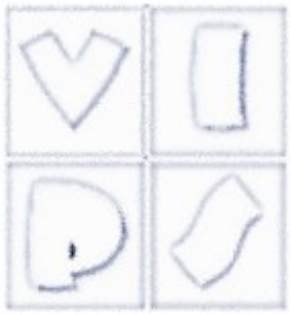
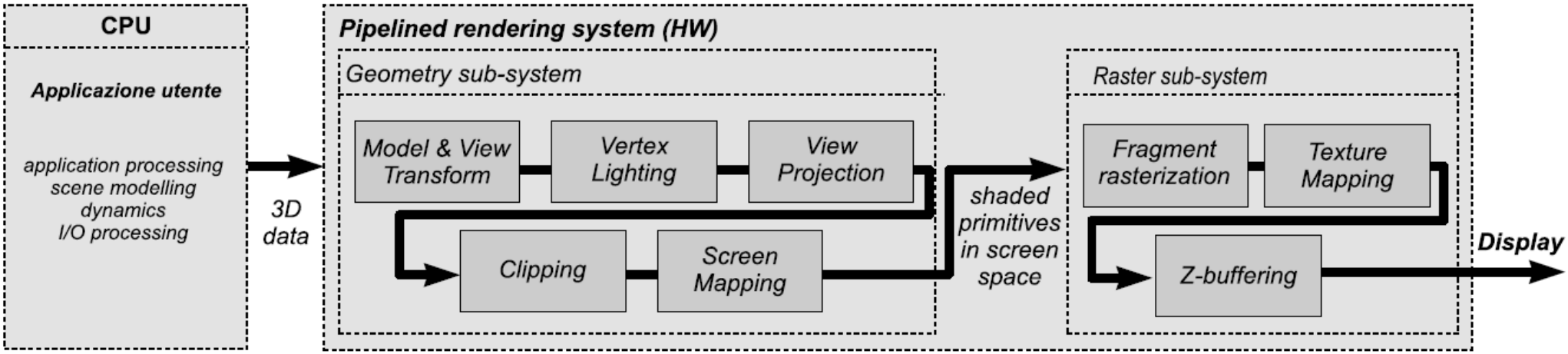


