Corso di Informatica A.A. 2009-2010

Lezione 3

Codifica di Immagini

Non c'è un "quanto" naturale di informazione elementare come la cifra per i numeri o la lettera per i testi.

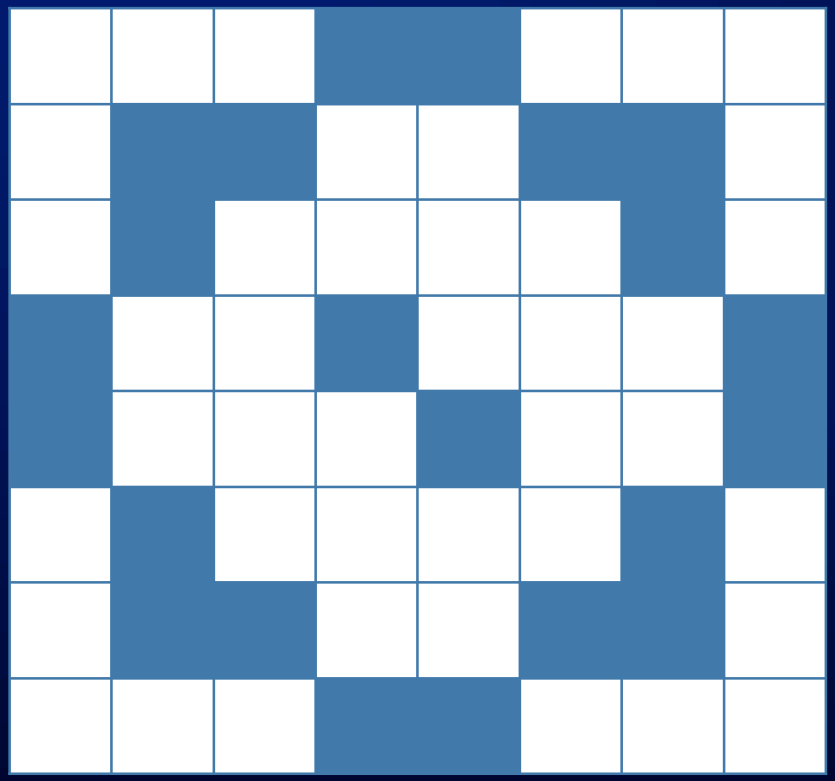
Si introduce allora un reticolo di punti detti *pixel (**pic**ture **el**ement)*

Ad ogni pixel viene poi associato un certo numero di bit, che indica l'intensità luminosa di ciascun colore primario, ovvero della combinazione RGB RedGreenBlue .

Una tipica codifica utilizza 8 bit per colore (=256 livelli di intensità) ovvero 24 bit/pixel. In questo modo si possono rappresentare $2^{24} \sim 16$ milioni di diverse tonalità di colore.

Il numero di bit/pixel impiegati viene chiamato profondità di colore dell'immagine.

Codifica di Immagini - Bitmap



0	0	0	1	1	0	0	0
0	1	1	0	0	1	1	0
0	1	0	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	0	0	1	0	0	1
0	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0
0	0	0	1	1	0	0	0

0001100001100110010000101001000110001001010000100110011000011000

Codifica di Immagini

- Problema :
  - la rappresentazione accurata di una immagine dipende
    - dal numero di pixel (*definizione*)
    - dalla codifica del pixel
  - … e richiede generalmente molta memoria, ad esempio :

tipo	*defin*	*numero colori*	*num. byte*
imm. televisiva	720x625	256	440 KB
SVGA	1024x768	65536	1.5 MB
foto	15000x10000	16milioni	430 MB

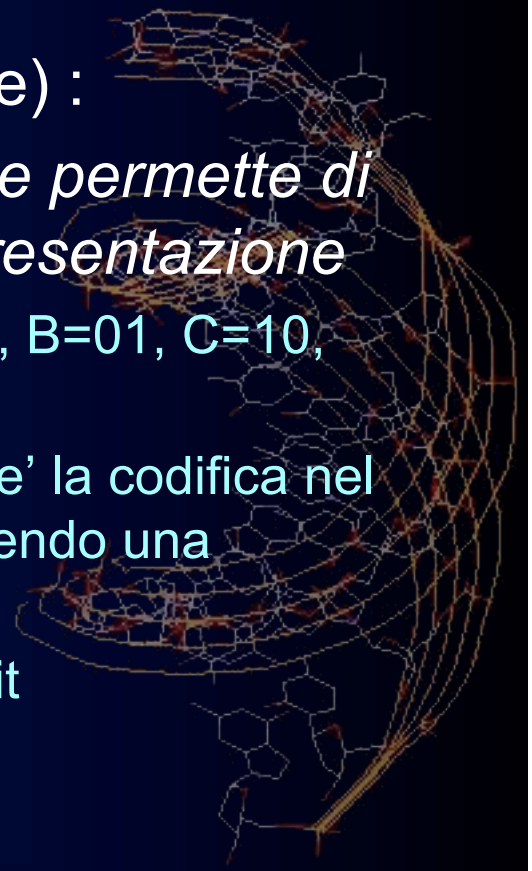
Compressione di Immagini

Algoritmi *lossless* (senza perdita di informazione) :

operano un cambiamento di codifica dei dati che permette di diminuire il numero di bit necessari alla rappresentazione

- esempio : sequenza di 1 milione di caratteri, A=00, B=01, C=10, D=11, totale 2 milioni di bit di codifica
- se A compare il 90% delle volte posso 'comprimere' la codifica nel seguente modo A=0, B=100, C=110, D=111 ottenendo una lunghezza di :

900 000 * 1 + 100 000 * 3 = 1 200 000 bit



Compressione di Immagini

Algoritmi *lossy* (che perdono informazione)

- generalmente sono specifici di un certo campo e sfruttano le caratteristiche degli oggetti da rappresentare per 'buttare via' informazione poco importanti
- 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

