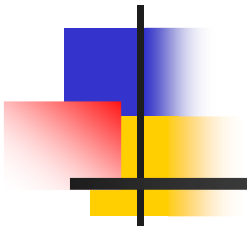
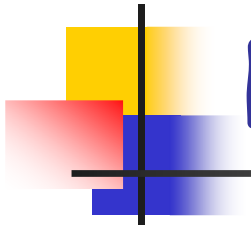


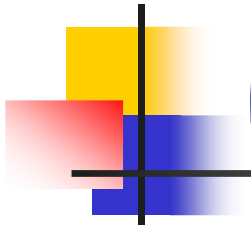
La codifica dell'informazione



Rappresentazione binaria

- Tutta l'informazione interna ad un computer è codificata con sequenze di due soli simboli : 0 e 1
- L'unità elementare di informazione si chiama *bit* da *'binary digit'*

Rappresentazione posizionale in base 10

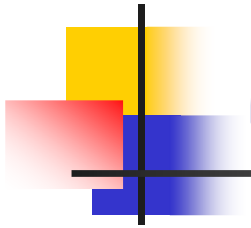


- Un numero (es. 5) può essere rappresentato in molti modi :

  cinque, *five*, *5*, V,

- Rappresentazioni diverse hanno proprietà diverse

  moltiplicare due numeri in notazione romana è molto più difficile che moltiplicare due numeri in notazione decimale ….



Base 10

- La rappresentazione di un numero intero in base 10 è una sequenza di cifre scelte fra 0 1 2 3 4 5 6 7 8 9:

  es: 23, 118, 4

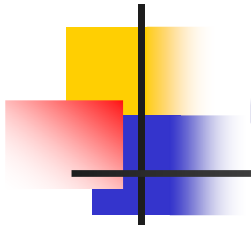
- Il valore di una rappresentazione $c_N \ldots c_0$ è dato da

  $c_N * 10^N + c_{N-1} * 10^{N-1} \ldots + c_1 * 10^1 + c_0 * 10^0$

  esempi :

  $23 = 2*10^1 + 3 * 10^0 = 20 + 3$

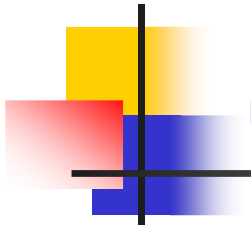
  $118 = 1*10^2 + 1*10^1 + 8 * 10^0 = 100 + 10 + 8$



Base 10

- Il massimo numero rappresentabile con N cifre è 99….9 (N volte 9, la cifra con valore più alto), pari a 10^N-1

  es: su tre cifre il massimo numero rappresentabile è 999 pari a $10^3-1 = 1000-1$



Base 10

- Quindi se voglio rappresentare K diverse configurazioni (cioè *0 1 2 …K-1*) mi servono almeno almeno *x* cifre dove 10^x è la più piccola potenza di 10 maggiore od uguale a K

  es : se voglio 25 configurazioni diverse mi servono almeno 2 cifre perché $10^2=100$ è la più piccola potenza di 10 maggiore di 25

Rappresentazione in base 2

- La rappresentazione di un numero intero in base 2 è una sequenza di cifre scelte fra 0 1 :

  es: 10, 110, 1

- Il valore di una rappresentazione $c_N..c_0$ è dato da

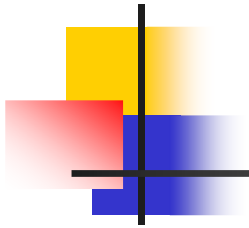
  $c_N * 2^N + c_{N-1} * 2^{N-1} ....+ c_1 * 2^1 + c_0 * 2^0$

  esempi :

  $10 = 1*2^1 + 0 *2^0 = 2$ (decimale)

  $110 = 1*2^2 + 1*2^1 + 0 * 2^0 = 4 + 2 + 0 = 6$ (decimale)

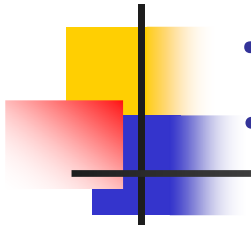
  $1 = 1 *2^0 = 1$ (decimale)



