Codifica delle immagini

- Abbiamo visto come la codifica di *testi* sia (relativamente semplice)
  - Un **testo** è una sequenza di **simboli**
  - Ogni **simbolo** è codificato con un **numero**
  - Ed ecco che il testo è trasformato in un numero, e quindi elaborabile da un calcolatore
- Con le immagini la questione è più complessa

Codifica delle immagini

- Qual è “il numero” delle immagini qui sotto?

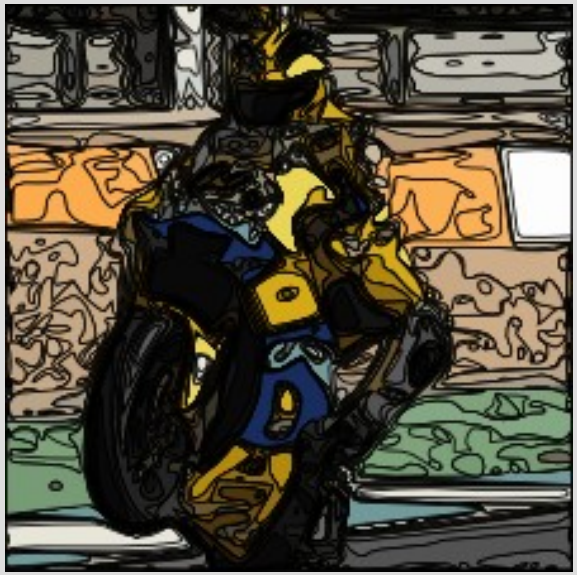
Immagini raster e vettoriali

- Distinguiamo subito due tipi di immagini
  - Le immagini **raster** sono di tipo fotografico; si rappresenta individualmente ogni singolo punto dell'immagine
  - Le immagini **vettoriali** sono essenzialmente disegni; si rappresentano i punti e le curve che formano il disegno

Immagini raster e vettoriali

- A volte la stessa immagine può essere rappresentata come raster...

- ... o, con qualche cambiamento visibile (qui esagerato), come immagine vettoriale



Immagini raster e vettoriali

- In linea di principio, i due tipi sono interscambiabili...
- ... ma in pratica, le “**foto**” sono raster, i “**disegni**” vettoriali
  - nulla vieta comunque di avere dipinti iper-realistici che sembrano foto, o foto posterizzate che sembrano disegni
- Cominciamo a studiare le immagini raster

Immagini raster

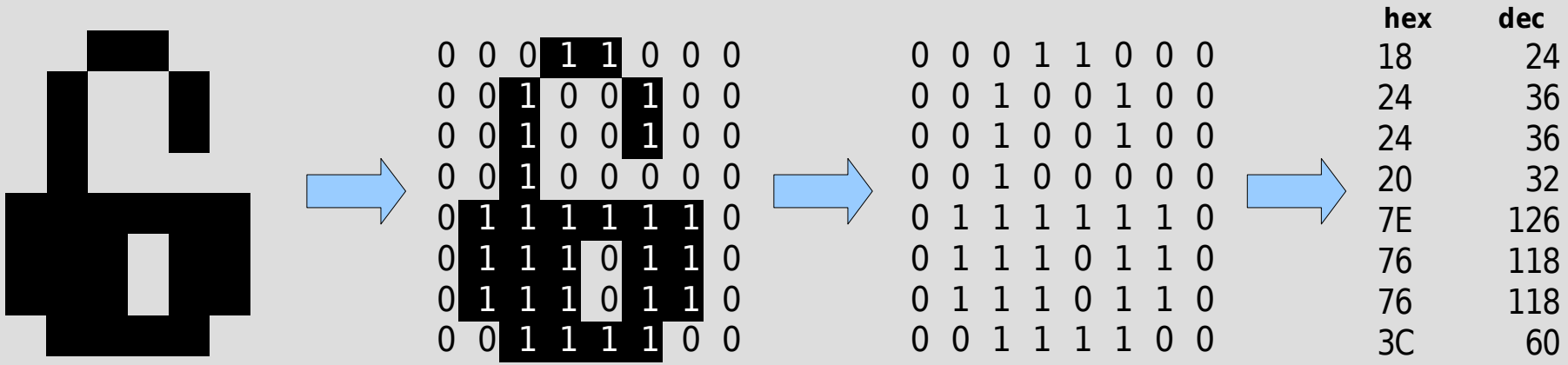
- Un'immagine raster è definita **per punti**
- L'immagine è scomposta in un numero elevato di “punti”, tipicamente quadrati o rettangoli quasi-quadrati
- Ciascuno di questi punti è detto **pixel** (da *picture element*, elemento dell'immagine)
- La densità della griglia di scomposizione, più o meno fitta, è detta **risoluzione**

Immagini raster

Immagini raster

Immagini raster

- Esempio di immagine raster (in bianco e nero):



- Questa è la tecnica base per convertire un'immagine in un numero

Immagini raster

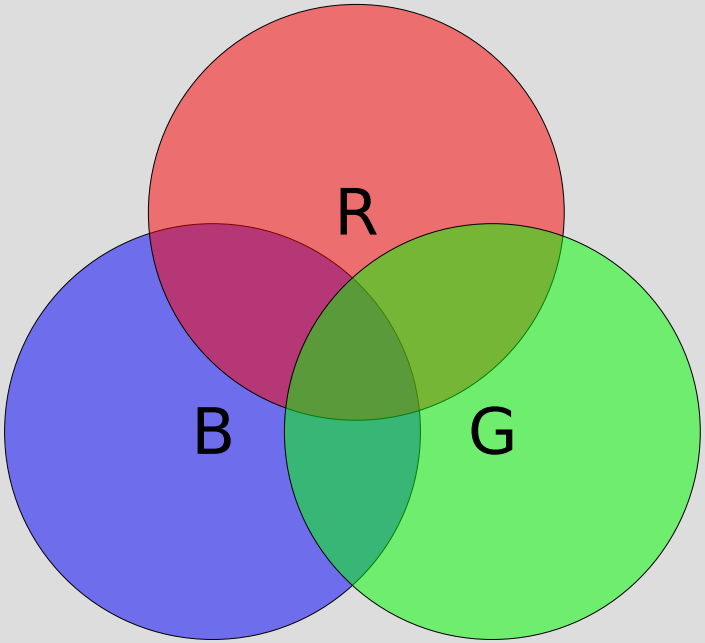
- Di ciascuno dei pixel che compongono l'immagine vengono definite alcune proprietà:
  - il colore (sempre)
  - la trasparenza (in alcuni casi)
- Applicazioni più specializzate definiscono altre proprietà (per esempio, nel trattamento di immagini astronomiche) che non considereremo