Grafica al calcolatore
-
Computer Graphics

2 – Applicazioni grafiche 3D

Pipeline di rendering interattivo

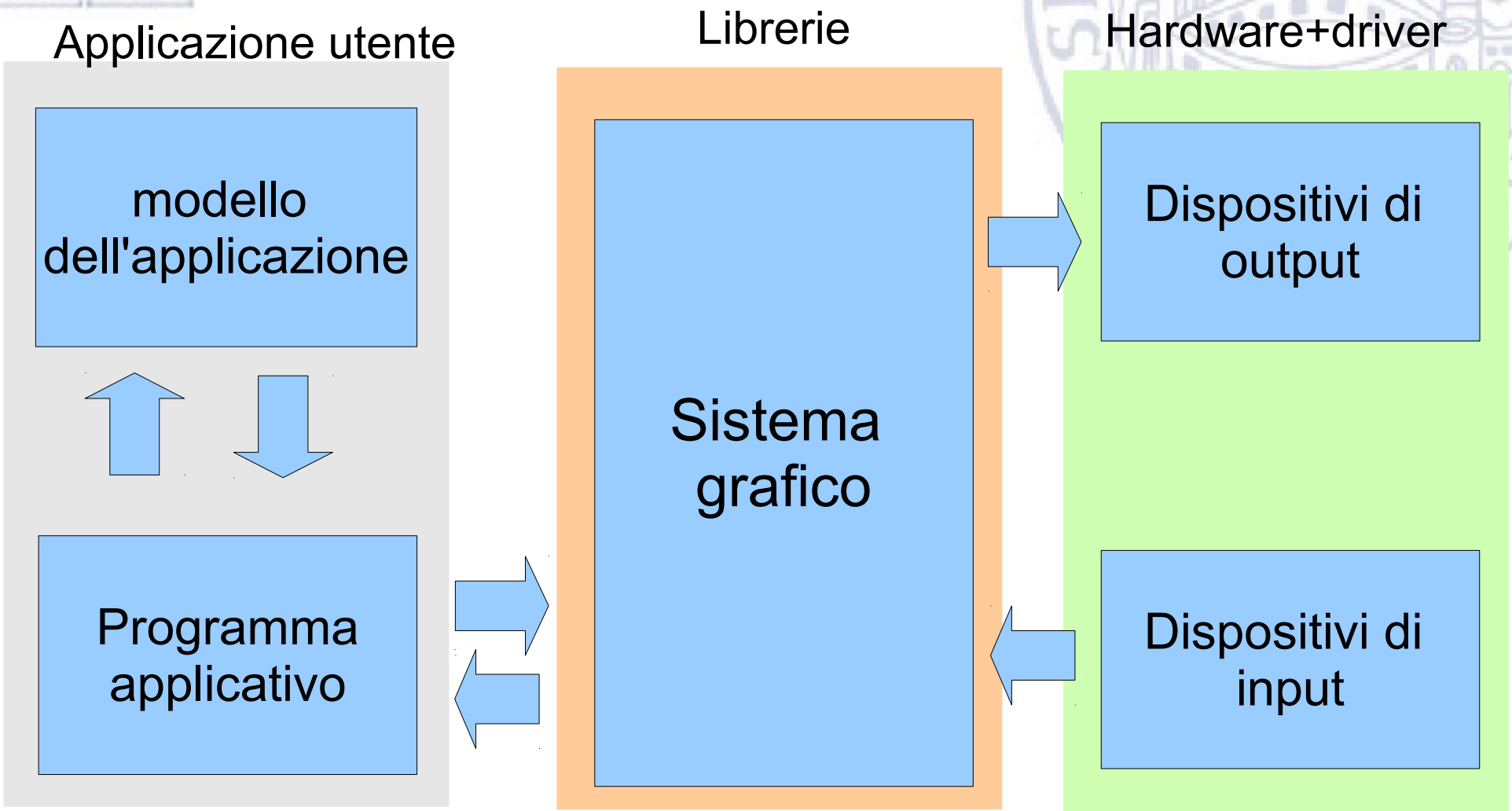
- Tre principali fasi elaborative:
  - gestione e trasmissione della rappresentazione tridimensionale (a cura dell'applicazione)
  - gestione della geometria (Geometry Subsystem)
  - gestione della rasterizzazione (Raster Subsystem)

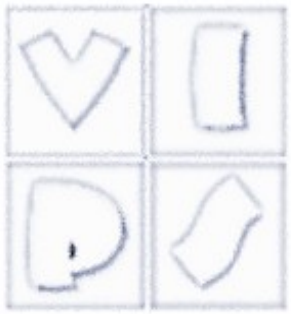


Modello dell'applicazione/programma

- Nell'applicazione ci saranno le scene (sistemi di riferimento spaziali) e gli oggetti per generare le imamgini
  - Modelli geometrici 3D, tipicamente mesh di triangoli, con coordinate punti e connettività
  - Alternativa: rappresentazioni volumetriche. Tipicamente cubetti pieni/vuoti (voxel) con caratteristiche dei materiali
  - Molte immagini (noto il punto di vista si può interpolare creando applicazioni grafiche interattive
- Il tutto deriva/è gestito dall'applicazione
  - Giochi
  - Dati da visualizzare (visualizzazione scientifica)
  - Simulazioni, ecc.

