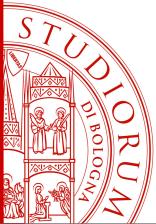


Hardware e Software per Grafica Interattiva





SOMMARIO

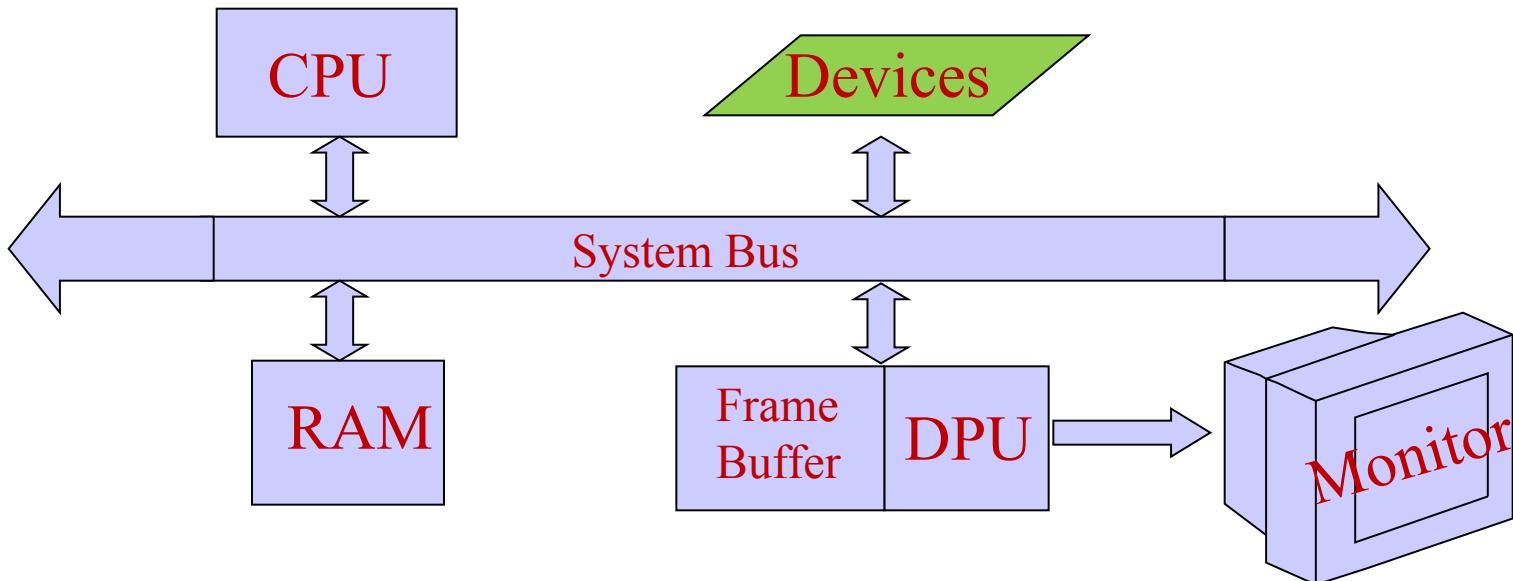
Architettura Grafica (dagli anni '80 ad oggi)

- Display, Frame Buffer, DPU
- Pipeline Grafica (transform & lighting)
- Schede grafiche e GPU
- CPU, GPU, GPU programmabile
- Prestazione delle GPU

Dispositivi Grafici Interattivi

- Mouse
- Il problema dell'interattività
- Ciclo di Polling
- Coda degli Eventi
- Programmazione Event-Driven

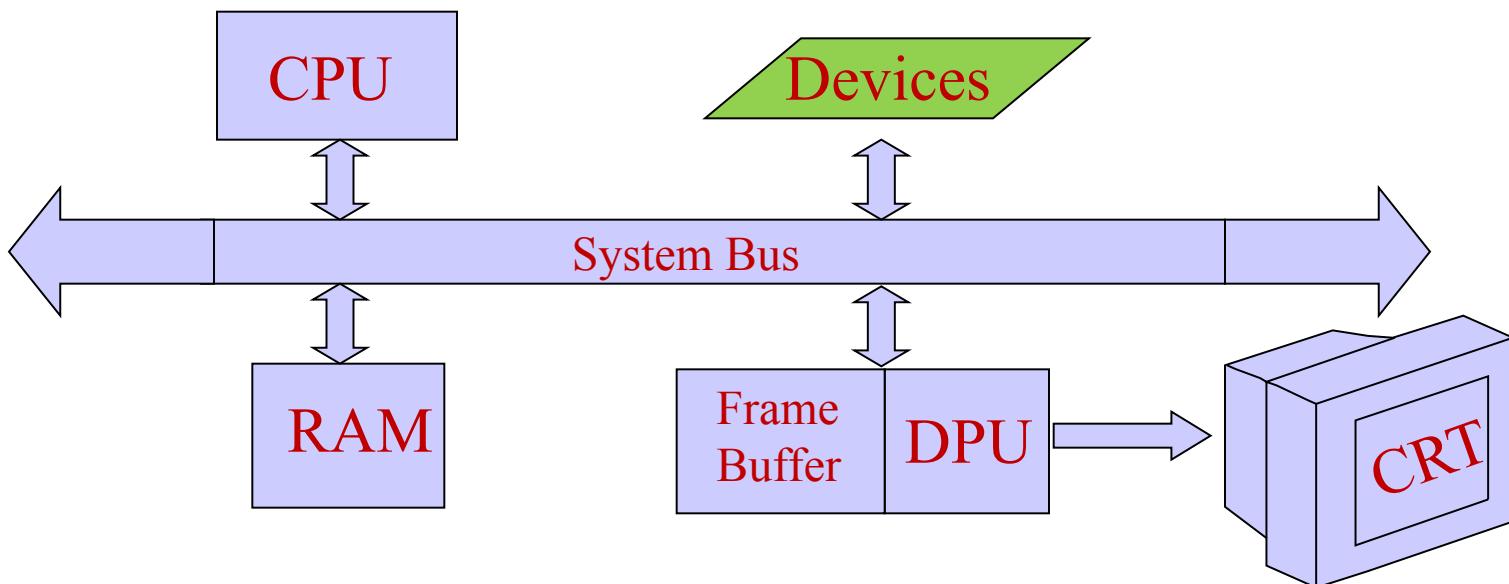
Architettura Grafica (anni '80)



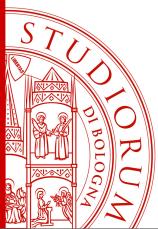
- Un disegno (o immagine) su uno schermo grafico è generato punto per punto (o pixel per pixel)
- disegnare consiste nel “settare” l’intensità luminosa di ogni pixel

pixel = picture element

Componenti Hardware



- **CRT:** Monitor su cui visualizzare l'immagine
- **Frame Buffer:** Memoria RAM in cui si memorizza l'intensità o colore di ogni pixel
- **DPU:** Processore grafico che scandisce il contenuto del Frame Buffer e produce un segnale per il Monitor



Monitor e Display negli anni

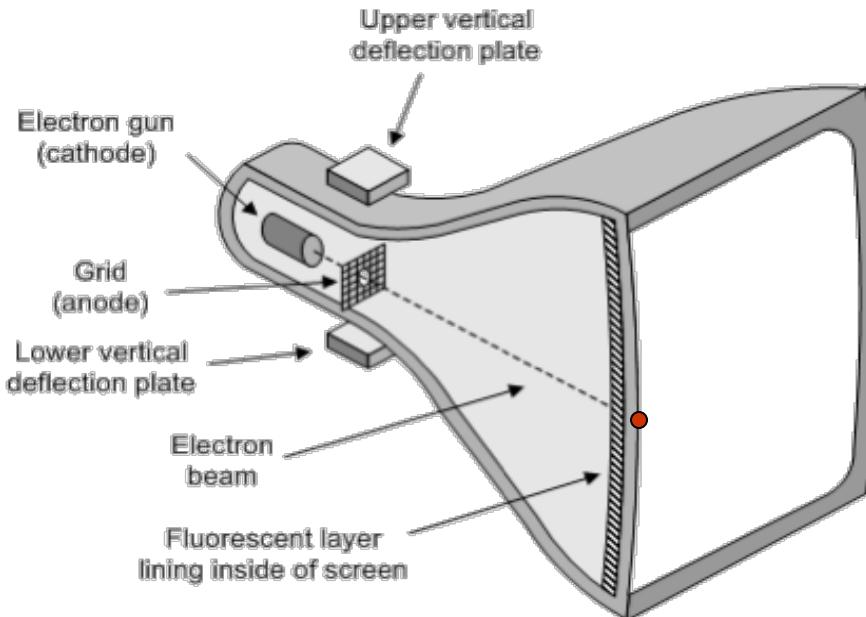
Il componente principale di un monitor è il display, cioè il dispositivo elettronico per la visualizzazione. In base alla tecnologia usata si hanno le seguenti tipologie di display:

- CRT (Cathod Ray Tube)
- LCD (Liquid Crystal Display)
- LED (OLED acronimo di Organic Light Emitting Diode)
- PDP (Plasma Display Panel)
- 3D
- ...