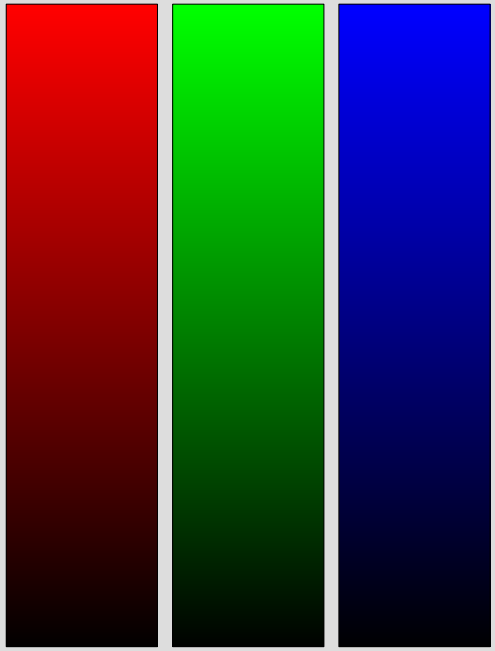
Definizione del colore

- In generale, il **colore** è dato dalla combinazione di più **componenti colore**
- Nel caso più comune, si usano le **componenti primarie**:
  - quantità di rosso (**R**, red)
  - quantità di verde (**G**, green)
  - quantità di blu (**B**, blue)
- Questo modello è detto dunque **RGB**

Definizione del colore

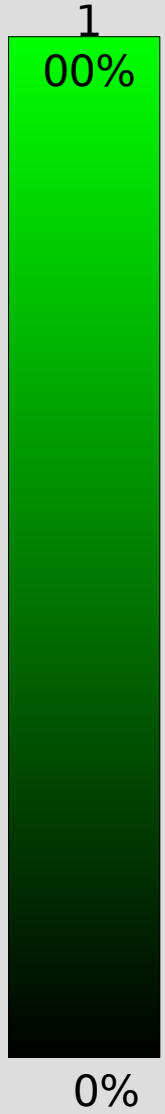
- Combinando quantità variabili di R, G e B si possono ottenere “tutti” i colori





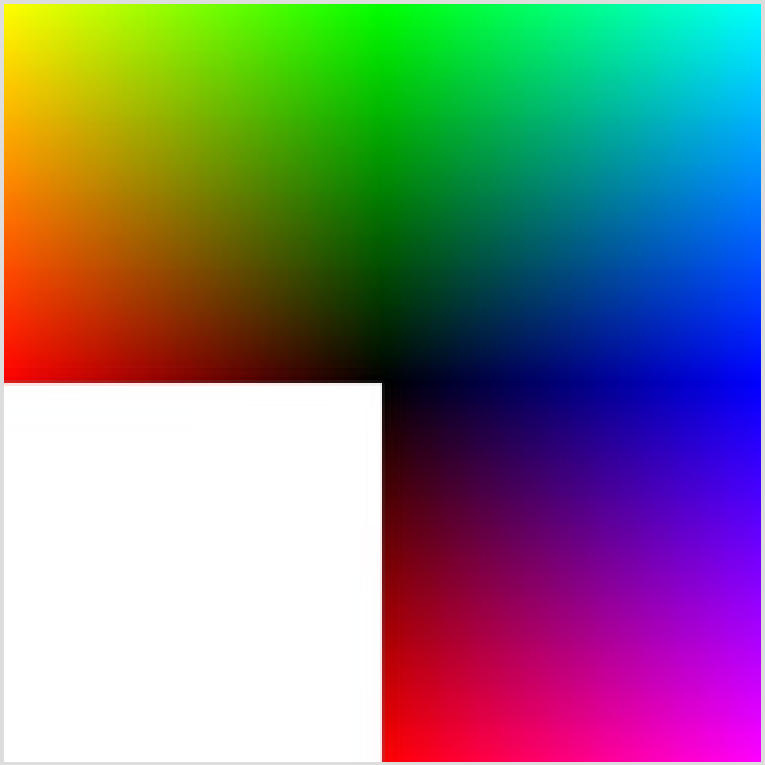
Definizione del colore

- L'**intensità** di ciascun componente deve essere rappresentata (come al solito) **da un numero**
- Concettualmente, si va da 0% (assenza totale del colore) a 100% (colore al massimo della saturazione)
- In pratica, si **discretizza** il valore con un numero intero (una potenza di 2, per esempio 0-7 o 0-255)