95.0%

Coefficienti di compressione

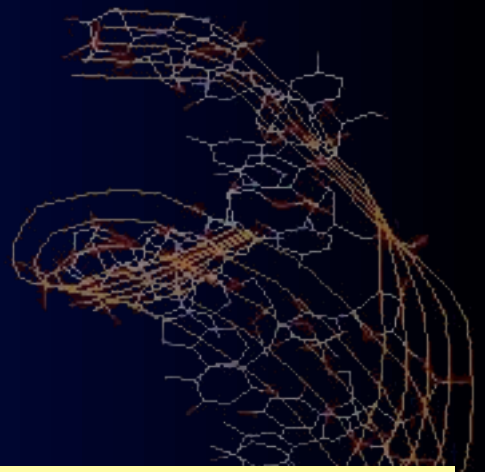
Coefficienti di compressione ottenibili	Lossless (non distruttiva)	Lossy (distruttiva)
Immagini naturali (foto digitali, scansioni)	1:1.5 - 1:2	1:1.5 - 1:30 senza perdita visibile di qualità 1:10 - 1:300 con perdita di qualità
Immagini artificiali (disegni, fumetti)	1:1.5 - 1:20	1:1.5 - 1:300

Codifica Immagini - Formati

- **RAW** *(matrice dei punti)*
- **TIFF** *Tagged Image File Format*
  (etichette che descrivono le propietà dei dati (24 bit, LZW))
- **GIF** *Graphics Iterchange Format*
  (Formato concepito per l'invio di immagini in rete, più immagini)
- **JPEG** *Joint Photographers Experts Group*
  (Per immagini fotografiche, trasparenza, interlace)
- **BMP** *Bitmap* (formato tipico di Windows)
- **PCX** *PC Paintbrush*
  (formato di questo popolare programma Windows)
- **PICT** Formato Macintosh (anche vettoriale)

.bmp 3841 Kb

.tiff 3842 Kb



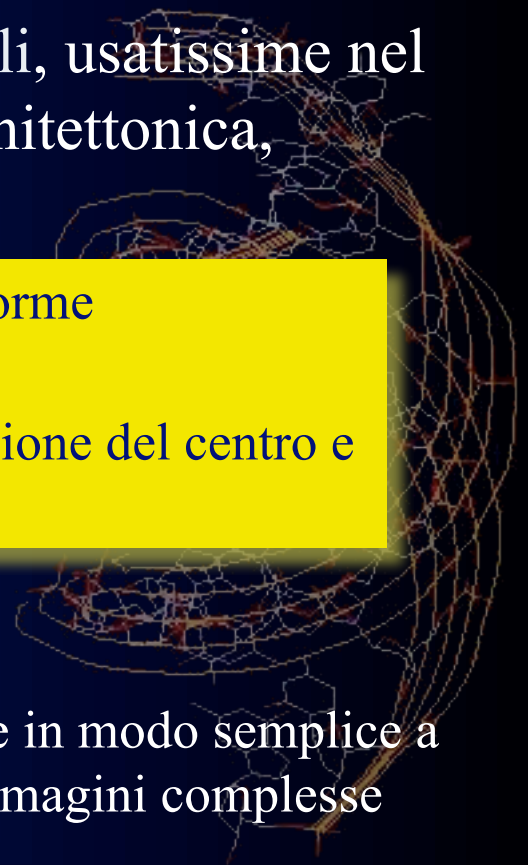
.jpeg 425 Kb

Codifica Immagini - Vettoriali

Un discorso a parte meritano le immagini vettoriali, usatissime nel campo della progettazione meccanica (CAD), architettonica, elettronica etc.

L'immagine, piuttosto che in punti, è scomposta in forme geometriche astratte come poligoni, cerchi etc.

Per descrivere una circonferenza bastano ad es. posizione del centro e raggio.



Vantaggi:

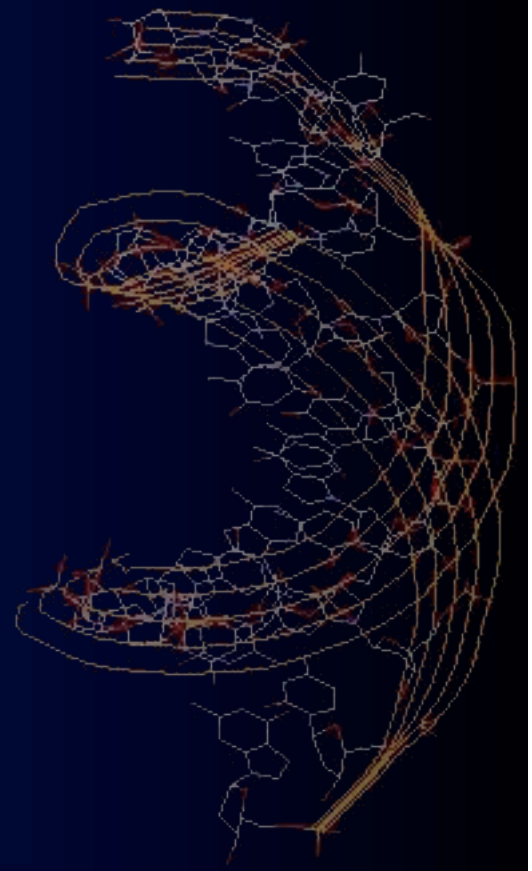
- indipendenza dalla risoluzione del dispositivo di visualizzazione/stampa
- dimensioni ridotte delle immagini
- Indipendenza dalla forma
- Scalabilità

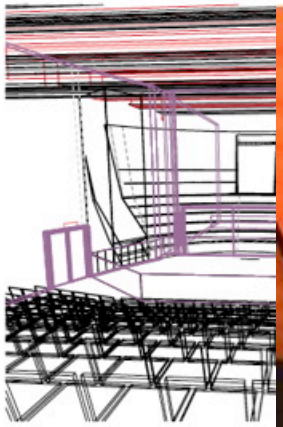
Svantaggi:

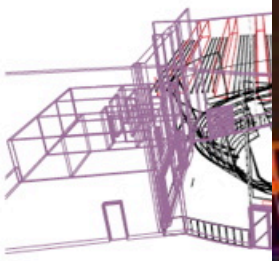
- non applicabile in modo semplice a fotografie e immagini complesse
- Cartoon-like