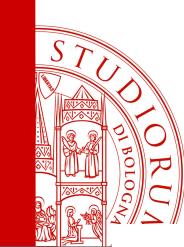




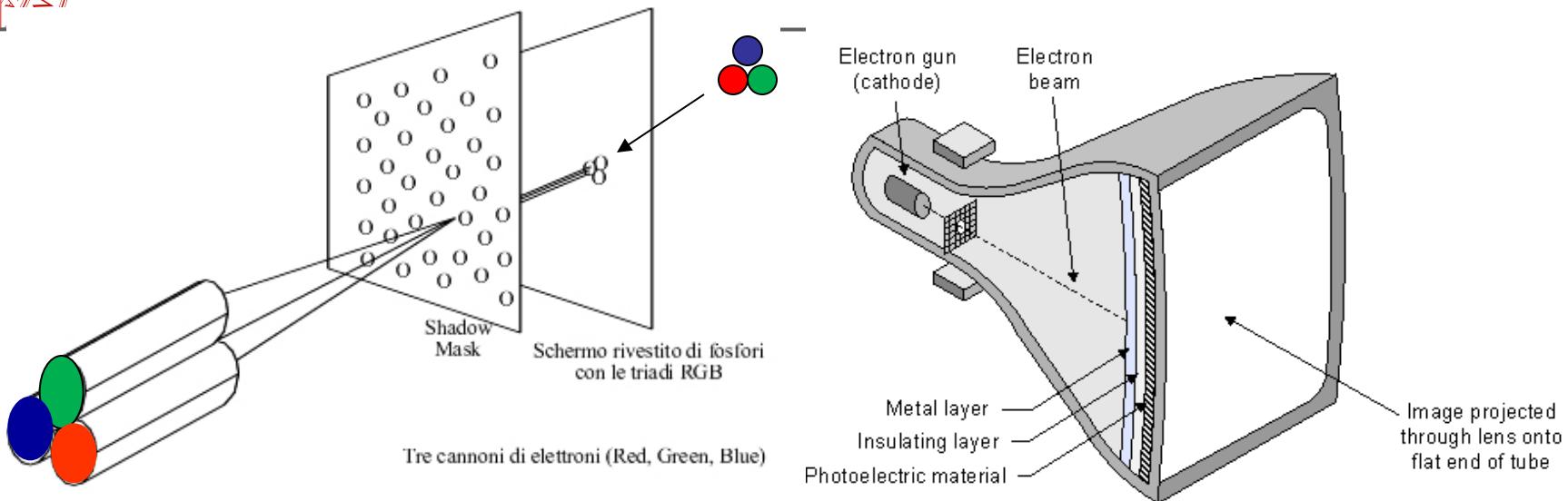
Display: CRT (Cathod Ray Tube)



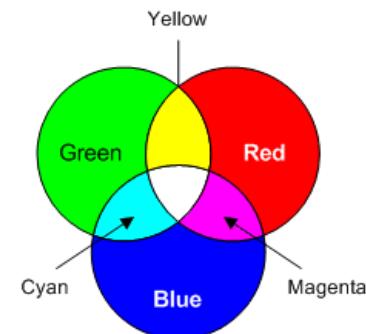
- Il voltaggio applicato al “cannone di elettroni” determina il numero di elettroni emessi
- L'intensità di un punto illuminato dipende dal numero di elettroni del pennello
- La luce emessa dal fosforo dura solo una piccola frazione di secondo
- Si deve rinfrescare l'immagine circa 60 volte al secondo per evitare l'effetto tremolio (FLICKERING); CRT a rinfresco.



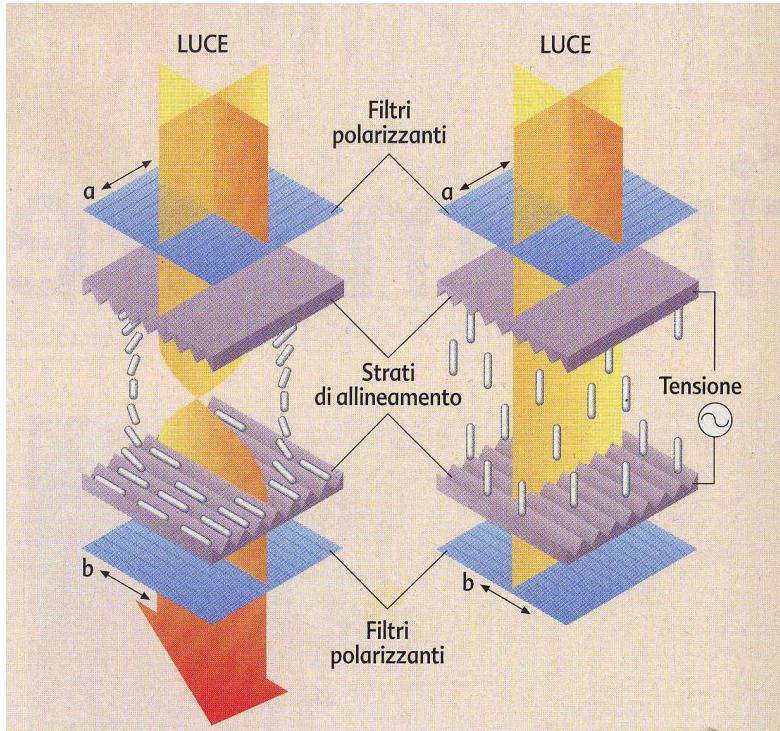
CRT a Colori (Shadow Mask)



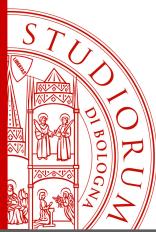
- La Shadow Mask consiste in una sottilissima lastra metallica con piccolissimi fori, montata vicino allo strato di fosforo;
- La Shadow Mask è perfettamente posizionata, così che i tre pennelli di elettroni (uno per il **rosso**, uno per il **verde** ed uno per il **blu**), possano colpire solo un tipo di fosforo;
- In figura è presentato un Delta CRT in quanto i punti di fosforo sono disposti a triangolo (Δ).



LCD (Display a cristalli liquidi)

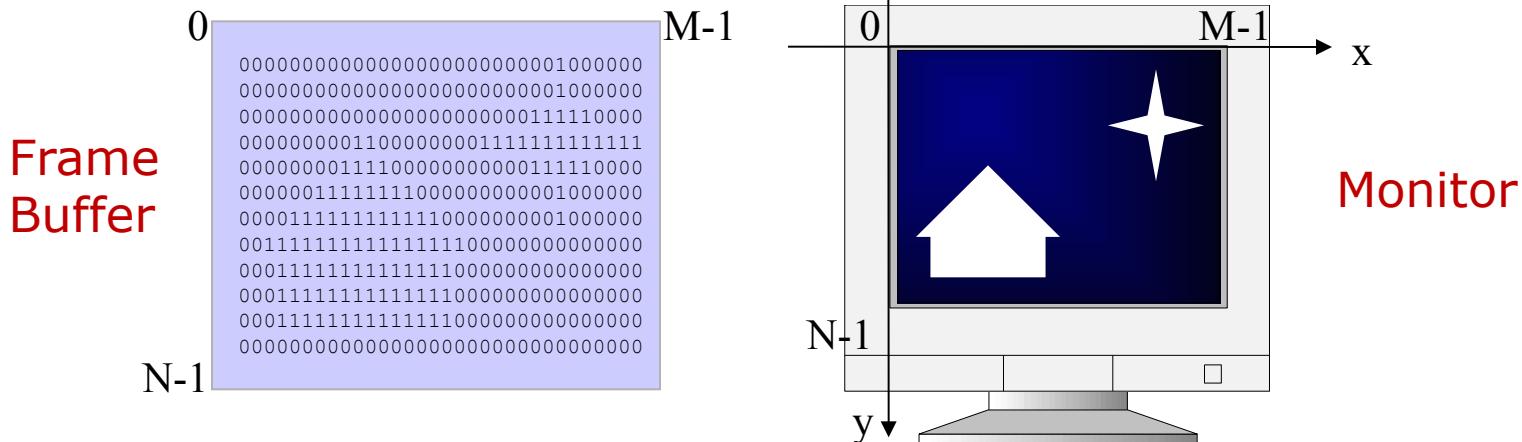


- Indirizzamento ad array bidimensionale dei punti con voltaggi $-V$ sulla griglia orizzontale e $+V$ su quella verticale
- Agire su un pixel corrisponde ad ottenere un punto spento
- Rinfresco a scanline o TFT (Thin Film Transistor) per display a matrice attiva
- Gestione colore con filtri colorati RGB