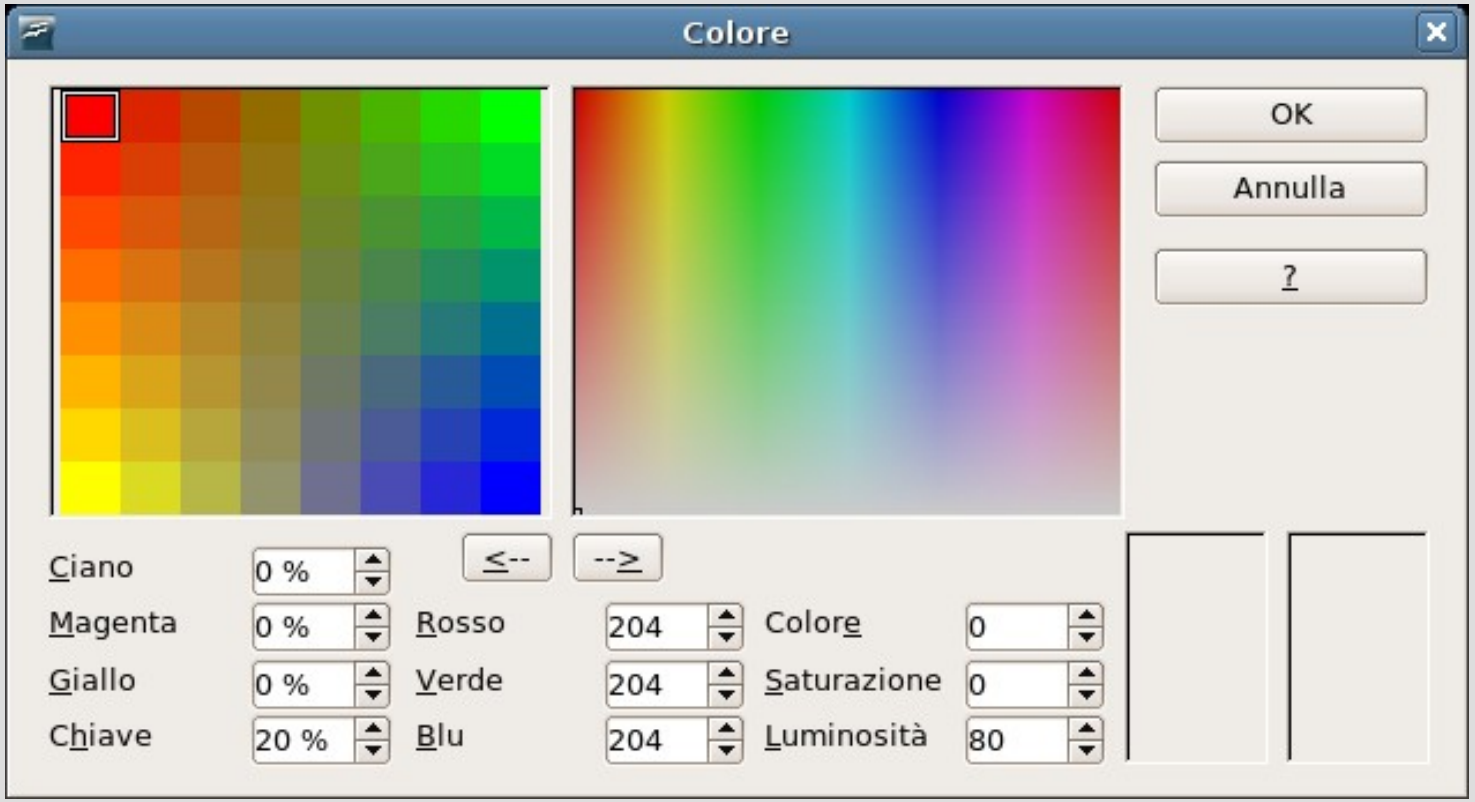
Definizione del colore

- L'assenza di R, G, B produce il nero
- La presenza di R, G, B alla massima intensità produce il bianco
- Un colore è un punto nello spazio dei colori
- Poiché ogni colore ha tre coordinate in questo spazio (R, G, B) la rappresentazione piana è complicata...



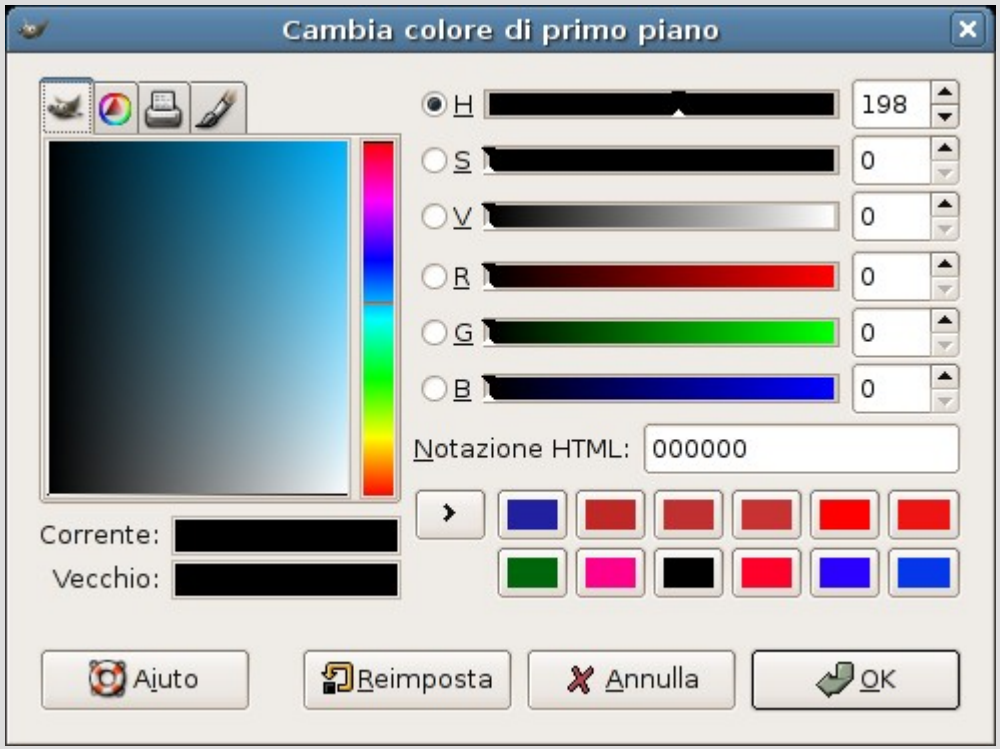
Definizione del colore

- I programmi di grafica offrono spesso varie tavolozze e metodi per definire un colore



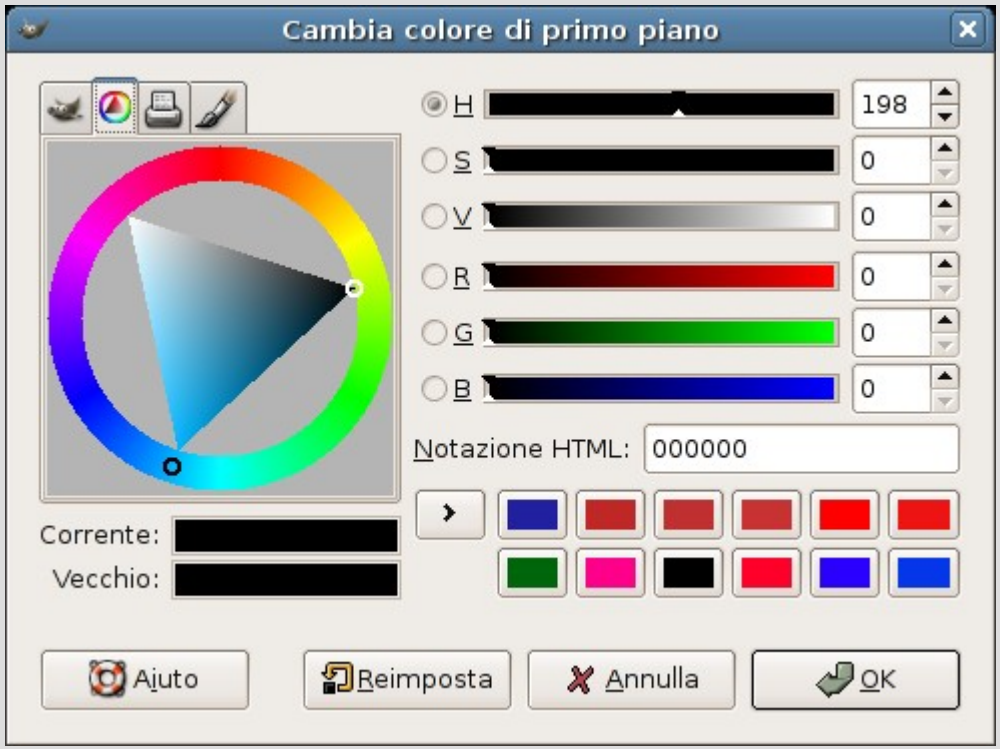
Definizione del colore

- I programmi di grafica offrono spesso varie tavolozze e metodi per definire un colore