Immagini Vettoriali

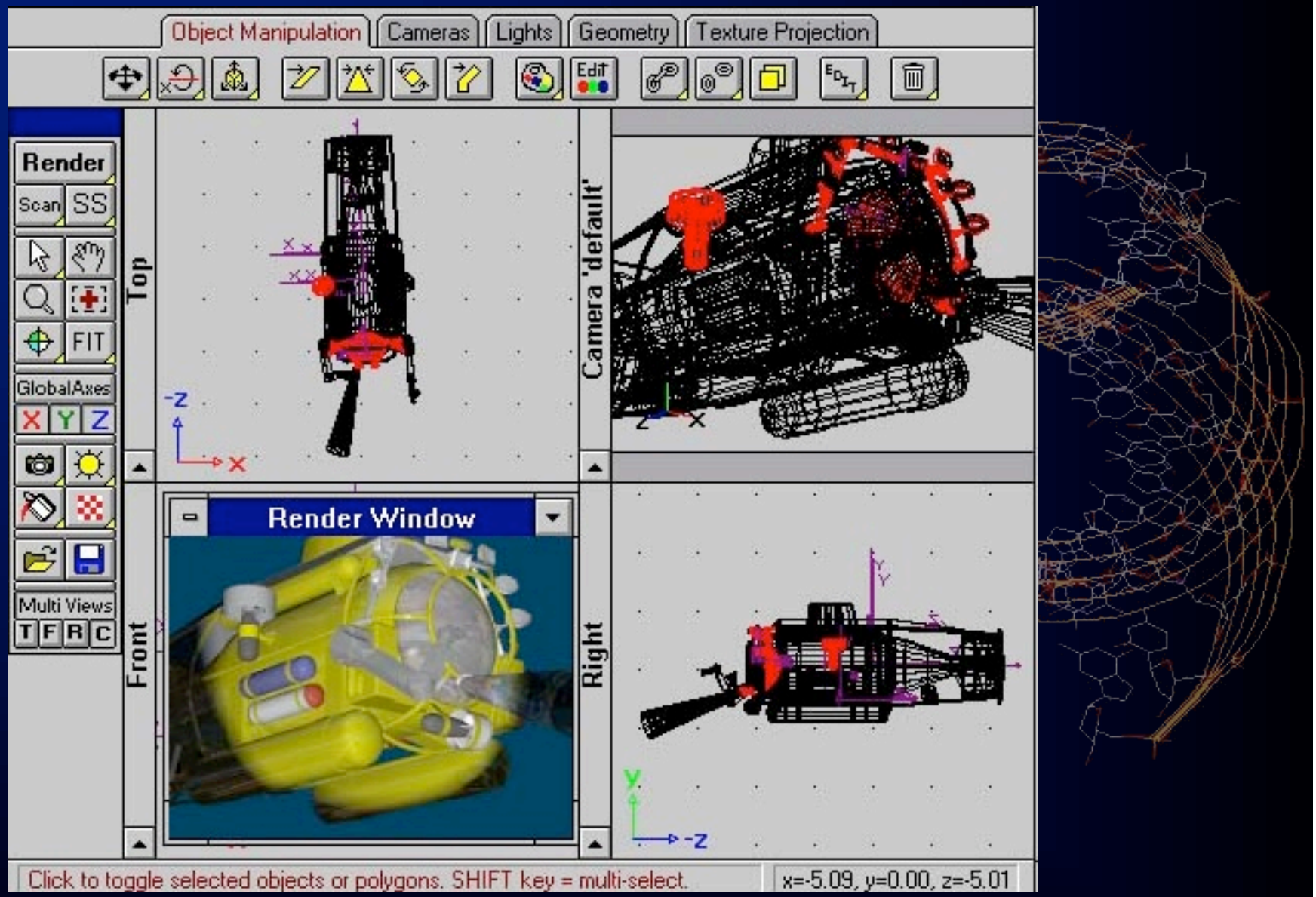
- **Documenti:**
  - Postcript (ps, eps, epsi)
  - PDF (Portable Document Format)
- CAD 3D
  - AI (Adobe Illustrator)
  - CDR (CorelDRAW)
  - CMX (Corel Exchange)
  - CGM Computer Graphics Metafile
  - DXF AutoCAD
  - HPGL AutoCAD
  - WMF Windows Metafile