In pratica

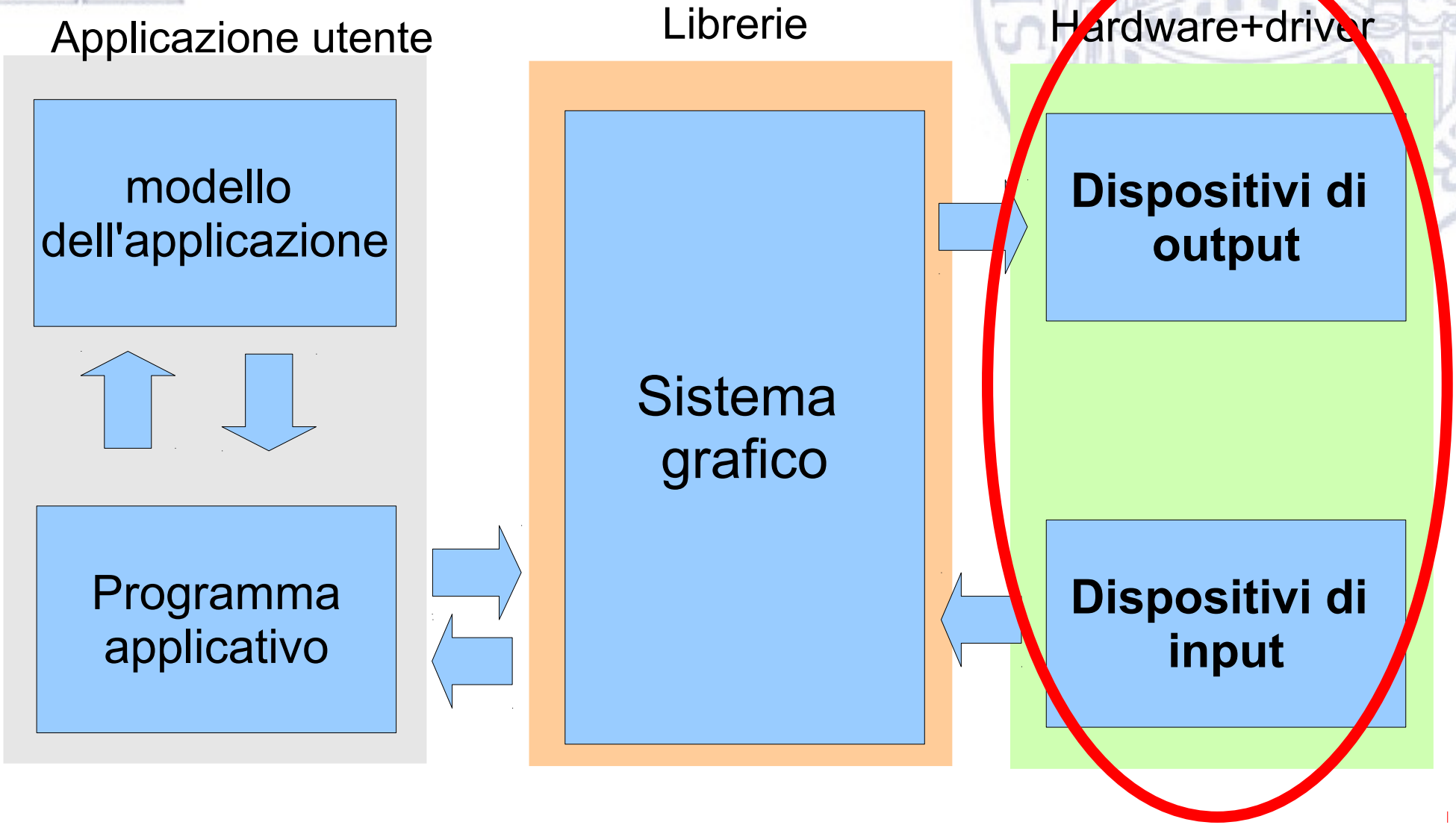

Applicazione utente
Librerie
Hardware+driver
modello dell'applicazione
Programma applicativo
Sistema grafico
Dispositivi di output
Dispositivi di input



Quindi

- L'applicazione grafica, per il momento una “black box” prenderà la descrizione dello stato della scena, modificabile dall'input nelle applicazioni interattive e genererà le immagini, demandando quanto possibile alle librerie grafiche e all'hardware
- In teoria potremmo usare molte metodologie per la generazione delle immagini, implementando approssimazioni di ciò che vedremo nel seguito del corso
- L'output deve essere un'immagine raster generata a frame rate interattivo nel “frame buffer”
- Facciamo ora una parentesi sull'input/output