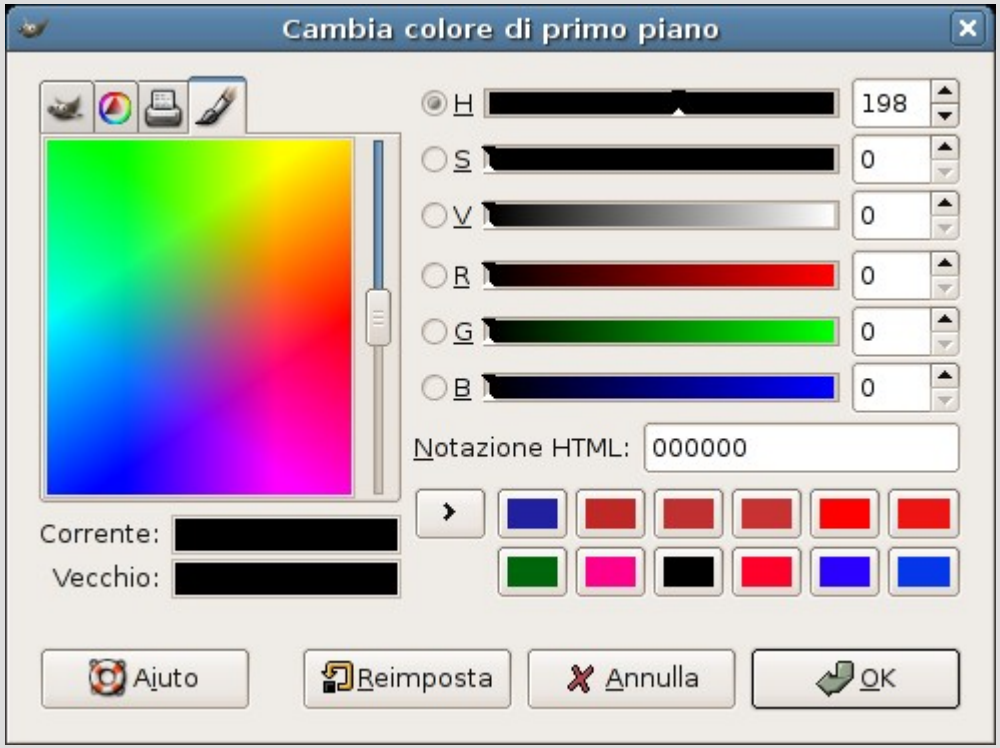
Definizione del colore

- I programmi di grafica offrono spesso varie tavolozze e metodi per definire un colore



Definizione del colore

- I programmi di grafica offrono spesso varie tavolozze e metodi per definire un colore



Definizione del colore

- Studieremo in dettaglio più avanti i diversi modelli colore
- Per il momento, ci basta sapere che un colore può essere codificato con una sequenza di numeri
  - RGB è un modello popolare
  - Si usano da 1 a 16 bit per componente
  - Il valore più comune è 8 bit per componente
  - Possiamo codificare 2^{3x8} = 16.777.216 colori distinti

Definizione del colore

- Esempi:

R (rosso)	G (verde)	B (blu)		Colore
0	0	0		Nero
0	0	128		Blu pieno
128	128	128		Grigio 50%
255	0	0		Rosso pieno
255	255	0		Giallo pieno
153	51	102		Bordeaux
53	94	0		Verde oliva
255	255	255		Bianco
102	51	0		Marrone
255	204	153		Pesca

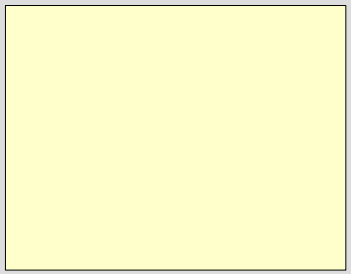
Quiz!

- Che colore corrisponde al seguente codice RGB su 8 bit?

(255,255,204) = ?

Quiz!

- Che colore corrisponde al seguente codice RGB su 8 bit?

(255,255,204) = 

Giallo chiaro

Torniamo alle immagini raster

- Abbiamo visto che un'immagine raster è codificata come una **matrice** di **pixel**
- Ogni pixel codifica il **colore** del punto corrispondente dell'immagine
- La larghezza e altezza dell'immagine, in pixel, danno la **risoluzione spaziale** dell'immagine
- Il numero di bit usati per codificare un colore dà la **risoluzione colore** o **profondità** dell'immagine

Esempi di risoluzione

- 170x170 pixel, RGB a 8 bit per componente
- La risoluzione spaziale non è molto elevata
- La risoluzione colore è buona
- Immagine “giusta” per una pagina web

Esempi di risoluzione

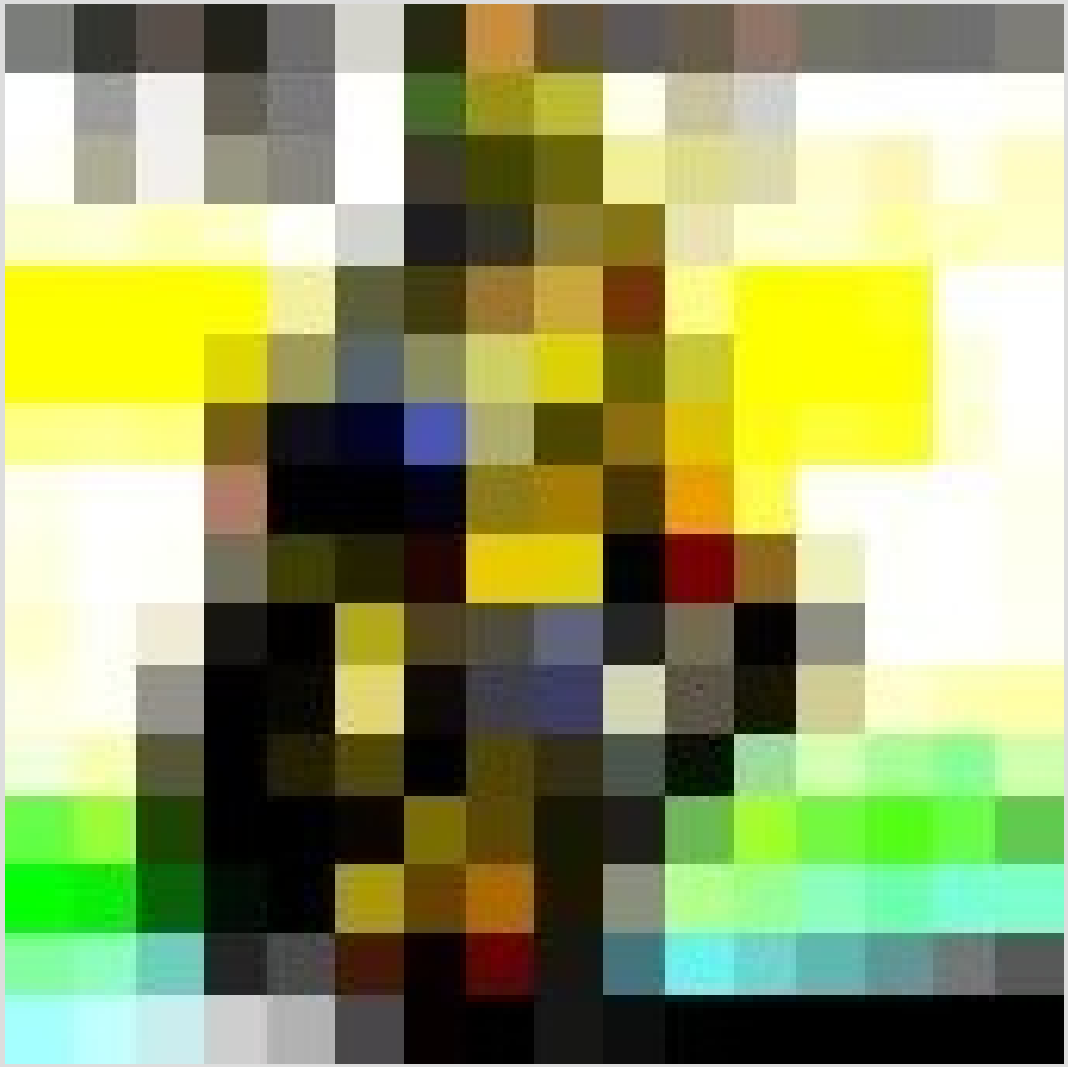
- 32x32 pixel, RGB a 8 bit per componente
- La risoluzione spaziale è molto bassa
- La risoluzione colore è buona
- Usabile forse come **miniatura** (*thumbnail*)