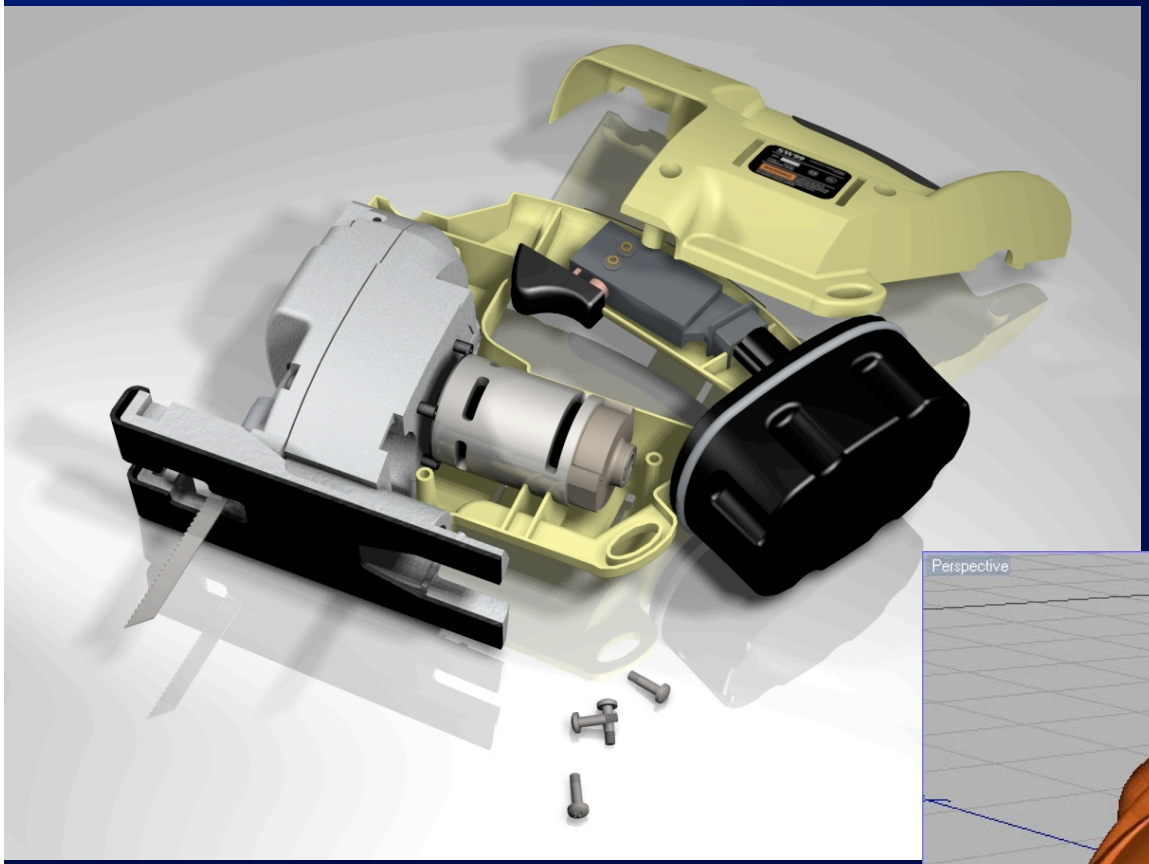


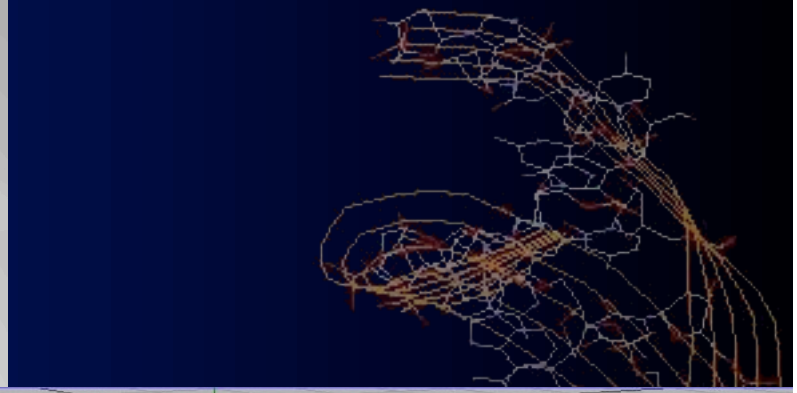


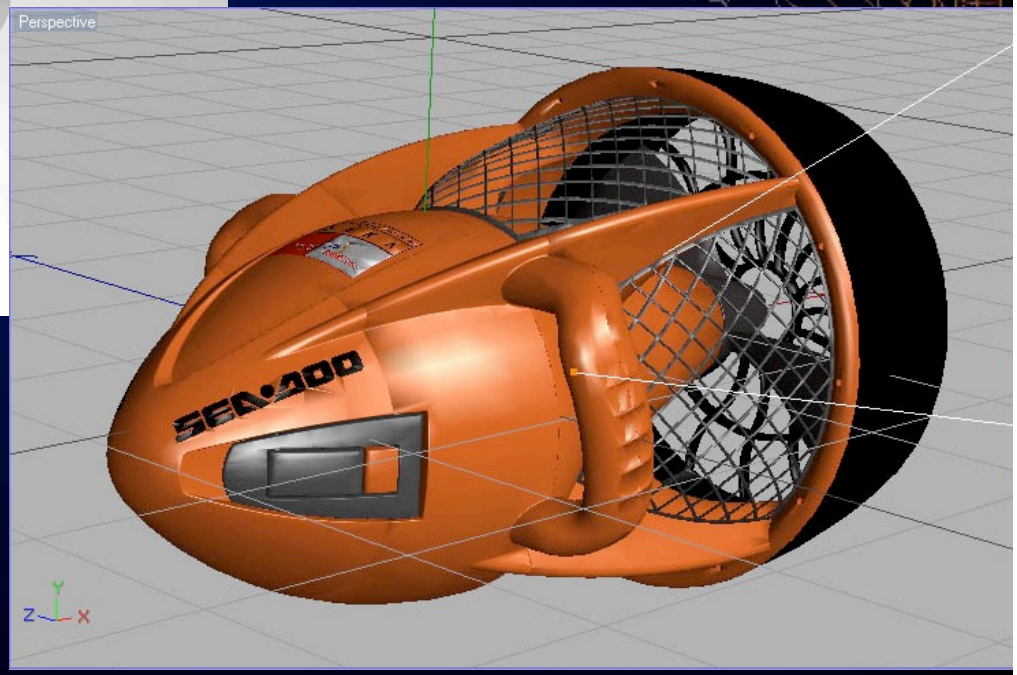


Rendering


Object Manipulation
Cameras
Lights
Geometry
Texture Projection
Render
Scan
SS
FIT
GlobalAxes
X
Y
Z
Multi Views
T
F
R
C
Top
Front
Right
Camera 'default'
Render Window
Click to toggle selected objects or polygons. SHIFT key = multi-select.
x=-5.09, y=0.00, z=-5.01






Perspective
SEA-DOO

Codifica Audio

Caratteristiche dell'audio (e dei segnali analogici)

Segnale analogico - Continuità

- nel tempo
- nell'ampiezza

Segnale Digitale - Discretizzazione

- nel tempo Campionamento
- Nelle ampiezze (Quantizzazione)

Codifica Audio

Campionamento dell'audio ad intervalli di tempo fissi

Codifica Audio

Campionamento dell'audio ad intervalli di tempo fissi

Codifica Audio

- Memorizzazione

  N_{bit}=Durata • f_c • bits/campione

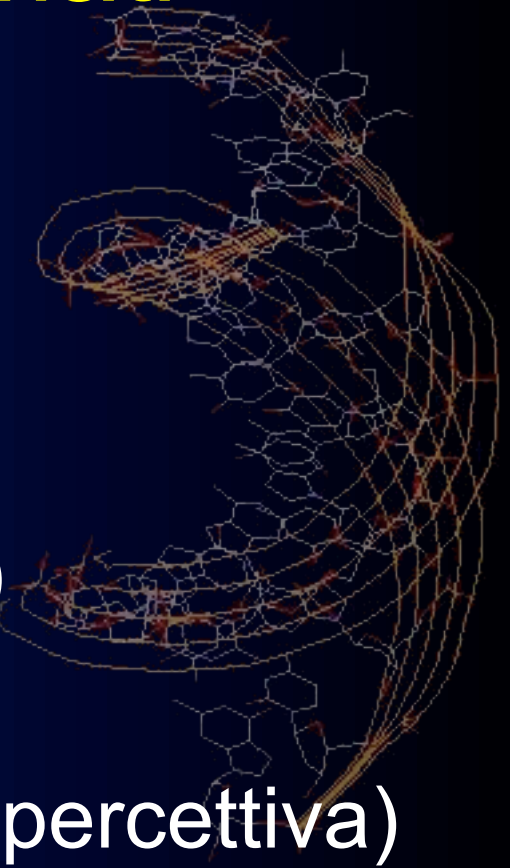
- Playback

  Richiede un certo bit rate bit/sec

- Esempi

	Telefonia	CD Audio
Segnale	Voce	Musica
Canali	1 (mono)	2 (stereo)
Banda	300 Hz-3.4 kHz	20 Hz-20 kHz
f_c	8 kHz	44.1 kHz
bits/campione	8	16
Bitrate	64kbps	1.4 Mbps