Output e input

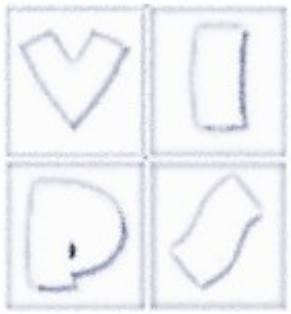


Output

- Raster display che consiste di una matrice di elementi denominati pixel
- Caratteristiche principali (non le uniche): risoluzione, (dimensioni della matrice di pixel), profondità di colore, (bit di memoria per pixel)
  - 8-bit significano 256 colori, mentre 24-bit (o truecolor) rappresentano all'incirca 32 milioni di colori
- frame buffer: memoria contenente l'immagine, array di valori per i pixel, che viene modificato direttamente dal programma di grafica video controller il quale legge il frame buffer e costruisce l'immagine sul display.

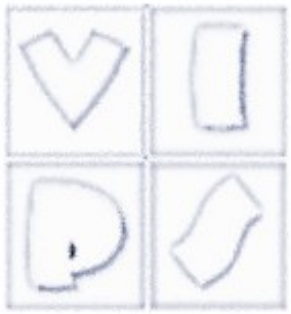
Output

- Display processor o graphics controller: in genere contenuto in schede grafiche dedicate, fornisce sia la memoria per contenere il frame buffer (liberando così la memoria principale del calcolatore) sia effettuando una serie di operazioni grafiche liberando così la CPU principale da tali incombenze.
- Il compito principale/minimale è la digitalizzazione dell' immagine tramite un processo denominato scan conversion. Poi c'è la pipeline grafica che vedremo



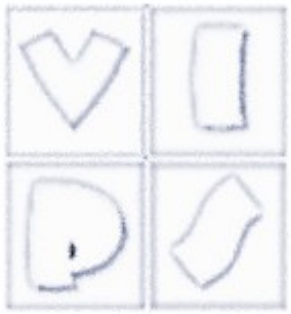
Tipi di schermo

- Vari tipi.
  - Display LCD/plasma
  - Display CRT
  - Schermi grandi a proiezione, ecc.
  - Sistemi immersivi, e.g. CAVE, occhiali stereoscopici



Caratteristiche del dispositivo

- Inizialmente (primi anni '60) dispositivi di tipo vettoriale, in grado di tracciare direttamente linee e punti (stesso concetto dei plotter a penna)
- La grafica di quegli anni usava quindi primitive di disegno di tipo vettoriale e modalità di visualizzazione wire frame