Esempi di risoluzione

- 170x170 pixel, RGB a 3 bit per componente
- La risoluzione spaziale non è molto elevata
- La risoluzione colore è bassa (8 rossi, 8 verdi, 8 blu; 512 colori totali)
- Immagine da “effetto speciale”

Esempi di risoluzione

- 16x16 pixel, RGB a 2 bit per componente
- Un disastro!
- Non va bene neanche come icona!



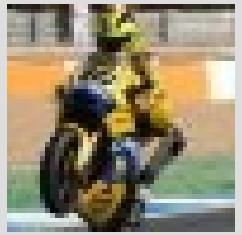
Occupazione di memoria

- Prevedibilmente, l'occupazione di memoria di un'immagine è legata alla sua risoluzione spaziale e di colore
- Tanto maggiori le risoluzioni, tanto migliore la qualità dell'immagine, e tanto maggiore l'occupazione di memoria
- Occupazione in bit = larghezza (in pixel) × altezza (in pixel) × profondità colore (in bit)

Occupazione di memoria

- 170x170 pixel, RGB a 8 bit per componente = 693.600 bit = **86.7 Kb**

- 32x32 pixel, RGB a 8 bit per componente = 24.576 bit = **3 Kb**



- 170x170 pixel, RGB a 3 bit per componente = 260.100 bit = **32.5 Kb**

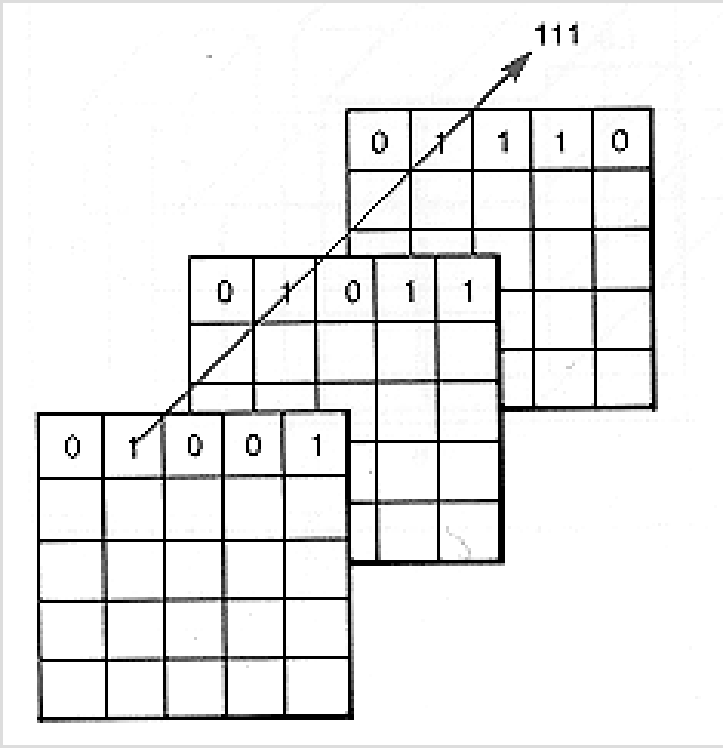
- 16x16 pixel, RGB a 2 bit per componente = 512 bit = **64 byte**

Organizzazione dei dati

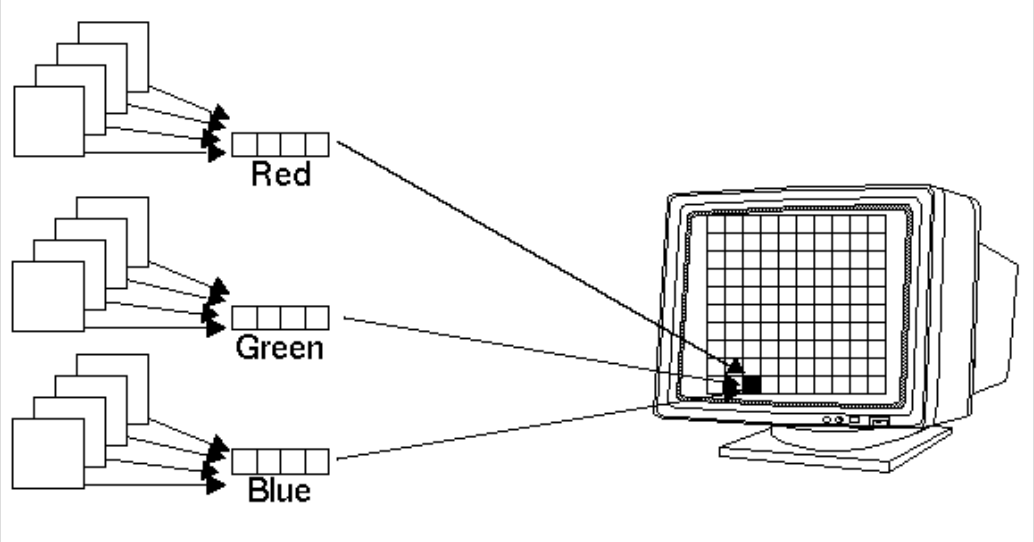
- L'effettiva organizzazione in memoria dei dati che codificano l'immagine può variare a seconda dei casi
- Storicamente, sono stati sviluppate e adottate numerosissime organizzazioni
  - spesso strampalate, legate alle limitazioni hardware di una macchina particolare
- Esistono almeno tre modelli abbastanza diffusi: **a bitplane**, **planari**, **a chunk**

Organizzazione a bitplane

- In questa organizzazione, l'immagine è scomposta in tanti “piani” quanti sono i bit di profondità colore
- I bit che occupano la stessa posizione in piani diversi formano il codice numerico del colore



Organizzazione a bitplane



- Per esempio:
  - 5 bitplane codificano il rosso
  - 6 bitplane codificano il verde*
  - 5 bitplane codificano il blu

---

* L'occhio umano è più sensibile al verde!