Codifica Audio - Formati



- PCM (Pulse Code Modulation)
- WAV (Wave)
- AAC (Advanced Audio Coding)
- AIFF (Audio Interchange Format)
- RA (RealAudio)
- MP3 (MPEG-1 Layer 3) Codifica percettiva)

Codifica Audio - MP3

- Non è un semplice algoritmo di compressione dati
- Codifica Percettiva basata sull'effetto mascheramento
- Diffusissimo per l'alto fattore di compressione

120 mins = 700 Mbytes (WAV) = 50 Mbytes (mp3)

Codifica Filmati

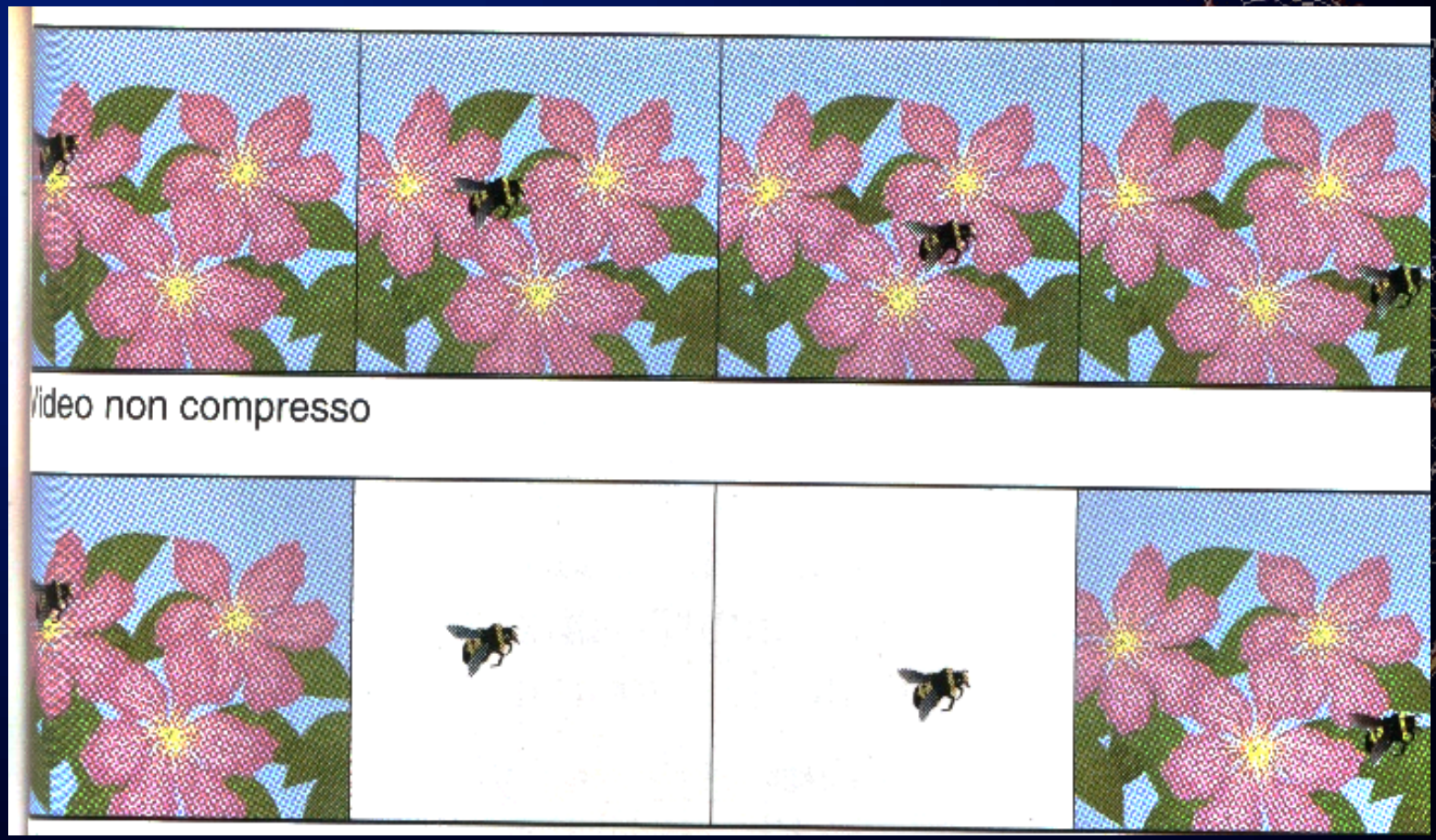
- Tecnica di decodifica
  - Campionamento di immagini

    (fps =frames per second)
  - La persistenza sulla retina dà la continuità
- Dimensioni tipiche
  - Video signal: 25 frames per second => 60 MB/s
  - VHS video quality:
    - 352x288 pixels per frame, 12 bits per pixel => 30.4 Mbits/s
  - Television quality: 704x576 pixels per frame,
    - 12 bits per pixel =>121.7 Mbits/s

Codifica Filmati

La compressione è d'obbligo su supporti digitali

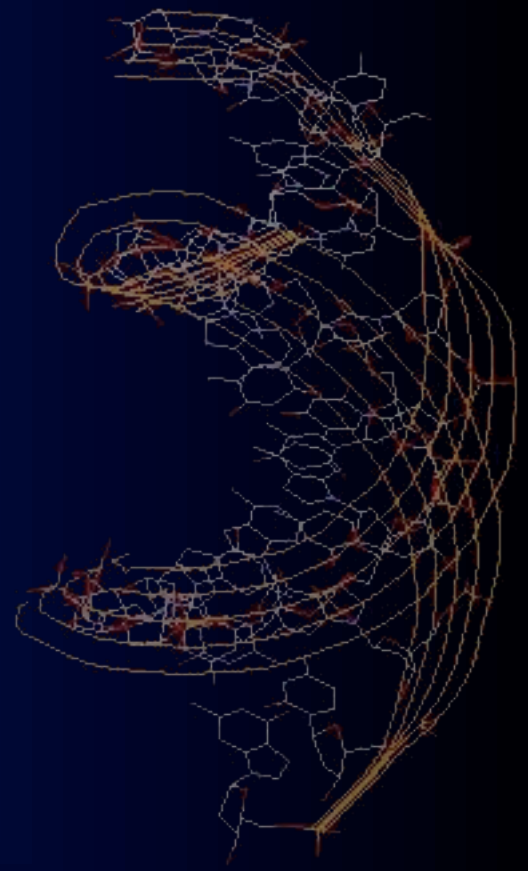
Video CODEC



Video compresso con tecnica intraframe

Codifica Filmati - CODEC

- AVI (Audio Video Interleave)
- MOV (QuickTime Format MacOSX)
- WMV (Windows Media PLayer)
- REALVideo (Real Network)
- MPEG-1/2/3/4 (IEEE Standard)
- XviD, DiVX, 3ivx