Caratteristiche del dispositivo

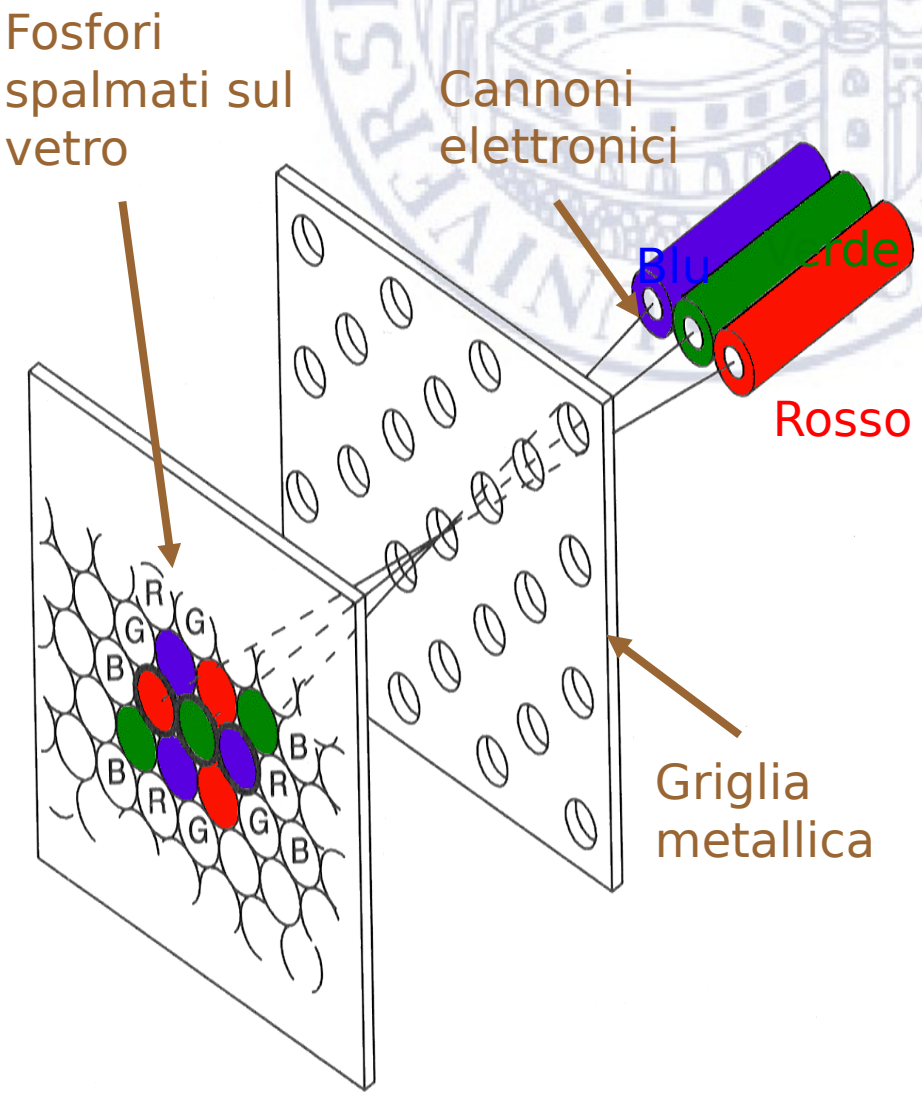
- I display attuali: tecnologia raster
- Spazio di output discreto bidimensionale
- Il termine risoluzione video indica il numero di pixel (picture element) dello schermo
- Importante oltre alla risoluzione la dimensione fisica del singolo pixel (il dot pitch o pixel pitch)
- Minore è la dimensione maggiore è la qualità del display (e il suo costo)

Caratteristiche del dispositivo

- Componente principale del sottosistema di pilotaggio del display: i banchi di memoria dedicati alla gestione del display stesso
- Si chiama memoria di quadro (*frame buffer*) e contiene le informazioni utili a generare ogni singolo pixel
- La memoria display è di tipo *dual-ported*, ossia supporta sia la scrittura che la lettura in modo indipendente
  - Lettura con frequenza costante (ordine di 60-85 Hz)
  - Scrittura variabile a seconda dell'applicazione con dati che provengono dal *raster subsystem*