Organizzazione a bitplane

- Vantaggi dell'organizzazione a bitplane:
  - si usa **esattamente** la memoria necessaria, non ci sono sprechi
  - i bitplane che codificano i bit più bassi possono essere scartati con piccole perdite di qualità
    - è facile ridurre il numero di colori per risparmiare memoria!
  - alcuni algoritmi di *image processing* funzionano bene sui bitplane
  - supportato direttamente dall'hardware

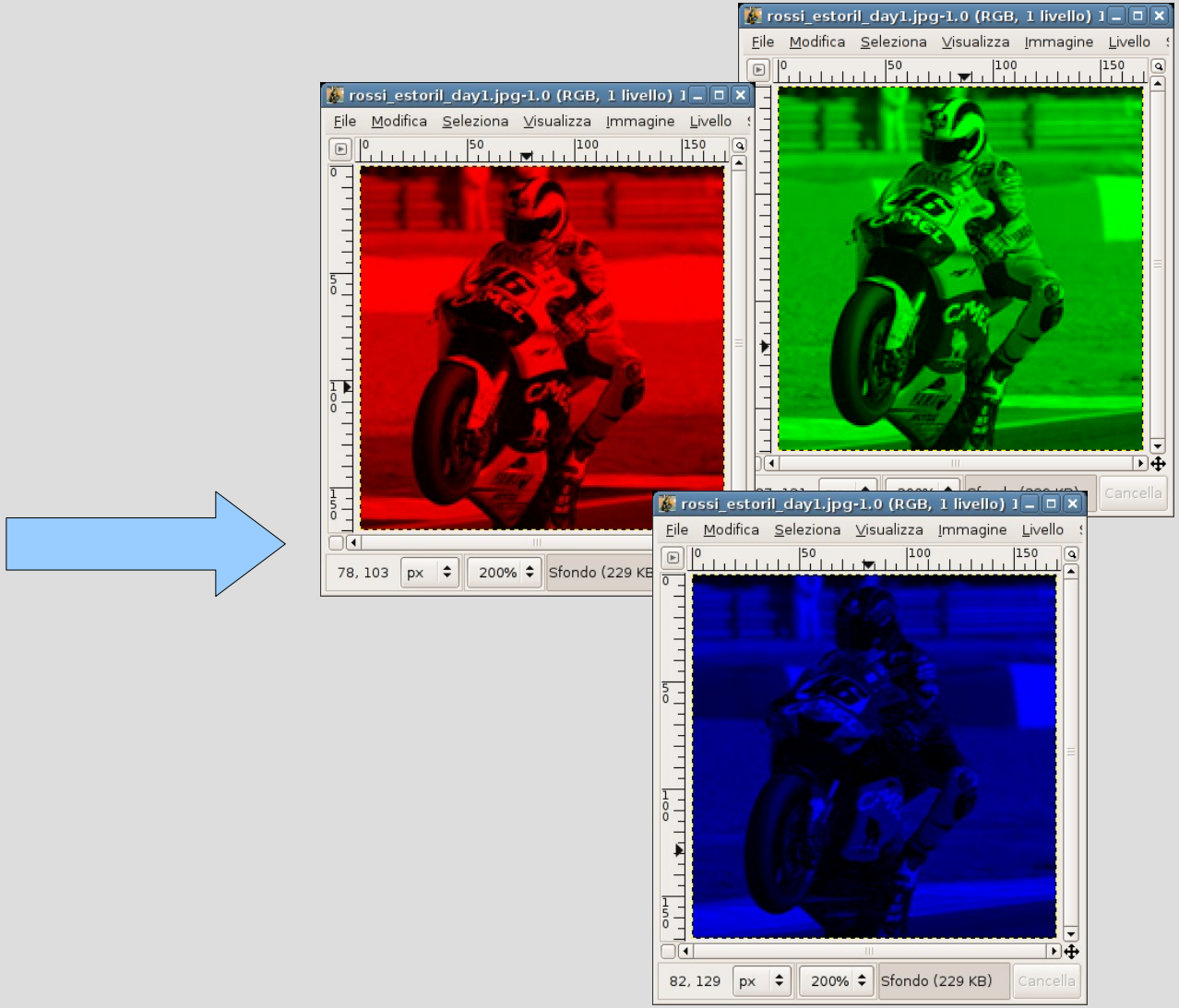
Organizzazione a bitplane

- Svantaggi dei bitplane:
  - per sapere di che colore è un pixel bisogna guardare in vari punti diversi della memoria
    - questo rende inutili le *cache* delle CPU e peggiora le prestazioni della memoria virtuale
  - molte operazioni di disegno sono poco efficienti
    - es: per disegnare un cerchio di colore (255,0,0) devo disegnare 8 cerchi distinti, uno per bitplane
  - se non supportati direttamente dall'hardware, richiedono costose conversioni