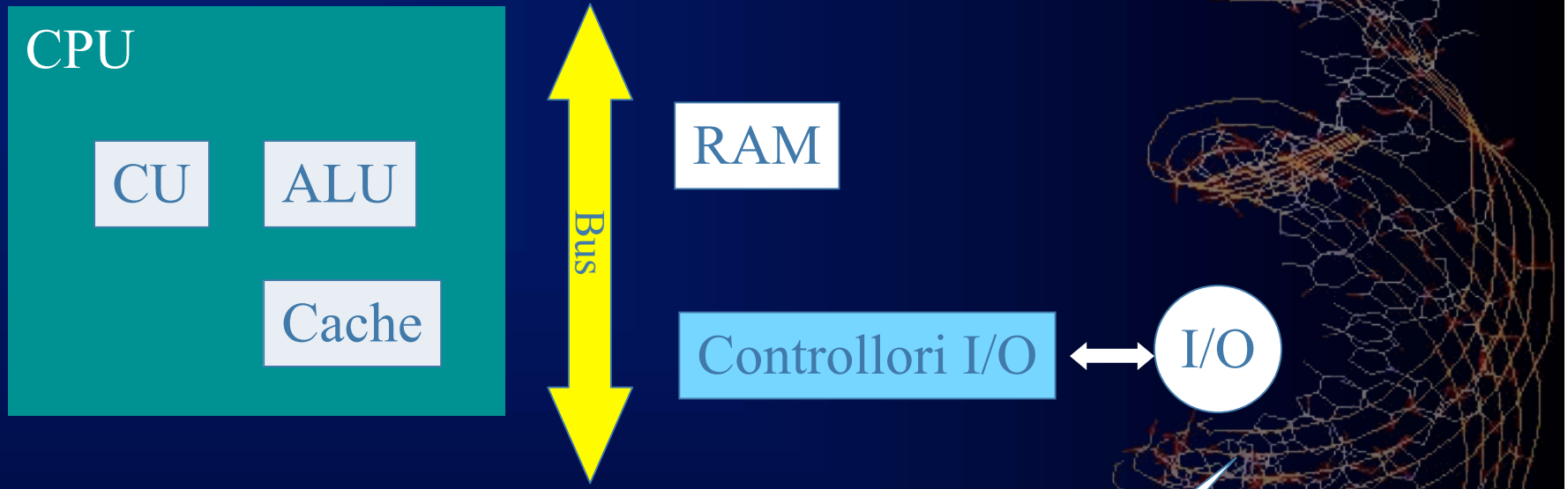
Conclusioni

Le tecniche di compressione dei dati, unite a studi di psicoacustica hanno permesso di realizzare rappresentazioni di filmati e di sequenze audio di dimensioni accettabili, dell'ordine di qualche MB. Questo ha aperto la strada all'utilizzo della multimedialità nell'informatica.

Funzioni del Sistema Operativo

- Concetto di OS
  - Gestione dei processi
  - Gestione della memoria
  - Gestione dell'I/O
  - Gestione del file system

La macchina nuda



010101011
101110001
101011001

La macchina virtuale

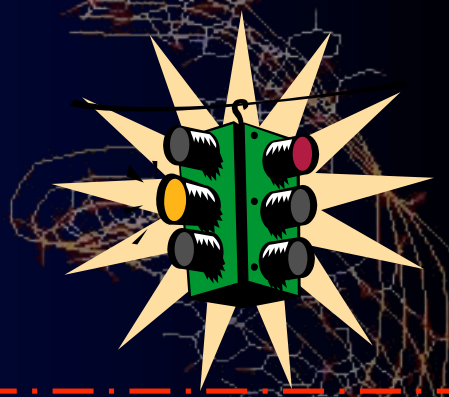
Tra la macchina nuda e l'utente si inserisce, con funzione di interfaccia, la macchina virtuale.

Si tratta di un insieme di programmi (software), che da una parte comunicano con l'hardware per gestire la macchina reale e dall'altra forniscono all'utente un ambiente virtuale i cui servizi fondamentali sono:

- Accesso semplificato alla realizzazione ed esecuzione di programmi
- Accesso e gestione delle risorse di I/O "ad alto livello"
- Accesso a dati e informazioni e protezione degli stessi

Il sistema operativo

L'OS gestisce le risorse hardware e controlla l'esecuzione di tutti gli altri programmi.



Servizi richiesti dagli utenti
(programmi, operazioni di I/O etc.)

Livello utente

Livello kernel (privilegiato)

Interfaccia verso l'esterno (system call)

Kernel

- Gestione processi
- Gestione memoria
- Gestione I/O
- Gestione file system
- Traduttori

Hardware

Protezione e condivisione

Oltre a costituire una macchina virtuale un sistema operativo (OS) deve essere in grado di condividere le risorse fra le applicazioni, nel contempo proteggendo ciascuna dai danni potenzialmente arrecati dalle altre.

Esempi:

Memoria non protetta: un'applicazione può invadere lo spazio usato da un'altra o addirittura dallo stesso OS

I/O non protetto: un'applicazione può "appropriarsi" di un dispositivo di I/O a tempo indeterminato

Programmi e Processi

Un *programma eseguibile* è un insieme di istruzioni ben ordinato in linguaggio macchina che può essere caricato in memoria : è un oggetto *statico*

Un processo può essere definito, grosso modo, come un programma in esecuzione, ed è quindi per natura un oggetto *dinamico*.