Frame Buffer

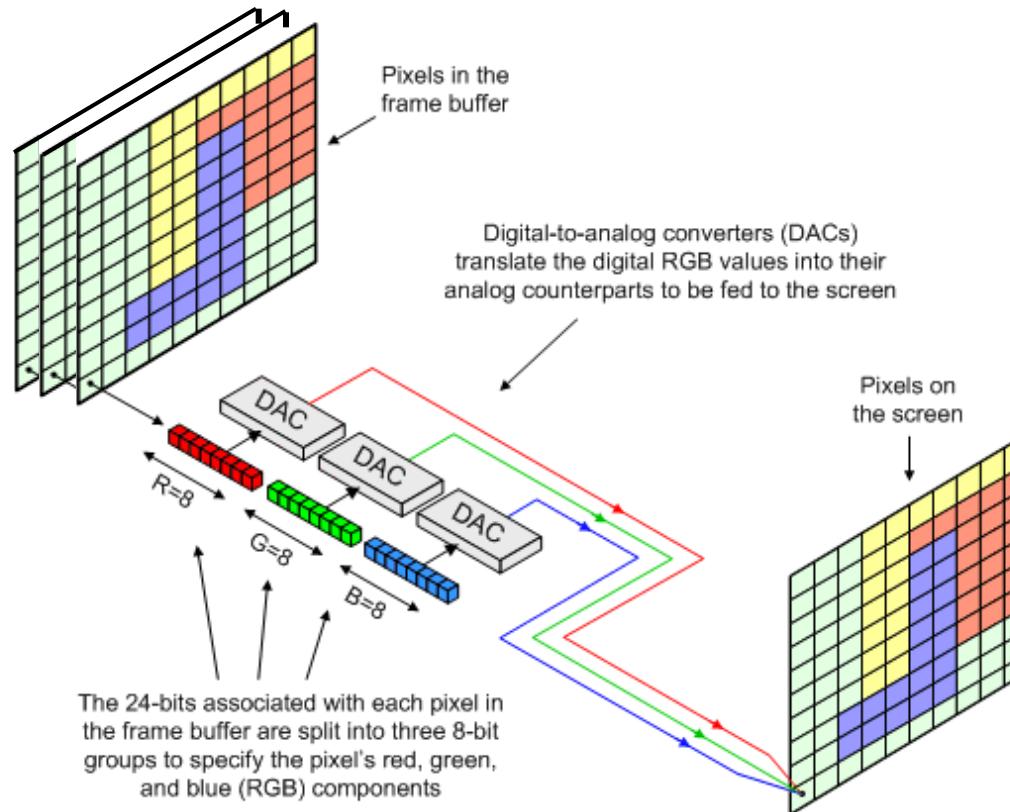
Memoria RAM identificata con un array bidimensionale



- Un elemento in una particolare riga e colonna contiene l'informazione relativa all'intensità o colore della corrispondente posizione $[x,y]$ sullo schermo
 - Una riga nel Frame Buffer corrisponde ad una scanline nel Monitor
 - Un "1" nel Frame Buffer corrisponde ad un pixel acceso nel Monitor (sistema monocromatico)
 - Ma il Frame Buffer è un array monodimensionale, per cui dato $[x,y]$, la relativa posizione di memoria viene calcolata da:

st fm + M * y + x

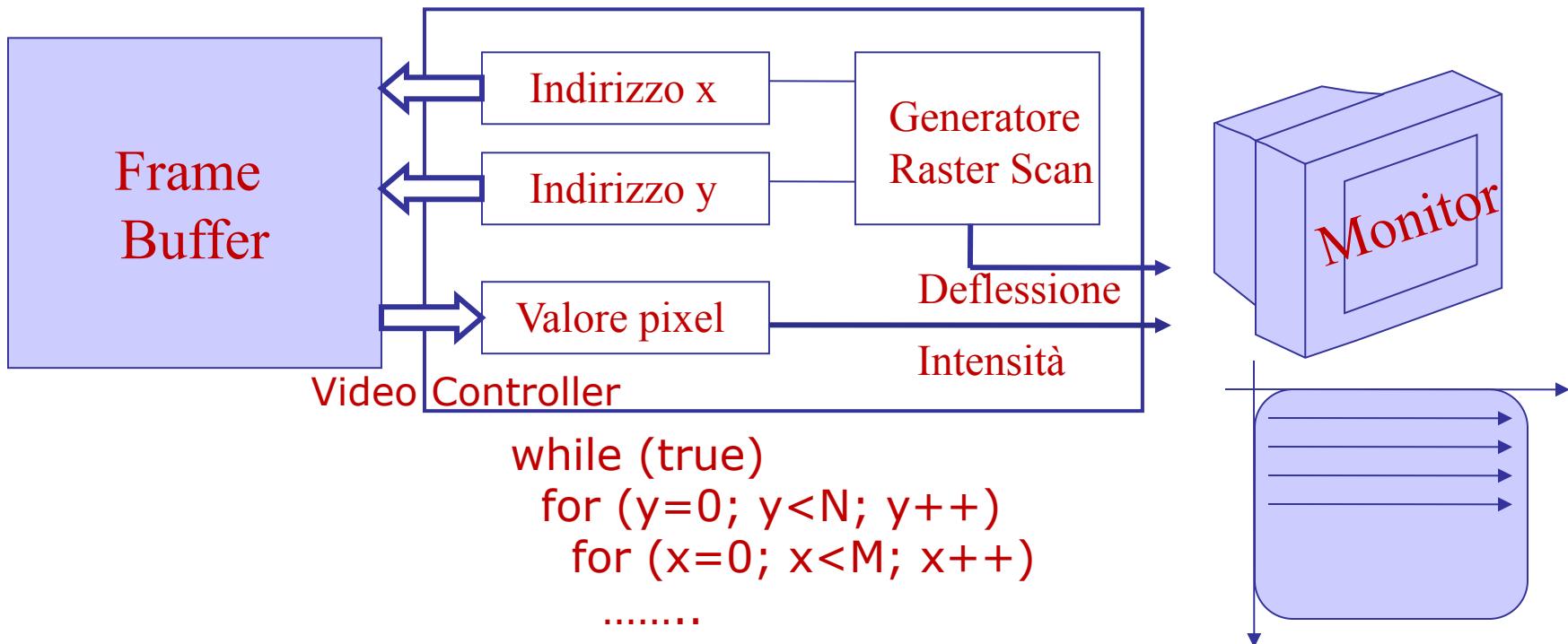
Frame Buffer 24 bit/color



- Oggi il più comune schema di Frame Buffer è ad **8 piani per colore**, per un totale di **24 piani**. Ogni gruppo di piani guida un DAC (Digital-Analogic Converter) ad 8 bit; questi sono combinati in $(2^8)^3 = 2^{24} = \textcolor{red}{16.777.216}$ possibili colori (Frame Buffer Full Color)



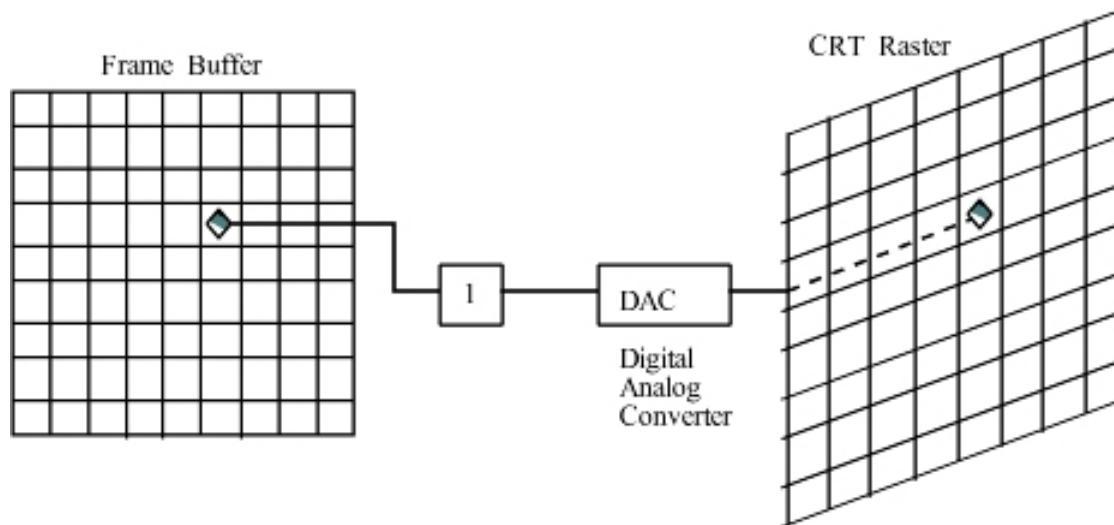
DPU (Display Processor Unit)



- A sistema funzionante, per accendere un pixel nella posizione $[x,y]$, sarà sufficiente inserire nella corrispondente posizione del Frame Buffer, l'informazione "1" (acceso in un sistema monocromatico).
- Tutti i software grafici possiedono una funzione per far questo:

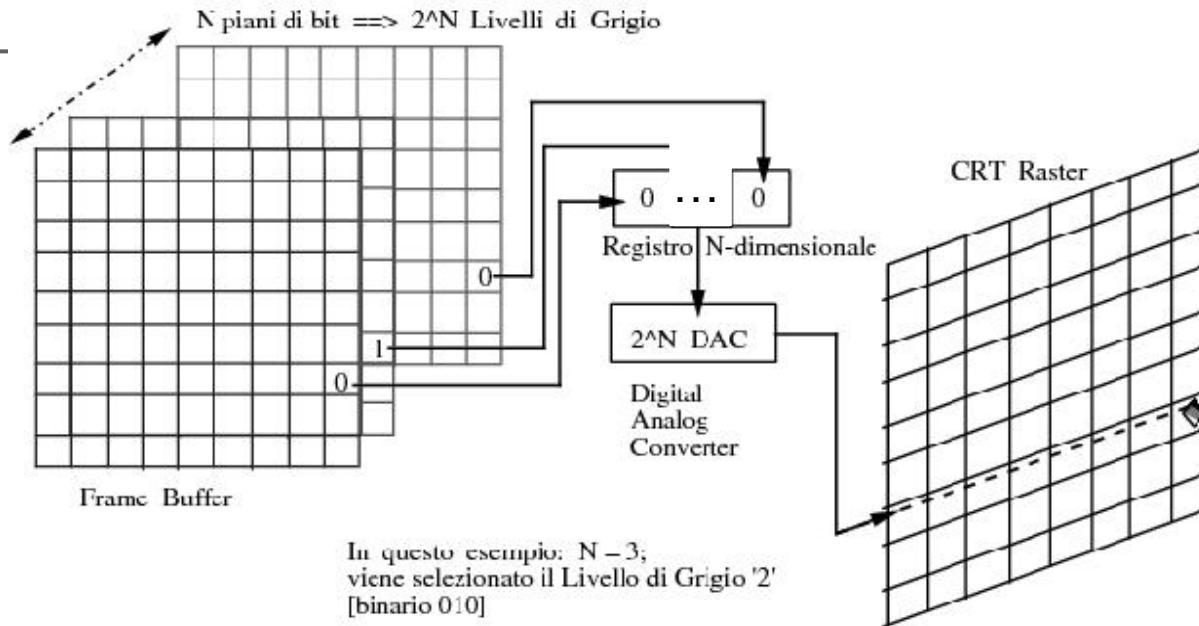
draw_point(int x, int y, unsigned long color)

Tipi di Frame Buffer (1/5)



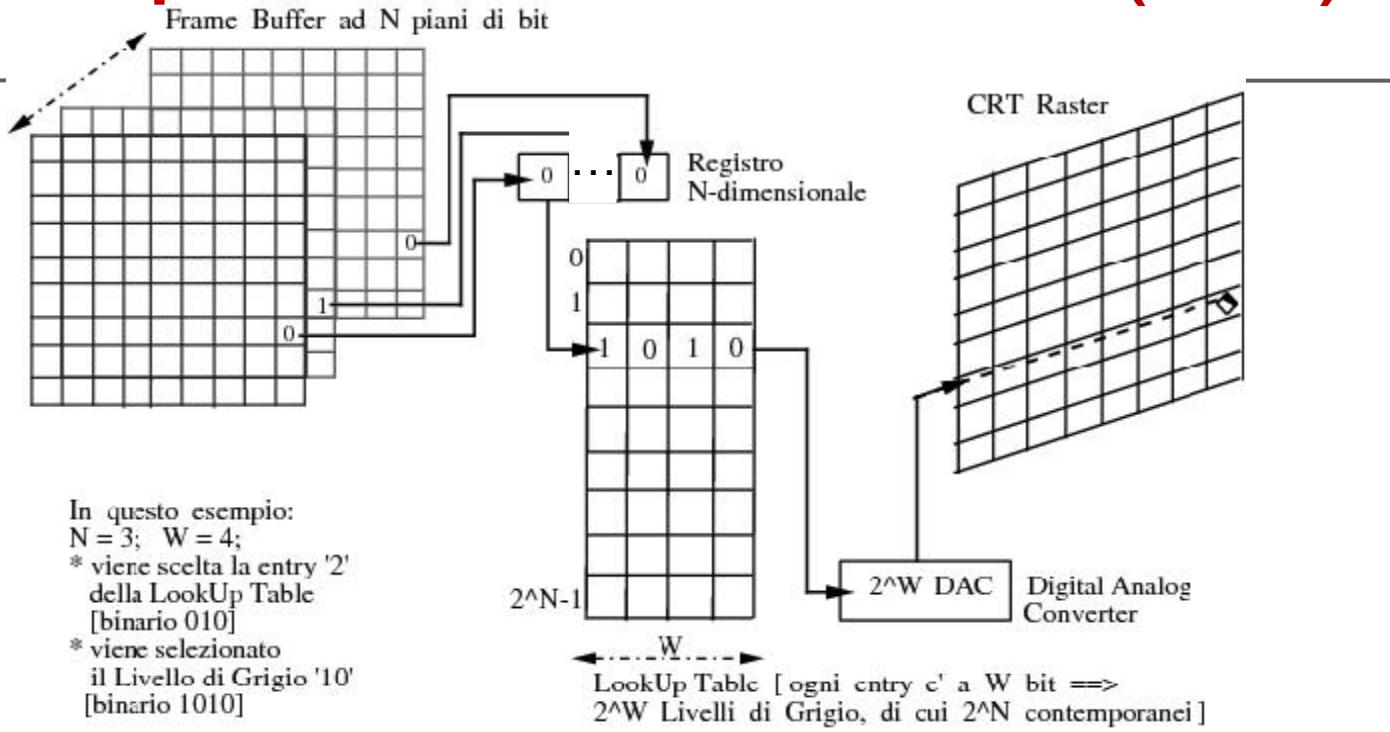
- Il Frame Buffer più piccolo: **1 bit per pixel**
- Un sistema 512×512 richiede $512 \times 512 = 2^{18} = 262.144$ bit di memoria
- Poiché 1 bit ha due stati, un Frame Buffer di 1 piano solo comporta un Display monocromatico
- Poiché un Frame Buffer è un dispositivo **digitale**, mentre il CRT è **analogico**, deve essere effettuata, come già detto, una conversione digitale-analogica usando un **DAC**.

Tipi di Frame Buffer (2/5)



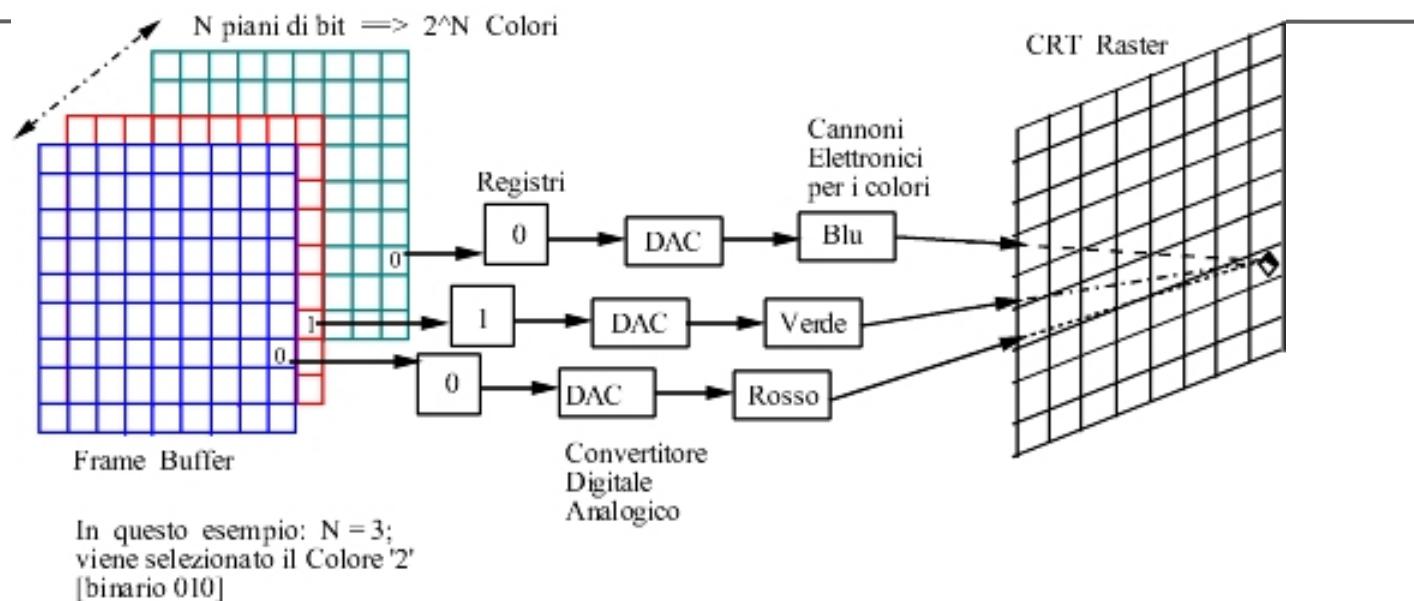
- Per più livelli di grigio è necessario un Frame Buffer con più di 1 piano;
- L'intensità di ciascun pixel sul CRT è controllata dalla corrispondente posizione del pixel in ciascuno degli N piani;
- Il valore binario (0 o 1) di ciascuno degli N piani viene caricato nella corrispondente posizione di un registro;
- Il risultante numero binario viene interpretato come un livello di intensità fra 0 (scuro) e $2^N - 1$ (massima intensità).
- Questo viene convertito in un voltaggio analogico fra 0 ed il max
- Sono possibili 2^N livelli (3 piani, $512 \times 512 = 786.432$ bit di memoria).