Base 2

- Quindi se voglio rappresentare K diverse configurazioni (cioè *0 1 2 …K-1*) mi servono almeno almeno *x* cifre dove 2^x è la più piccola potenza di 2 maggiore o uguale a K

  es: se voglio 25 configurazioni diverse mi servono almeno 5 cifre perché $2^5=32$ è la più piccola potenza di 2 maggiore di 25



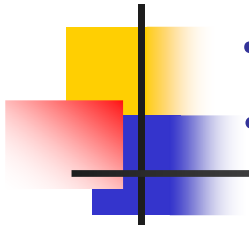
Il codice ASCII

Per codificare l'insieme dei caratteri dell'alfabeto e dei simboli speciali sono sufficienti 8 bit.

Con 8 bit si codificano infatti 256 simboli.

Il codice ASCII (American Standard Code for Information Interchange) utilizza un codice a 8 bit (1 byte)

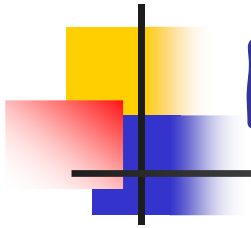
Il codice ASCII base usa 7 bit (non codifica i caratteri con gli accenti)



Il byte= unità di misura

Per rappresentare l'occupazione di memoria si usano i multipli del byte

- 1 KB (kilo byte = 2^{10} = 1024)
- 1 MB (Mega byte = 2^{20} = 1 KB*1024)
- 1 GB (Giga byte = 2^{30} = 1 MB*1024)
- 1 TB (Tera byte = 2^{40} = 1 GB*1024)



Rappresentazioni di immagini

- Le immagini sono un 'continuo' e non sono formate da sequenze di oggetti ben definiti come i numeri

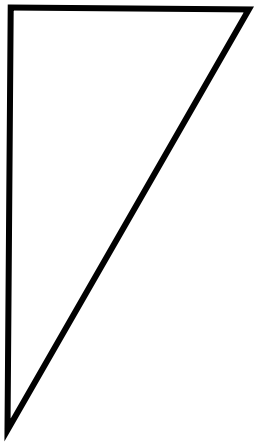
- Bisogna quindi prima 'discretizzarle' ovvero trasformarle in un insieme di parti distinte che possono essere codificate separatamente con sequenze di bit

Immagini bitmap (raster)

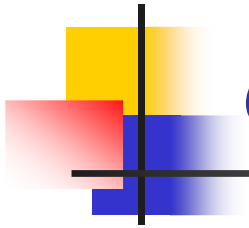
- Immagini 'bitmap' :

  L'immagine viene scomposta in una griglia di elementi detti *pixel* (da picture element)

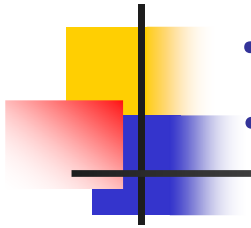
```
00000000000000000000000
00000000011111111000000
00000000010000010000000
00000000010000100000000
00000000010001000000000
00000000010010000000000
00000000010100000000000
00000000011000000000000
00000000010000000000000
```



Immagini a toni di grigio o colori



- Rappresentazioni dei pixel :
  - la rappresentazione in 'toni di grigio' : un byte per pixel, con 256 gradazioni di grigio per ogni punto, o più byte per pixel, per avere più gradazioni possibili
  - rappresentazione a colori RGB (red, green,blu): comunemente 3 byte (24 bit) per pixel che definiscono l'intensità di ciascun colore base. In questo modo ho circa 16 milioni di colori diversi definibili

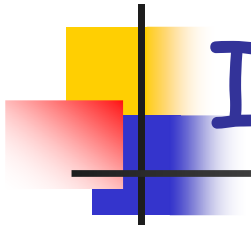


Immagini raster o vettoriali

Il formato bitmap viene anche definito formato raster, nome che in inglese indica l'insieme di linee orizzonali che la televisione traccia sullo schermo, punto dopo punto, al fine di riprodurre l'immagine.

Il formato bitmap è idoneo per le fotografie e per tutte le immagini composte da forme non regolari.