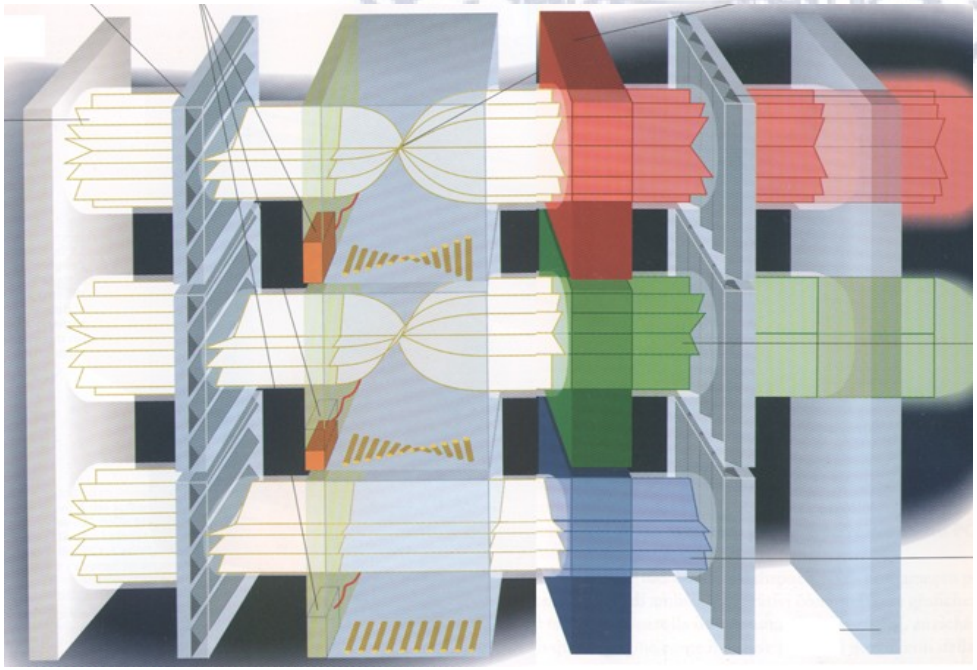
Display CRT

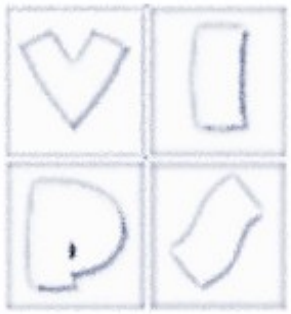
- Il singolo pixel è generato da tre fosfori che emettono luminosità sulle tre bande di colore *RGB*, di intensità proporzionale a quanto i singoli fosfori siano stati stimolati dal pennello di raggi catodici nella fase di refresh



Display TFT

- Piano fluorescente posto al fondo dello schermo
- Nella matrice attiva uno strato di cristalli liquidi guidati da un array di triplette di transistor che "torcono" i cristalli
- Colore tramite filtri colorati



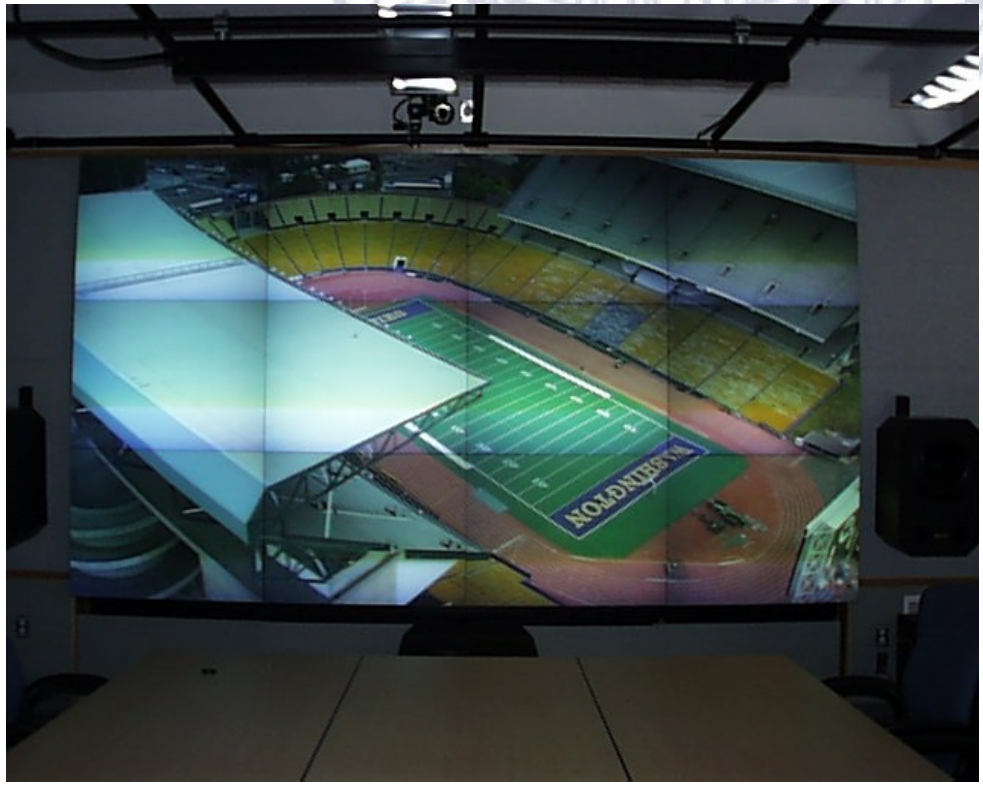


Sistemi a proiezione

- Ampia dimensione della superficie visibile
- Basati su tecnologia LCD o DLP (Digital Light Processing)
- LCD: come schermi TFT con proiezione a distanza
- DLP: costituiti da un pannello di micro-specchietti, uno per ogni pixel, di cui si comanda la rotazione su un asse; riflettono la luce incidente in proporzione al loro orientamento; si ottiene maggiore luminosità e nitidezza delle immagini