Tipi di Frame Buffer (3/5)



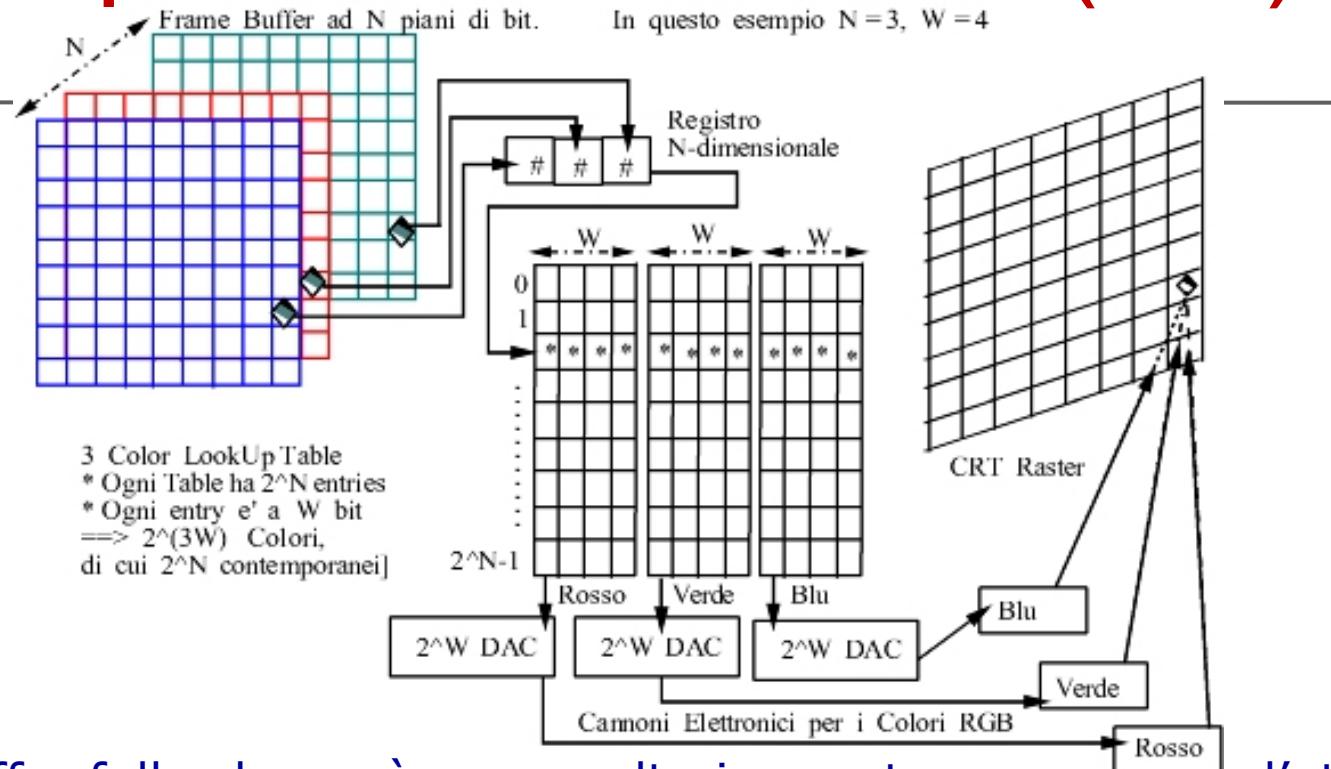
- Mediante l'uso di una **lookup table** si può ottenere un aumento del numero di livelli di intensità al prezzo di un modesto aumento di memoria.
- Il numero risultante dai bit dei piani nel Frame Buffer viene usato come indice nella lookup table.
- La lookup table deve contenere 2^N entry; ogni entry nella lookup table è formato da W bit. W può essere maggiore di N ; allora si dice che sono disponibili 2^W intensità, ma solo 2^N di queste sono contemporaneamente utilizzabili.

Tipi di Frame Buffer (4/5)

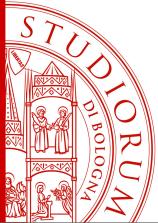


- Un semplice Frame Buffer per il colore si può ottenere con 3 piani ognuno per uno dei colori primari;
- I tre colori primari, **RED**, **GREEN** e **BLUE**, sono combinati nel CRT per avere gli otto colori (Black, Red, Green, Blue, Yellow, Cyan, Magenta, White).
- Possono essere utilizzati dei piani aggiuntivi per ciascuno dei tre colori; come detto, il più comune è uno schema di Frame Buffer con 8 piani per colore. Ogni gruppo di piani guida un DAC ad 8 bit; questi sono combinati in $(2^8)^3 = 2^{24} = 16.777.216$ possibili colori (Frame Buffer Full Color)

Tipi di Frame Buffer (5/5)



- Il Frame Buffer full color, può essere ulteriormente espanso con l'utilizzo dei piani come indici di **color lookup table**;
- Per **N piani/colore** con delle color lookup table di W bit, si possono avere contemporaneamente $(2^3)^N$ colori da una palette di $(2^3)^W$ possibili colori; per esempio per $N=8$ e $W=10$ si possono ottenere 2^{24} colori da una palette di 2^{30} .
- Per N piccolo ($<=4$) risulta più vantaggioso se le lookup table vengono implementate contigue con 2^N entry.



Youtube

How a TV Works in Slow Motion - The Slow Mo Guys

<https://youtu.be/3BJU2drirtCM>



Storia delle schede grafiche

CPU: computer di uso generale ('60);

VGA (Video Graphic Array) **Hardware controller (DPU)** (anni '80): sistema grafico speciale;

Graphics Hardware Unit ('90): sistema grafico a pipeline (circuiti VLSI (Very Large Scale Integration)), Silicon Graphics International (SGI) e Evans Sutherland progettano un costoso multichip;

GPU (Graphics Processor Unit) (fine '90): GPU a chip singolo, più economico, in PC e console per videogiochi.



Storia delle schede Grafiche

Con GPU ci si riferisce ad un microprocessore di una scheda grafica. Le GPU implementano in hardware la pipeline grafica per velocizzare la rappresentazione grafica 3D (accelerazione grafica 3D). La pipeline grafica è la sequenza delle operazioni per ottenere una immagine 2D a partire da una scena 3D; questa assume che il mondo 3D sia fatto di punti, segmenti e triangoli.





Storia delle schede grafiche

Le **GPU** si sono evolute da un'implementazione hardware della pipeline grafica ad un substrato computazionale programmabile in grado di supportarla.

Le unità a funzione fissa per trasformare i vertici e i pixel sono state incluse in una griglia di processori, o shader, in grado di eseguire queste attività.

Questa evoluzione ha avuto luogo nel corso di diverse generazioni sostituendo gradualmente i singoli stadi della pipeline grafica con unità programmabili.

Ad oggi si sono avute cinque generazioni di **GPU**. Si va verso un sistema grafico per il **rendering offline**.

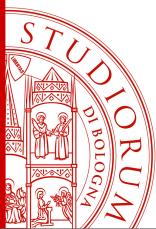


Schede Grafiche

GPU con pipeline fissa

I generazione (- 1998) GPU a chip singolo. TNT (NVIDIA), RAGE (ATI), Voodoo3 (3dfx), trasformazione vertici su CPU, elaborazione pixel su GPU, insieme limitato di operazioni matematiche su pixel;

II generazione (1999 - 2000) GeForce2 (NVIDIA), Radeon 7500 (ATI), Savage3D (S3), trasformazione e illuminazione vertici vengono eseguite anche in hardware (utilizza pipeline grafica fissa);