Viene rimpiazzato dal formato vettoriale nel disegno tecnico e architettonico, dovunque si debbano tracciare figure geometriche regolari o forme comunque complesse riconducibili a un insieme di triangoli e poligoni.



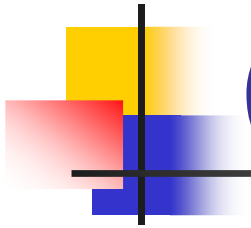
Immagini raster o vettoriali

Il vettore identifica il punto di partenza e di fine di una retta, la sua direzione, il suo spessore e il suo colore, ma non definisce ciascun punto della retta che viene costruito invece dal programma al momento della sua visualizzazione.

Tutte le immagini stampate su una rivista o visualizzate in televisione devono essere in formato bitmap.

Le immagini create da un programma di progettazione e di disegno tecnico sono quasi sempre vettoriali. Le immagini prodotte dai giochi sono in parte bitmap (gli sfondi e alcune superfici degli oggetti) e in parte vettoriali (gli oggetti che si muovono e che cambiano nello spazio).

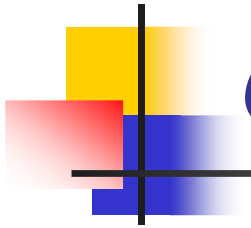
Occupazione in memoria (raster)



- 128 x 128 toni di grigio 128Kb
- 1024x768 colori (RGB) circa 18MB

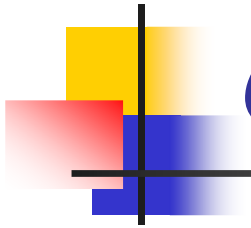
Compressione delle immagini raster

- Quindi si cerca di 'risparmiare' memoria :
  - con l'uso di una 'tavolozza' (*palette*) che contiene il sottoinsieme dei colori rappresentabili che compare in una foto
    - ogni pixel codifica un indice all'interno della tavolozza
  - con *tecniche di compressione* che non codificano ogni pixel in modo autonomo ma cercano di raggruppare i le aree che hanno caratteristiche comuni
- Formati più usati : TIFF (tagged image file format), GIF (graphics interchange format), JPEG (Joint photographers expert group)



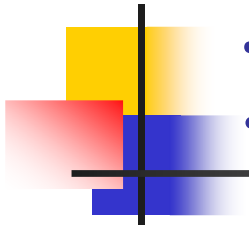
Compressione senza perdita

- Algoritmi *lossless* (senza perdita di informazione) : *operano un cambiamento di codifica dei dati che permette di diminuire il numero di bit necessari alla rappresentazione*



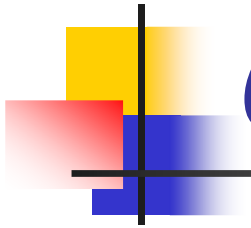
Compressione con perdita

- Algoritmi *lossy* (che perdono informazione)
  - gli algoritmi di compressione usati nei formati GIF e JPEG per immagini fisse sfruttano la caratteristica dell'occhio umano di *essere poco sensibile a lievi cambiamenti di colore in punti contigui*, e quindi eliminano questi lievi cambiamenti appiattendo il colore dell'immagine
  - generalmente è possibile specificare quanto siamo disposti a perdere attraverso alcuni parametri



Immagini video

- Il movimento è rappresentato già in modo discreto nei media : infatti con un numero abbastanza alto di fotogrammi fissi (24-30 al secondo) l'occhio umano percepisce il movimento come un continuo
  - potrei,in principio, codificare separatamente ogni fotogramma come immagine fissa, ma lo spazio di memoria richiesto sarebbe enorme (650 MB, un intero CD per un minuto di proiezione …)
  - sono stati quindi sviluppati metodi di codifica che economizzano, codificando solo le 'differenze' fra un fotogramma e l'altro (MPEG)



Glossario della lezione

Alcune definizioni usate in questa lezione sono state prese dal glossario dei termini informatici del Sistema Informativo Territoriale della provincia di Napoli.

http://sit.provincia.napoli.it/

consultatelo!