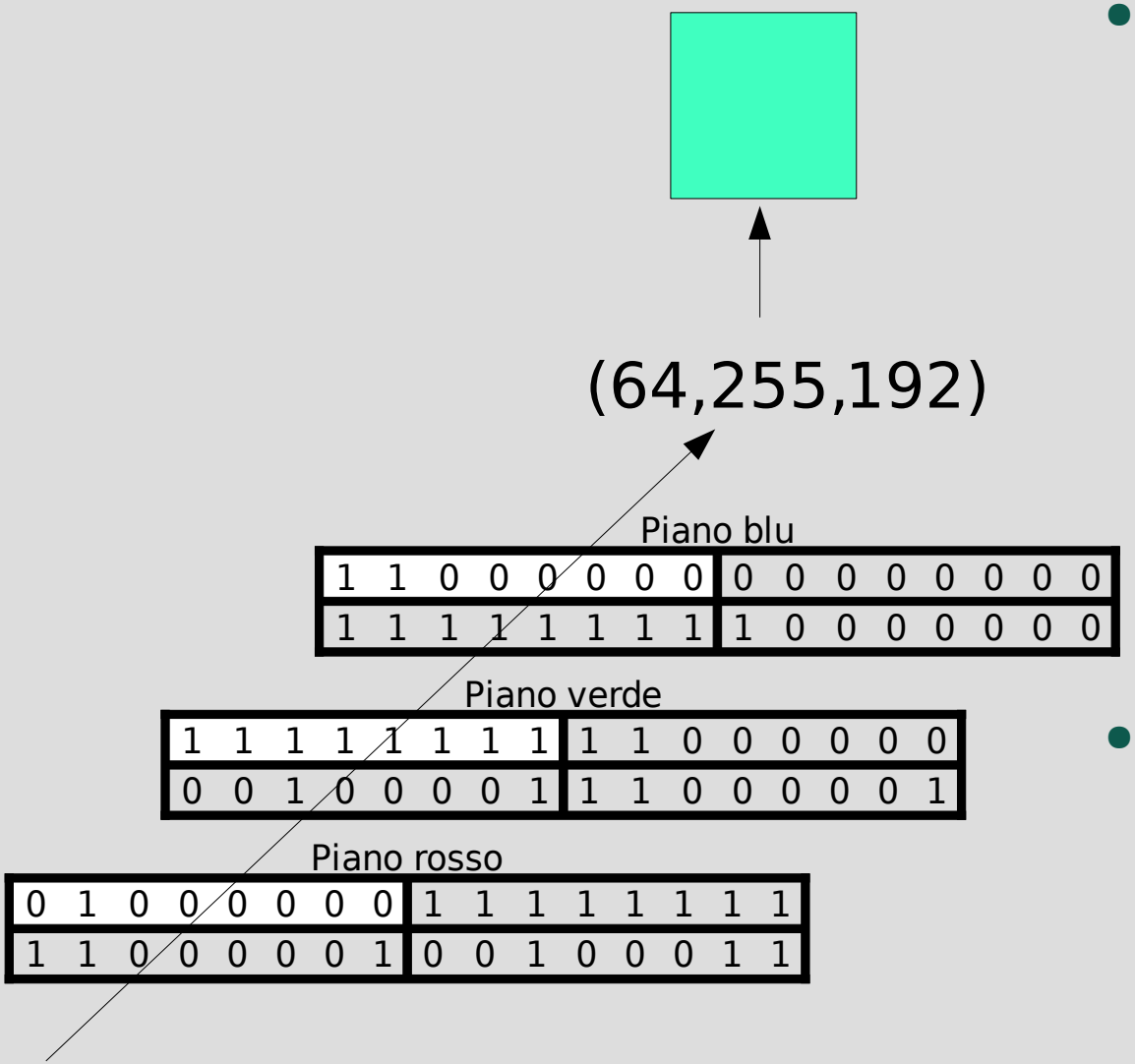
Organizzazione planare

- In questo caso, l'immagine è scomposta in tanti **piani** (o **canali**) quanti sono i componenti colore
- Di solito, avremo quindi un piano rosso, un piano blu, un piano giallo
- In ogni piano, ciascun pixel è codificato con l'intero codice-colore della componente relativa

Organizzazione planare



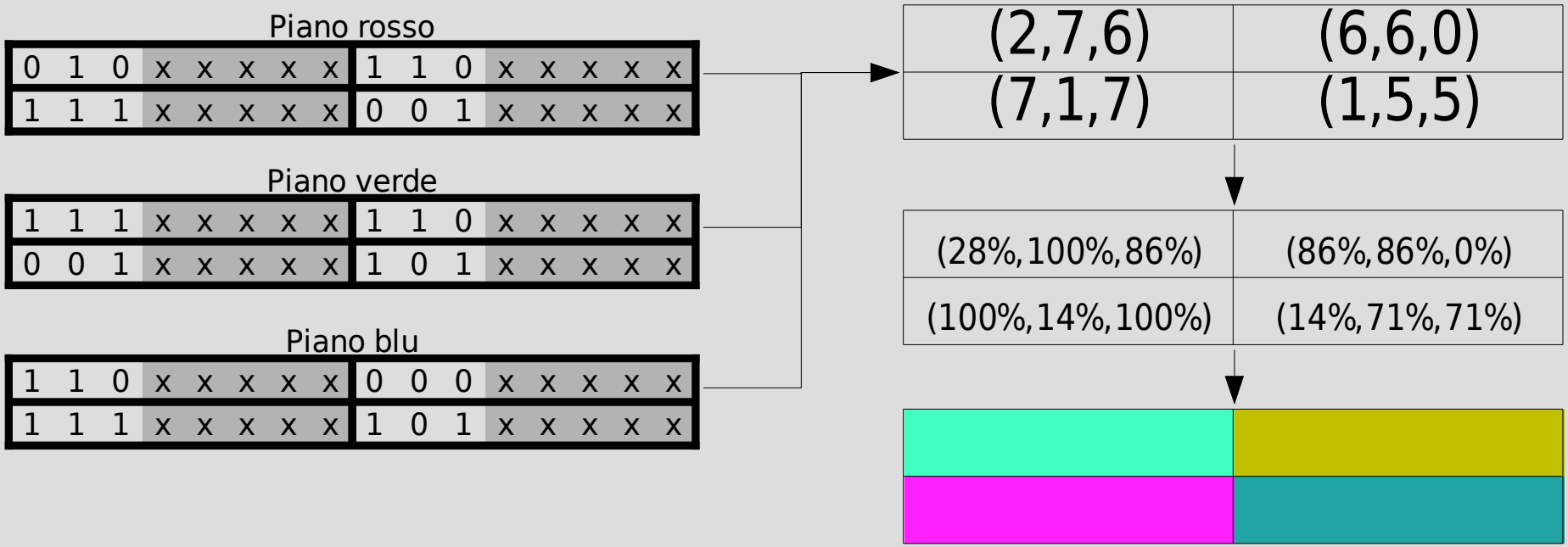
Organizzazione planare



- Byte che occupano la stessa posizione nei vari piani formano il codice colore dello stesso pixel
- Analogo ai bitplane, ma per byte anziché per bit

Organizzazione planare

- Questa organizzazione funziona molto bene quando il numero di bit di ogni componente è un multiplo di 8 (1 byte)
- Altrimenti, si spreca spazio! Es., con 3 bit:



Organizzazione planare

- Vantaggi dell'organizzazione planare:
  - ciascun piano è a sua volta un'immagine b/n del colore indicato, facile da interpretare
  - molti algoritmi di analisi funzionano benissimo su immagini planari
  - se il numero di bit per colore è un multiplo di 8, è efficiente in termini di memoria
  - la codifica RGB è direttamente leggibile in memoria