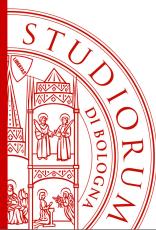
Schede Grafiche

GPU con pipeline programmabile

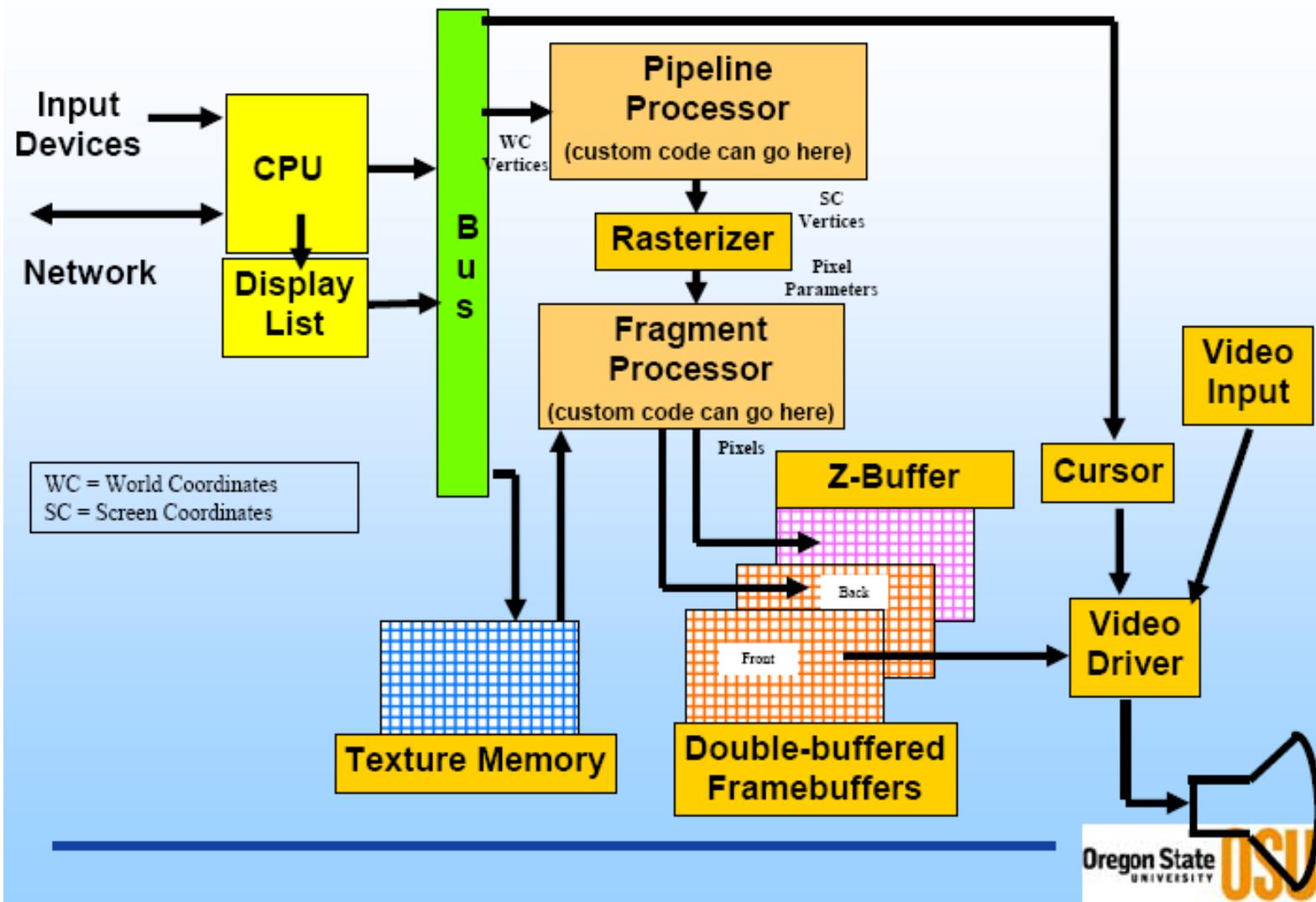
III generazione (2001) GeForce3, GeForce4 (NVIDIA), Radeon 8500 (ATI), hanno introdotto i vertex shader programmabili, la scheda grafica consente ai programmatore di scaricare assembly program per controllare trasformazione e illuminazione dei vertici; nessuna programmabilità dei pixel;

IV generazione (2002-2006) GeForce FX (NVIDIA), Radeon 9700 (ATI), Quadro4 XGL (NVIDIA), NVIDIA ha rilasciato Cg. Programmabilità per vertice e per pixel. Aumento dell'uso di effetti di illuminazione. Utilizzo di 32 bit in virgola mobile;

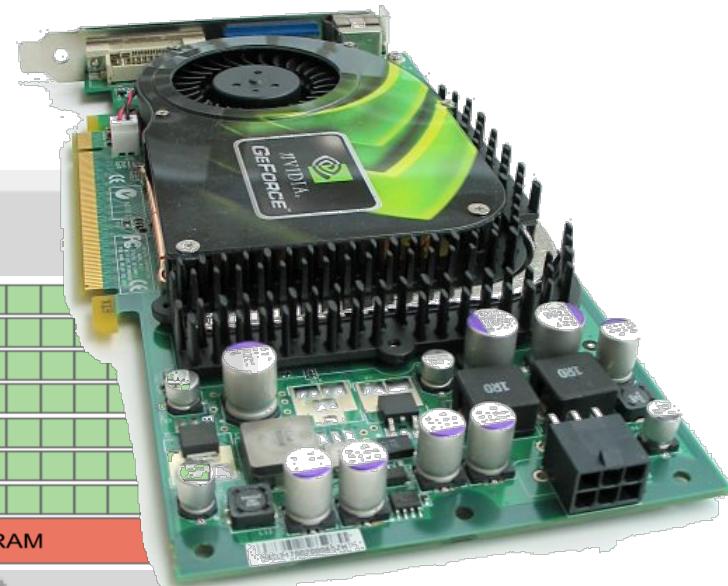
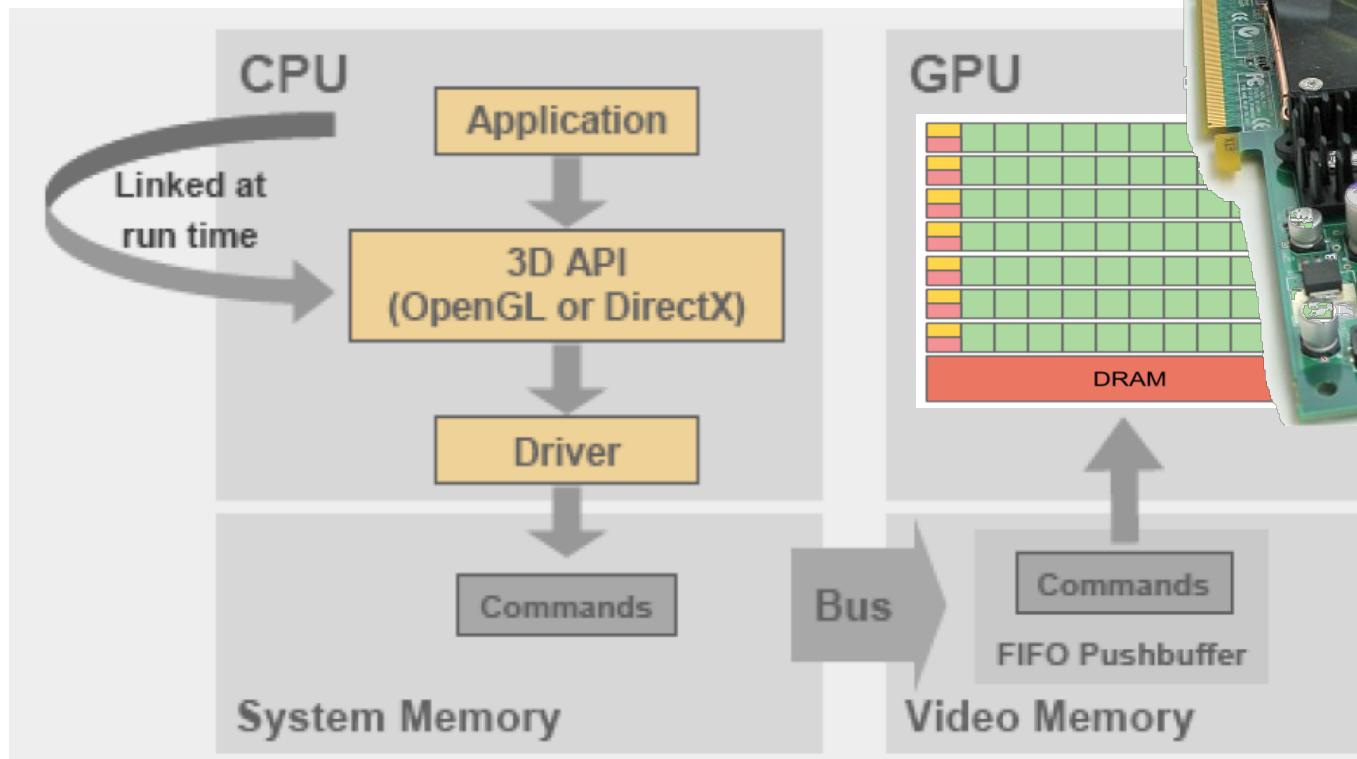
V generazione (ora) GPGPU GPU General Purpose: shader unificati, introdotti nelle console dei videogiochi, XBOX360, GeForce 8800 (128 proc.); 32 bit in virgola mobile in tutta la pipeline, GeForce 6+, INTEL (Larrabee), NVIDIA GeForce GTX285, AMD Radeon HD 4890.