Un processo è descritto non solo dal programma, ma anche da tutte le informazioni (dinamiche) sullo stato di esecuzione (program counter, registri, dati etc.) dello stesso.

Monotasking e Multitasking

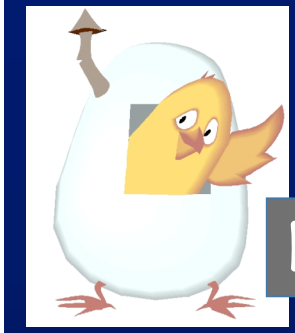
I primi OS erano Monotasking: un solo processo poteva essere in esecuzione in un dato momento.

Questo sistema è molto inefficiente nell'utilizzo delle risorse.

Per esempio un processo *I/O bound,* che spende la maggior parte del tempo in attesa di risposta dall'I/O blocca inutilmente la CPU, mentre un processo di tipo *CPU bound* può bloccare per lungo tempo l'interazione con l'utente "congelando" la macchina.

Multitasking: il controllo della CPU viene passato a turno da un processo all'altro. L'operazione di passaggio prende il nome di context switching.

Stati di un processo

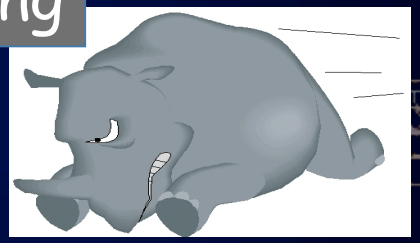


New

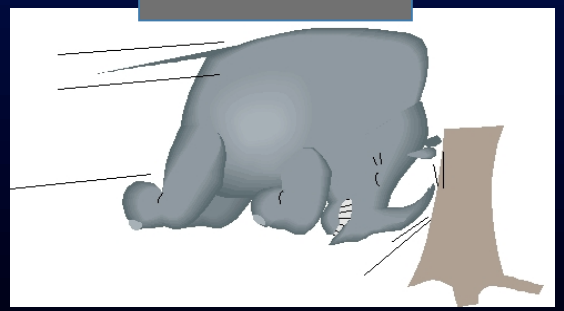
Ready

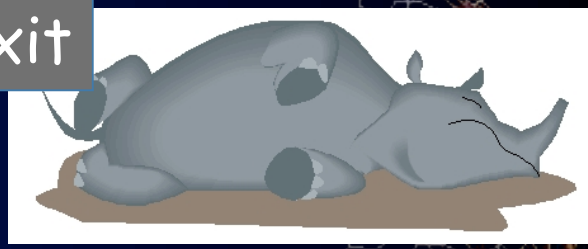
Running

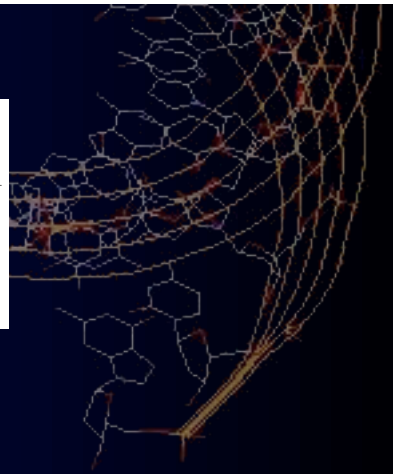
Exit



Blocked







Scheduler

Interrupt

Lo scheduler della CPU

La sua funzione, come visto, è di selezionare uno dei processi che sono nello stato "Ready" e dargli il controllo della CPU.

Può far partire un nuovo processo quando

1. Un processo si blocca (running => blocked)
2. Una time slice finisce (running => ready)
3. Un processo si sblocca (blocked => ready)
4. Un processo termina (running => exit)

Time sharing

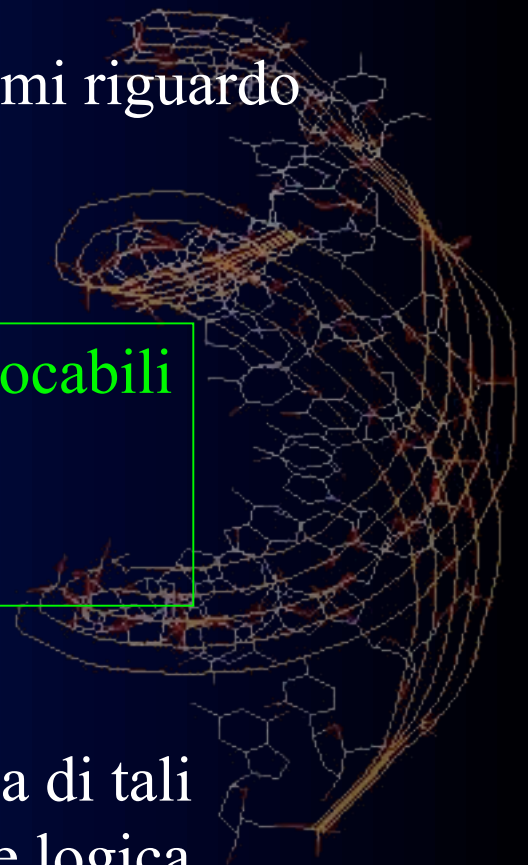
Esistono varie tecniche di assegnazione della CPU:

First In First Out, Shortest Job First, Round Robin…

Gestione della memoria

Il sistema multitasking pone una serie di problemi riguardo la gestione della memoria:

1. Necessità di utilizzare programmi rilocabili
2. Protezione delle risorse
3. Condivisione delle risorse



Il sistema di gestione della memoria si occupa di tali questioni, nonché dell'organizzazione fisica e logica della memoria.

Swapping

Un processore a 32 bit può indirizzare 2^{32}=4 GB di RAM, superiore alla memoria fisica della massima parte dei computer.

La struttura a “memoria virtuale” suggerisce di utilizzare questa possibilità per utilizzare parte della memoria di massa (hard disk) per “simulare” memoria e contenere processi inattivi.

Questa opportunità va però usata con moderazione perché la lettura e scrittura su/da disco (swapping) di pagine di memoria è molto lenta e rischia di paralizzare il computer. In UNIX lo swap viene abilitato solo se la RAM libera è al di sotto di una soglia prestabilita.


