solo un esempio di Spazio Colore



10

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”)



11

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
```

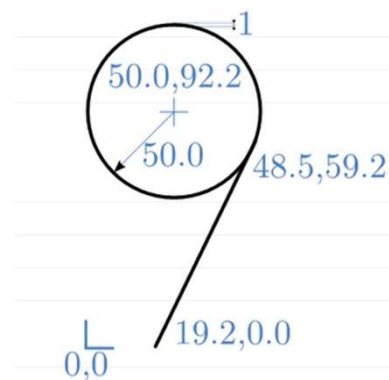
```
SEGMENT
```

```
endpoint_one 19.2, 0.0
```

```
endpoint_two 48.5, 59.2
```

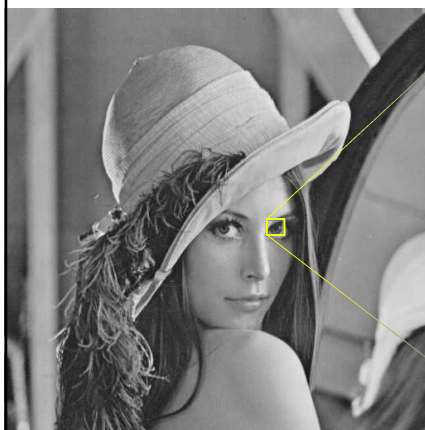
```
line_color 0.0, 0.0, 0.0
```