Generico Sistema Grafico (ad oggi)



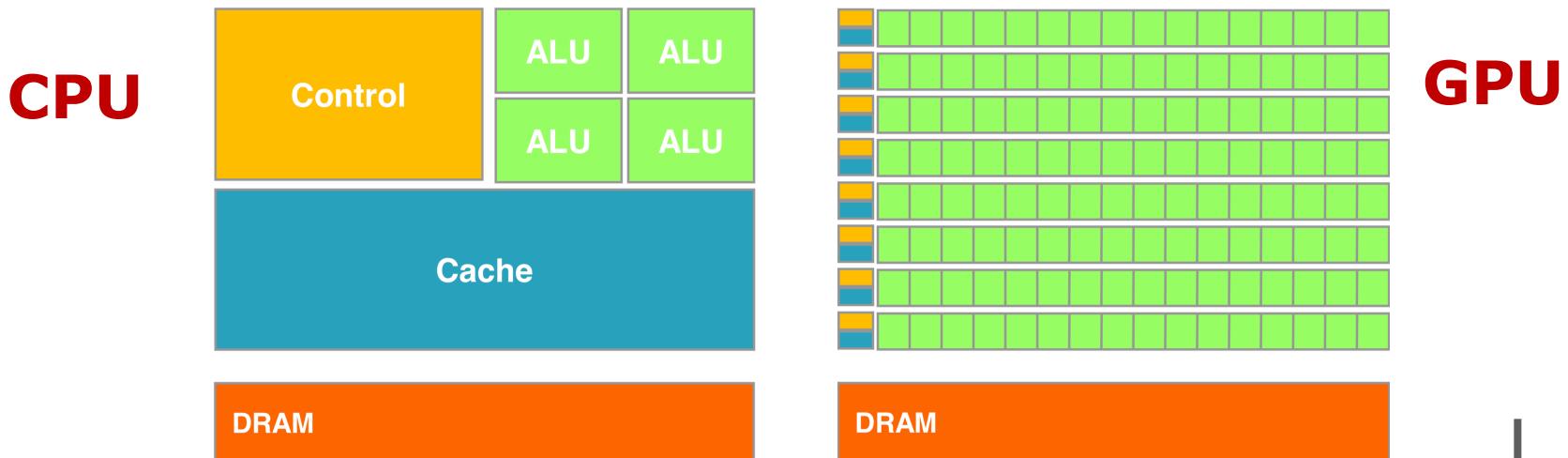
Graphics Processor Unit (GPU)



CPU vs GPU Architecture

Le GPU sono processori specializzati per problemi classificati come **intense data-parallel computations**, cioè la stessa computazione/algoritmo è eseguito su molti dati differenti in parallelo

- controllo di flusso molto semplice (control unit ridotta)
- limitata località spaziale (parallelismo a granularità fine)
- alta intensità aritmetica che nasconde le latenze di load/store (cache ridotta)



Performance delle GPU

-Nvidia Geforce 6800 GTX¹

6.4 billion pixels/sec

-Nvidia Geforce 7900 GTX²

15.6 billion pixels/sec

-Xbox 360³

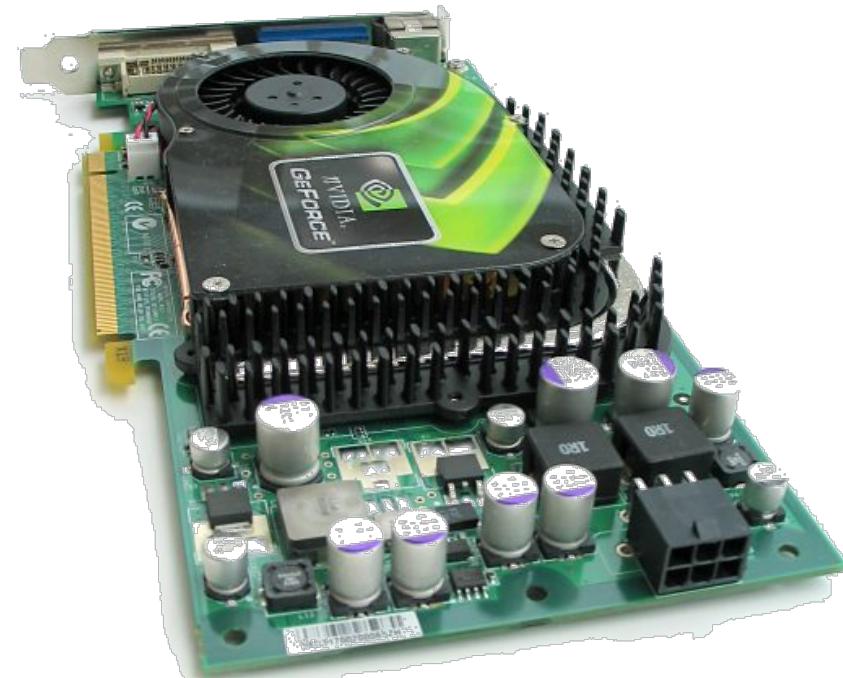
16 billion pixels/sec (4X AA)

-Nvidia Geforce 8800 GTX⁴

36.8 billion pixels/sec

-Nvidia Geforce GTX 280⁵

48.2 billion pixels/sec



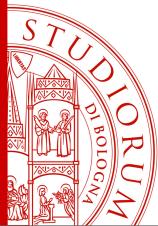
1: http://www.nvidia.com/page/geforce_6800.html

2: http://www.nvidia.com/page/geforce_7900.html

3: http://news.com.com/Xbox+specs+revealed/21001043_35705372.html

4: http://www.nvidia.com/page/geforce_8800.html

5: http://www.nvidia.com/object/geforce_gtx_280.html



Potenza dei Processori Paralleli

Nvidia Geforce GTX 280 (2008)

240 stream processor programmabili

1.3 GHz ciascuno

141.7 GB/s memory bandwidth

1 GB memoria

48.2 billion pixels/sec



Nvidia Geforce GTX 285 (2008)

240 stream processor programmabili

1.47 GHz ciascuno

159 GB/s memory bandwidth

1 GB memoria

51 billion pixels/sec

Nvidia Geforce GTX 295 (2009)

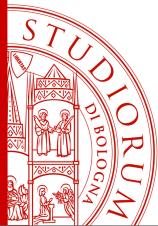
480 stream processor programmabili

1.24 GHz ciascuno

223 GB/s memory bandwidth

1.79 GB memoria

92 billion pixels/sec



Potenza dei Processori Paralleli

ATI Radeon HD 4870 (2008)

- 800 stream processor programmabili
- 0.75 GHz ciascuno
- 115 GB/s memory bandwidth
- 1 GB memoria

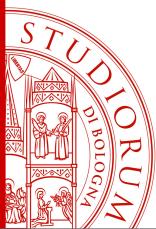


ATI Radeon HD 4850 X2 (2008)

- 1600 stream processor programmabili
- 0.625 GHz ciascuno
- 128 GB/s memory bandwidth
- 2x1 GB memoria

ATI Radeon HD 4870 X2 (2008)

- 1600 stream processor programmabili
- 0.75 GHz ciascuno
- 230 GB/s memory bandwidth
- 2x1 GB memoria



Potenza dei Processori paralleli



ATI Radeon HD 4850, 1.2 TFLOPS

ATI Radeon HD 4870 X2, 2.4 TFLOPS



IBM's ASCI Red, 1.338 TFLOPS

Computer più veloce al mondo nel 1997

Il presente:
GPGPU (General Purpose GPU)

Dispositivi Grafici Interattivi

- Tastiera
- Penna ottica
- Tavoletta grafica
- Joystick
- TrackBall
- TrackPad
- Mouse
- Touch Screen
- Scanner 3D Touch Probe
- Game Pad
- Leap



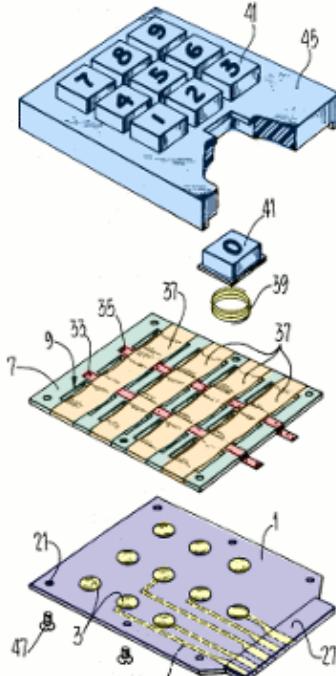
The Leap Motion Controller

Tecnologie dei Dispositivi Grafici

Tastiera

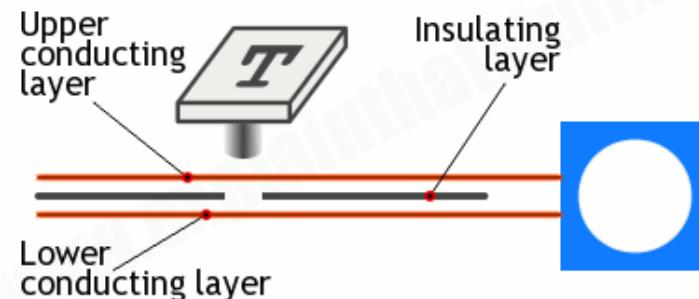
United States Patent

{72} Inventor James Martin Comstock
Livermore, Calif.



Courtesy US Patent & Trademark Office

www.explainthatstuff.com



James Martin Comstock, The Singer Company,
depositato nel 1969 e concesso nel 1971.