Linguaggio C

Operatori aritmetici, relazionali e logici

Operatori aritmetici:

+ - * / %

Operatore modulo:

x%y è il resto della divisione di x per y e quindi vale zero se x è multiplo di y

Operatori relazionali:

> >= < <= == !=

Operatori di uguaglianza e disuguaglianza:

x = = y è vera se x è uguale a y

x != y è vera se x è diverso da y

Operatori logici:

&& ||

AND logico

OR logico

Operatori di incremento e decremento

Spesso in un programma occorre incrementare o decrementare di uno una variabile. Il C dispone di particolari operatori dedicati a ciò, che sono:

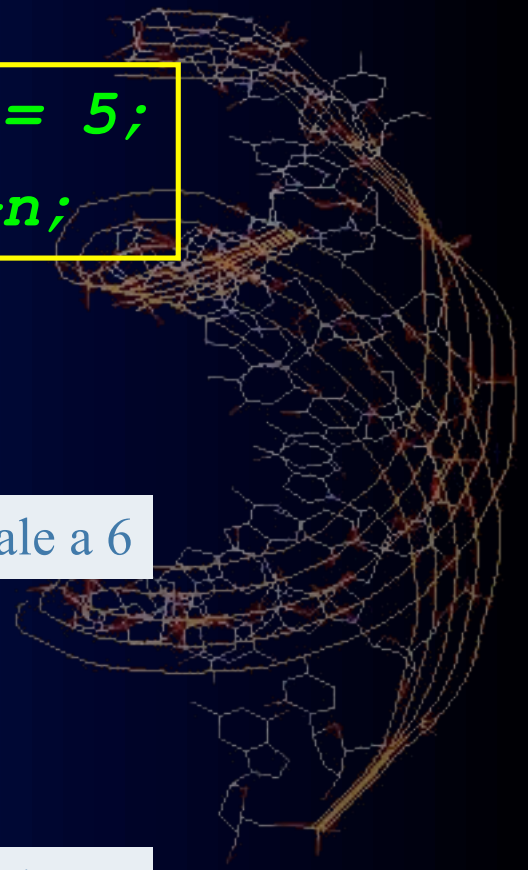
Se usati rispettivamente prima o dopo di una variabile ne determinano il preincremento (variabile prima incrementata e poi utilizzata) o il postincremento (variabile prima utilizzata e poi incrementata).

Esempi

```
n = 5;
n++;
```

→ Da ora in poi n è uguale a 6 ←

```
n = 5;
++n;
```



```
n = 5;
k = n++;
```

→ Da ora in poi k è uguale a 5 ed n è uguale a 6

```
n = 5;
k = ++n;
```

→ Da ora in poi k è uguale a 6 ed n è uguale a 6

Operatori di assegnamento

Un'altra particolare classe di operatori del C permette di scrivere in forma compatta le operazioni in cui una variabile viene modificata a partire da sé stessa. Ad esempio:

`x += 2`	è equivalente a	`x = x + 2`
`x -=3`	è equivalente a	`x = x - 3`
`y *= x`	è equivalente a	`y = y*x`
`y *= x + 1`	è equivalente a	`y = y*(x + 1)`
`z %= x`	è equivalente a	`z = z%x`

```c
/* Illustrates the modulus operator. */
/* Inputs a number of seconds, and converts to hours, */
/* minutes, and seconds. */

#include <stdio.h>

/* Define constants */

#define SECS_PER_MIN 60
#define SECS_PER_HOUR 3600

unsigned seconds, minutes, hours, secs_left, mins_left;

int main()
{
    /* Input the number of seconds */

    printf("Enter number of seconds (< 65000): ");
    scanf("%d", &seconds);

    hours = seconds / SECS_PER_HOUR;
    minutes = seconds / SECS_PER_MIN;
    mins_left = minutes % SECS_PER_MIN;
    secs_left = seconds % SECS_PER_MIN;

    printf("%u seconds is equal to ", seconds);
    printf("%u h, %u m, and %u s\n", hours, mins_left, secs_left);

    return 0;
}
```

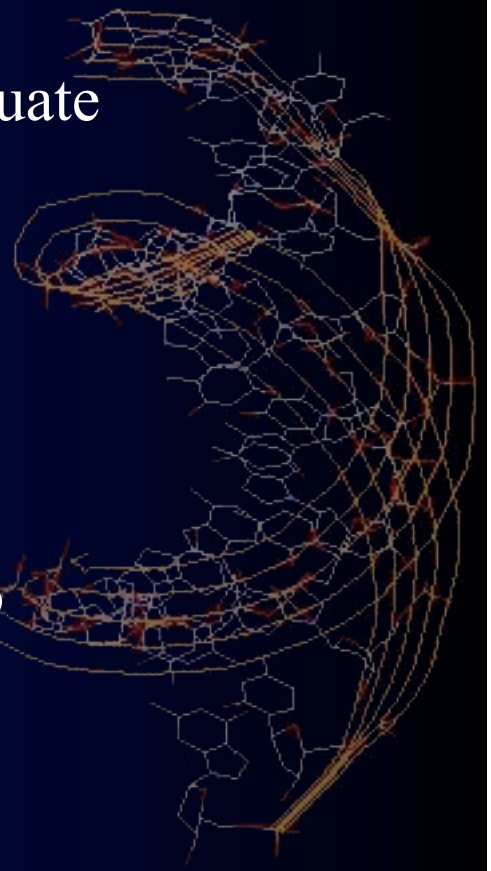
Regole di precedenza

In ogni “espressione” in C le operazioni sono effettuate secondo il seguente ordine

1. Incremento e decremento
2. Moltiplicazione, divisione e modulo
3. Addizione e sottrazione

Se un’espressione contiene più di un operatore con lo stesso diritto di precedenza allora le operazioni sono eseguite da sinistra a destra così come compaiono nell’espressione stessa

Parentesi



DO & DON'T

Utilizzare le parentesi per rendere chiaro l'ordine di esecuzione

Ricordare che VERO corrisponde a 1 e FALSO a 0 quando si Lavora con operatori relazionali

Non confondere ==, operatore relazionale con = operatore di assegnamento

```c
/* Demonstrates the evaluation of relational expressions */

#include <stdio.h>

int a;

int main()
{
    a = (5 == 5);               /* Evaluates to 1 */
    printf("\na = (5 == 5)\na = %d", a);

    a = (5 != 5);               /* Evaluates to 0 */
    printf("\na = (5 != 5)\na = %d", a);

    a = (12 == 12) + (5 != 1); /* Evaluates to 1 + 1 */
    printf("\na = (12 == 12) + (5 != 1)\na = %d\n", a);
    return 0;
}
```