```
line_thickness 1pt
```



12

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	138	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	173
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)

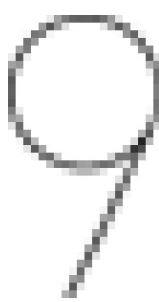


13

Raster Images: resolution



10x19



20x37



38x70



158x300



14

Raster Image: RGB image



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



15

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, come numero 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, $0x964D67$

$\underbrace{\hspace{1.5cm}}_{150}$
 $\underbrace{\hspace{1.5cm}}_{77}$
 $\underbrace{\hspace{1.5cm}}_{103}$
 - per es, in HTML o CSS



Red: 150
Green: 77
Blue: 103

Red:

Green:

Blue:

Hex:



16

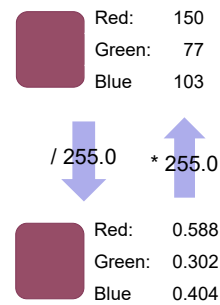
Rappresentazioni di un colore in spazio RGB

✓ 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

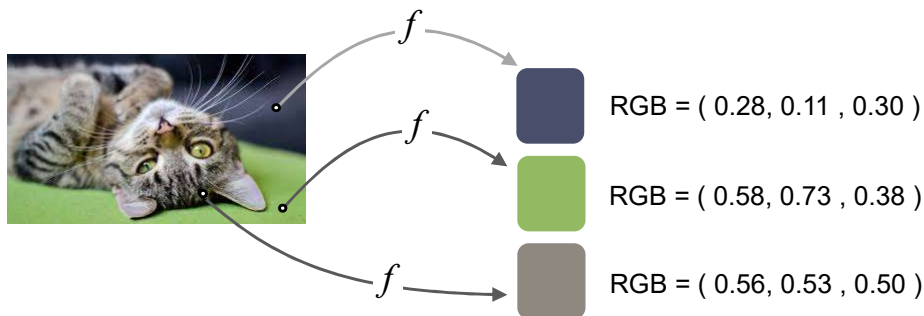
⇒ Per es, nel linguaggio GLSL
il tipo `vec3` si usa
per rappresentare
punti, vettori, o colori
e le sue tre componenti scalari
possono essere chiamate
intercambiabilmente

- x, y, z (coordinate del punto)
- r, g, b (canali del colore)



17

Come rappresento digitalmente un colore?



	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)



18

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



19

Immagini Raster e RAM

- ✓ **Totale memoria: $\text{resX} \cdot \text{resY} \cdot \text{imageDepth}$**
 - ⇒ per 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»:
 $4000 \cdot 2000 \cdot 3 = 24 \text{ MegaByte}$
- ✓ **Compressione spesso necessaria**
 - ⇒ Molti schemi di compressione sono usati
- ✓ **Tipi di schemi di compressione:**
 - ⇒ Lossless : non deteriorano il dato
 - ⇒ Lossy : più aggressivi (es 1:10) deteriorano il dato
 - ⇒ Texture compression: una compressione lossy di tipo speciale utilizzabile dalle immagini di texture



21

Pros and Cons

✓ Vector images:

- ⇒ independent from the device resolution!
- ⇒ well suited for: logos, diagrams, stylized drawings ...
- ⇒ can be very compact in space
- ⇒ interpretation: difficult (a rendering required)
- ⇒ resolution (number of primitive, of control points...) is adaptive
- ⇒ when zoomed in:
there may be little detail to look at, but it looks still good

✓ Raster images

- ⇒ well suited for: natural images (e.g. photographs)
- ⇒ quality depends on image resolution
- ⇒ interpretation: direct (monitors can display them directly)
- ⇒ when zoomed in: artifacts



23

Raster Images: alcuni formati comuni (note)

✓ PNG (Portable Network Graphics):

- ⇒ lossless compression (based on same algorithm as Zipped files)
- ⇒ many formats, including with 3 or 4 channels
- ⇒ good for synthetic images

✓ JPEG (Joint Photographic Experts Group):

- ⇒ (typically) lossy compression (based on "DCT": discrete cosine transform)
- ⇒ 3 channels of 8 bits each
- ⇒ good for natural images (digital photography)

✓ JPEG 2000

- ⇒ an advancement over Jpeg (didn't catch on as much)

✓ GIF (compuserve)

- ⇒ strange *quirk* of the used image format
- ⇒ used for tiny animations over two decades ('90s, 2000's)
- ⇒ use will decline



24

Rarer Images: not so common formats

- ✓ **TIFF**
 - ⇒ can be lossless or lossy
 - ⇒ hi-dynamic range data (more than 8 bits per channel)
 - ⇒ used for hi-quality digital photography
- ✓ **RAW**
 - ⇒ a direct capture of a sensor (es camera CCD)
 - ⇒ non compressed
 - ⇒ non processed
- ✓ **PNM (portable any map)**
 - ⇒ no compression
 - ⇒ pixel values stored as ASCII (or binary)
 - ⇒ not very used
 - ⇒ but, useful: simple format, human readable values (ASCII numbers)
 - ⇒ and, trivial to parse (an importer / exporter can be written in any language in a few lines of code)



25

Vector Images: common formats

- ✓ **SVG:**
 - ⇒ developed by W3C
 - ⇒ a XML file describing the primitives
 - ⇒ based on Bèzier curves (including, filled ones)
 - ⇒ plus: basic shapes, text, colors, patterns ...
 - ⇒ directly embeddable in Web pages (understood by all browser)
 - ⇒ example of Opensource editor / authoring tool: inkscape
- ✓ **PostScript (PS):**
 - ⇒ designed for printers
 - ⇒ set of instructions to send to a printer to print the image
 - ⇒ includes ink control
- ✓ **Portable Document Format (PDF)**
 - ⇒ by Adobe
 - ⇒ includes subsets of PS



26

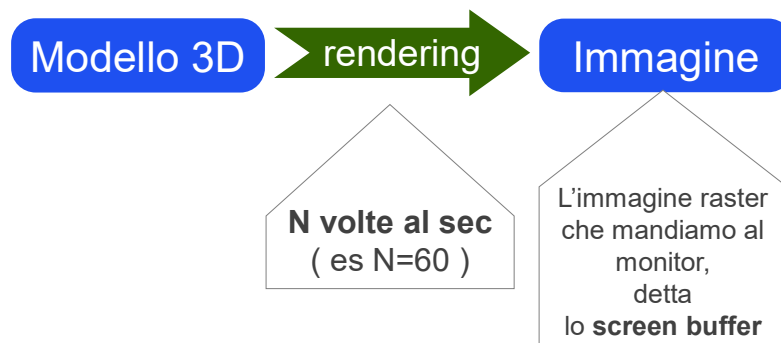
Immagini digitali: metadata

- ✓ Molti formati di file per immagini digitali includono utilissimi metadati, per es
- ✓ Exif -- Exchangeable image file format
 - ⇒ date
 - ⇒ used camera settings
 - ⇒ thumbnail
 - ⇒ description
 - ⇒ copyright
 - ⇒ geolocation (GPS coordinates of capture)
 - ⇒ ...



27

Real Time 3D Rendering



32

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)



33

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



34