Tecnologie dei Dispositivi Grafici

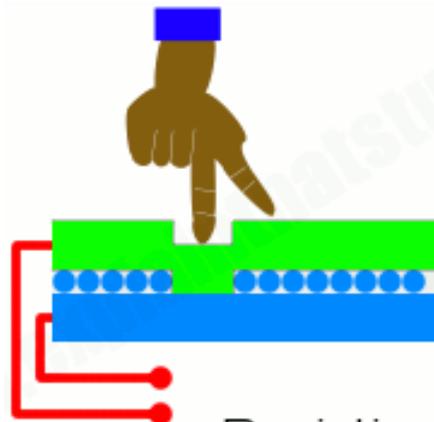
Mouse: la tecnologia attuale è quella ottica

- 1.Il LED che trasmette la luce verso la superficie d'appoggio;
- 2.la fotocellula che intercetta la luce riflessa dalla superficie;
- 3.il chip che si occupa della traduzione del segnale;
- 4.il sensore che intercetta i movimenti dell'eventuale rotella centrale del mouse;
- 5.lo switch che intercetta la pressione dell'eventuale rotella centrale del mouse;
- 6.lo switch che intercetta la pressione del tasto sinistro del mouse;
- 7.lo switch che intercetta la pressione del tasto destro del mouse;
- 8.il cavo di collegamento mouse-computer.

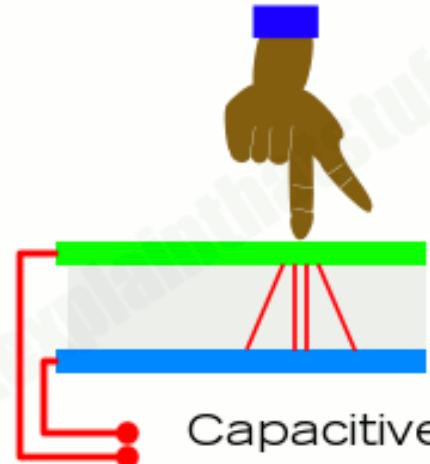


Tecnologie dei Dispositivi Grafici

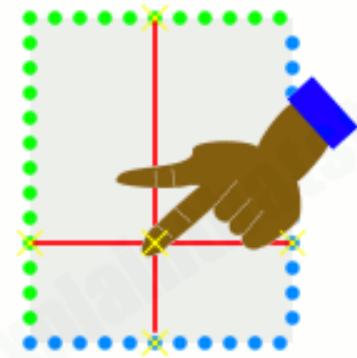
Touch Screen: ci sono diverse tecnologie, le principali



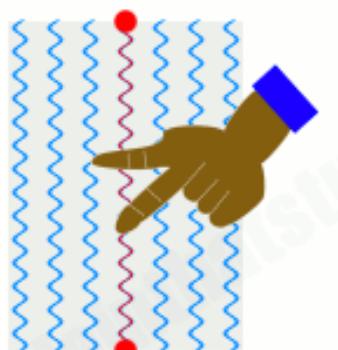
Resistive



Capacitive



Infrared



Surface acoustic wave



Near-field imaging



Il problema dell'interattività

1. Se ci sono più dispositivi di input, il programma non è in grado di prevedere quale verrà usato per primo;
2. Anche se ci fosse un solo dispositivo, non è possibile prevedere quando verrà usato.

Presi insieme, questi due problemi escludono l'uso di una classica istruzione bloccante come "scanf()" per leggere i dati di input.

Si fa ricorso ad una tecnica nota come **polling**, che periodicamente controlla lo stato di ogni dispositivo

- I dispositivi di input sono collegati al computer per mezzo di registri
- Questi vengono aggiornati a seconda dello stato dei dispositivi



Il problema dell'interattività

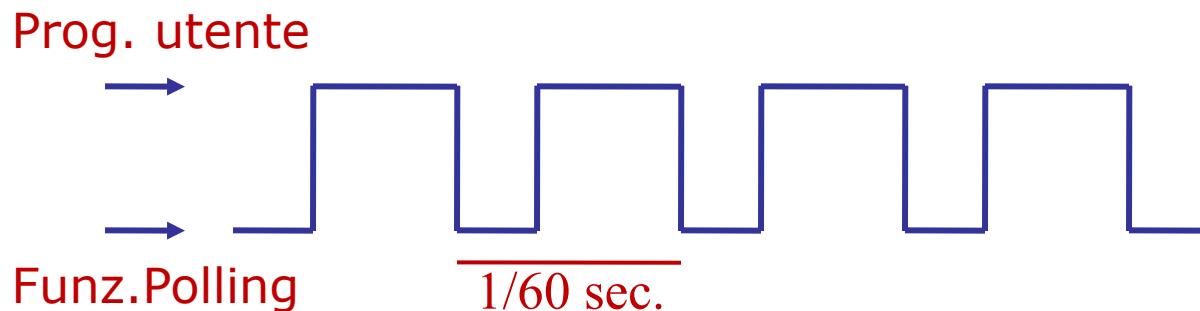
Per testare questi registri, ossia lo stato di ogni dispositivo, può essere utilizzato un semplice ciclo di polling.

```
while (TRUE)
{
    if (Bit stato tastiera è attivo)
        { azzerà Bit di stato;
          processa carattere;
        }
    if (Bit stato mouse è attivo)
        { azzerà Bit di stato;
          processa mouse;
        }
    .....
}
```

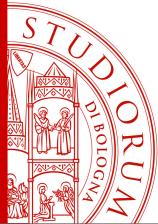
Il problema dell'interattività

Interazione fra routine di sistema che gestisce il ciclo di polling e il programma utente

- Mediante un clock interrupt viene generato un segnale ogni 1/60 di secondo
- Ad ognuno di questi istanti viene eseguita la routine di polling
- Una frequenza di 60 Hz è sufficientemente alta da assicurare che nessuna delle azioni dell'utente venga trascurata



Quando la funzione di polling rileva un cambiamento nello stato di una periferica di input, deve leggere i dati dai registri della periferica usata e trasmetterli al programma principale.



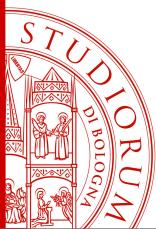
Coda degli Eventi

Per trasmettere i dati di input dalla funzione di polling al programma utente in maniera corretta, si utilizza un meccanismo denominato **coda degli eventi**; questa è costituita da una lista di azioni dell'utente o eventi.

- La funzione di polling aggiunge blocchi di eventi all'estremità della coda, richiamando il procedimento opportuno.
- Ogni software grafico interattivo possiede delle funzioni per interagire con la coda degli eventi

NextEvent(e) È una funzione bloccante; attende che si verifichi un evento, quindi restituisce le informazioni sull'evento nella struttura **e**

QueueEvent() È una funzione NON bloccante; ritorna il numero di eventi presenti nella coda



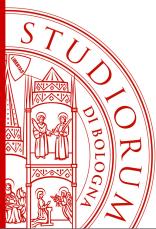
Programmazione Event Driven

La programmazione ad eventi è un paradigma di programmazione. Nei programmi di questo tipo il flusso è determinato dal verificarsi di eventi esterni.

Il codice è costituito da un **loop** all'interno del quale vi sono istruzioni che controllano la presenza delle informazioni da elaborare (per esempio la pressione di un tasto del mouse o della tastiera), e quindi nell'eseguire quella parte di programma scritta appositamente per gestire l'evento in questione.

I programmi event-driven sono composti da diverse piccole funzioni, chiamate **gestori degli eventi** (**event handlers**), eseguite in risposta agli eventi esterni, e da un **dispatcher**, che effettua materialmente la chiamata, a seguito dell'accesso alla **coda degli eventi** che contiene l'elenco degli eventi già verificatisi, ma non ancora "processati".

I programmi con interfaccia grafica sono realizzati secondo il **paradigma event-driven**.



Programmazione Event Driven

```
main( )  
{  
int done;  
....  
done=TRUE  
while (done)  
{  
    NextEvent(e )  
    switch (e.type)  
    {  
        .....  
        .....  
    }  
}  
}
```

Esempio 1

```
main( )  
{  
int done;  
....  
done=TRUE  
while (done)  
{  
    if (QueueEvent( ) >0)  
    {  
        NextEvent(e)  
        switch (e.type)  
        {  
            .....  
            .....  
        }  
    }  
}
```

Esempio 2

Ciclo principale del programma applicativo



Esempio: dragging di un'immagine

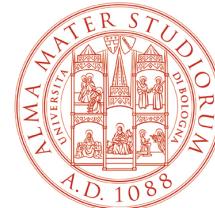
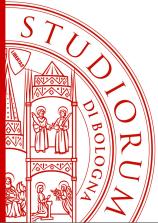
Problema: dal `ButtonPress` del mouse al `ButtonRelease` si vuole spostare un'immagine seguendo la posizione (coord. (x,y)) del mouse.

Le coordinate del mouse entrano nella coda degli eventi come eventi di `Motion`

Se l'immagine non è banale e il suo ridisegno comporta un certo tempo, non si avrà congruenza tra la posizione del mouse e l'immagine; si dice che non c'è `feedback`

Questo è dovuto al fatto che il sistema non riesce a processare gli eventi man mano che vengono generati e quando li processa sono diventati vecchi; in questo caso sarebbe meglio che la coda non contenesse mai più di un evento alla volta.

`PermitEvent(t)` Consente alla funzione di polling di aggiungere un evento di tipo `t` alla coda



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Giulio Casciola
Dip. di Matematica
[gjulio.casciola at unibo.it](mailto:gjulio.casciola@unibo.it)