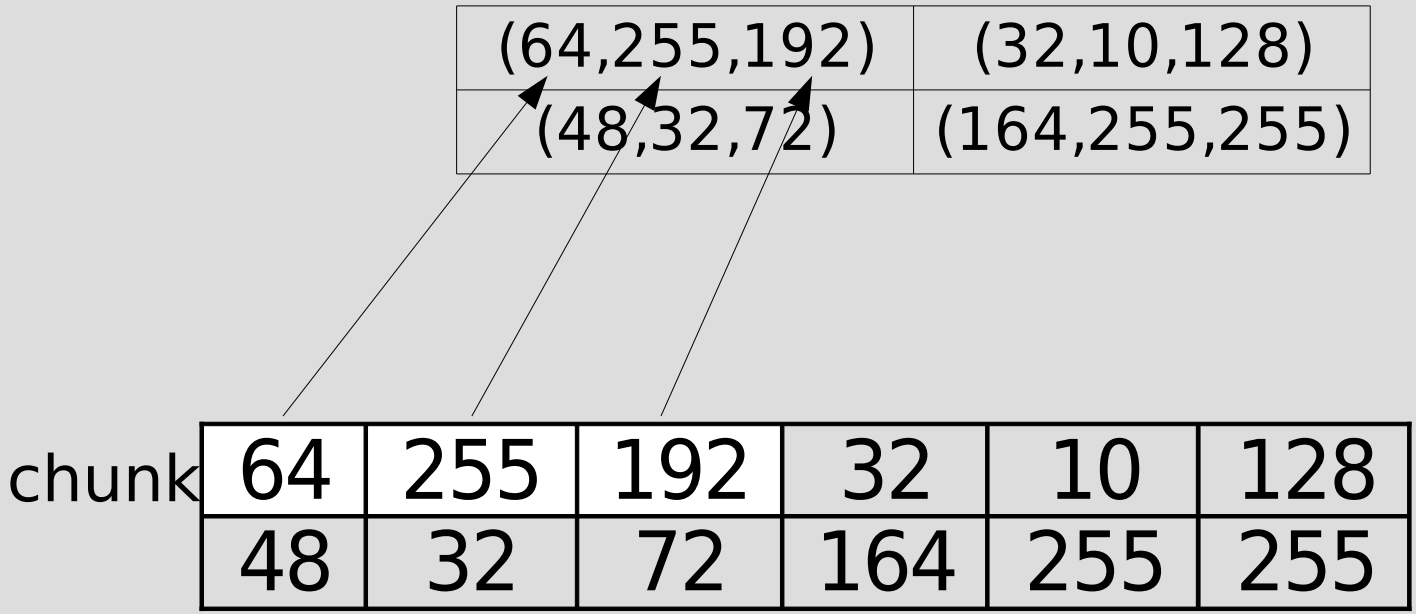
Organizzazione planare

- Svantaggi dell'organizzazione planare
  - spreco di spazio se il numero di bit per colore non è multiplo di 8
  - alcune operazioni sui colori non sono facili da implementare
- Non ha altri svantaggi importanti
- Infatti, è una organizzazione molto popolare!

Organizzazione a chunk

- L'organizzazione a **chunk** (blocchi) è la più diretta fra tutte
- Ogni pixel è rappresentato da una sequenza di bit (o byte) consecutivi (chunk) che descrive il suo codice colore
- Tutta l'immagine è memorizzata in un unico blocco contiguo di memoria

Organizzazione a chunk



Organizzazione a chunk

- Vantaggi dell'organizzazione a chunk
  - facile da spiegare!
  - nelle giuste condizioni, uso efficiente della memoria
  - fine dei vantaggi...

Organizzazione a chunk

- Svantaggi dell'organizzazione a chunk
  - se la profondità colore non è un multiplo di 8, anche qui si spreca memoria
  - organizzazione svantaggiosa per tutti gli algoritmi di analisi e compressione
  - praticamente mai supportata dall'hardware
  - comodo solo da leggere/scrivere da memoria di massa, infame per tutti gli altri usi

Organizzazioni indicizzate

- Tutte le organizzazioni di immagine che abbiamo visto fin qui sono del tipo **direct color**: la codifica di un pixel fornisce direttamente il suo colore
- Non sempre questa è la codifica più efficiente:
  - per esempio, un'immagine può usare relativamente pochi colori, ma non distribuiti uniformemente

Organizzazioni indicizzate

- In questi casi, il codice colore di un pixel (a sua volta ottenuto con uno qualunque delle organizzazioni precedenti) **non** fornisce la codifica RGB del pixel, ma un **indice** in un elenco di colori
- L'elenco può essere fisso, a variabile a seconda dell'immagine
  - quest'ultimo è il caso più frequente

Organizzazioni indicizzate

- Per esempio, se la mia immagine usa al più 256 colori distinti, è conveniente indicizzarla
- Ciascun pixel richiederà 8 bit (2^8=256 indici) anziché 24 (3x8=24 bit per colore)
- Ciascuno dei 256 colori indicizzati, richiederà 24 bit

Organizzazioni indicizzate

- Colore diretto:

Immagine

192	38	45	200	50	50
192	38	45	0	0	0
200	50	50	32	64	128

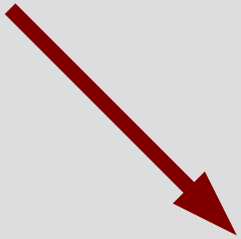
- Colore indicizzato:

Immagine

0	1
0	2
1	3
4	4

Tabella colori

0	192	38	45
1	200	50	50
2	0	0	0
3	32	64	128
4	192	192	192



(192,38,45)	(200,50,50)
(192,38,45)	(0,0,0)
(200,50,50)	(32,64,128)

Organizzazioni indicizzate

- Quando conviene l'una o l'altra?
- Dipende dal numero di colori *distinti* che siamo disposti a perdere
  - i colori persi possono però essere approssimati con altri molto simili
- Dipende anche dalla dimensione dell'immagine
  - Per immagini piccole e con tanti colori, la tabella colori può occupare troppo spazio (in proporzione al resto)

Esempio

- **Colore diretto**
  - 170x170x24
  - = **86.700** byte

16.777.216 colori

- **Colore indicizzato**
  - 170x170x8 + (immagine)
    256*24 (indice colori)
  - = **29.668** byte

256 colori

Altre organizzazioni

- Fra le molte altre organizzazioni possibili, alcune si incontrano con particolare frequenza:
  - bianco e nero
    - ogni pixel è un bit: 1=nero, 0=bianco (o viceversa)
    - usata per gestire testo come immagine (es., fax)
  - scala di grigi
    - ogni pixel è un valore da 0% (nero) a 100% (bianco)
    - tipicamente, si usano 8 bit, quindi 256 grigi
    - usata quando la destinazione finale è una stampa in bianco e nero (es., foto sui giornali)