Sistemi a proiezione

- Problema: risoluzione limitata al singolo dispositivo (standard 1024x768)
- Soluzione: sistemi multi-proiettore (video wall) con suddivisione regolare dell'area di proiezione



Visualizzazione per Realtà Virtuale (immersiva e non)

- VR in computer normali (non immersiva)
  - schermo standard, controllo da tastiera o mouse
  - prospettiva e movimento danno effetto 3D
- VR immersiva
  - visione stereoscopica
  - caschi VR
  - schermo più occhiali oscurati ecc.
  - Tute, guanti, ecc.

Display stereoscopici

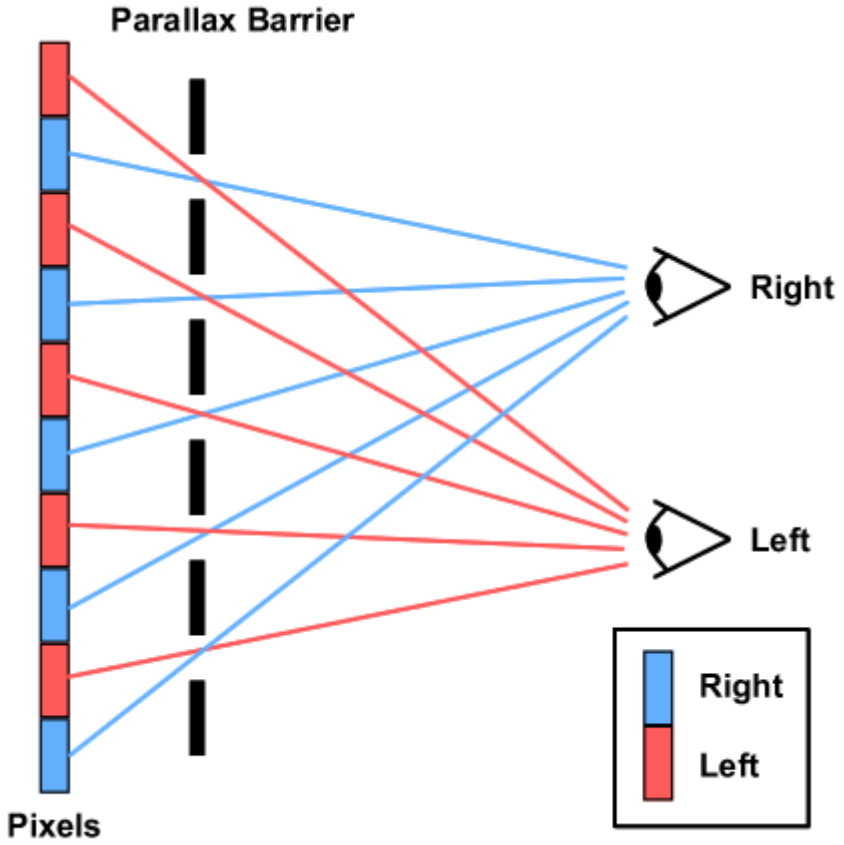
- Principio:
  - fornire due immagini leggermente diverse ai due occhi, in modo da ottenere la percezione di profondità (stereopsi)
- Stereoscopio (fine 19o secolo)
- Anaglifo
  - contiene due immagini sovrapposte, che rappresentano il punto di vista dei due occhi
  - occhiali con filtri cromatici

Head Mounted Display (HMD)

- 2 display LCD, uno per occhio
- LCD shutter glasses
  - Otturatore a LCD che diventa trasparente in sincrono con il display
  - Destra e sinistra alternati rapidamente
  - LCD con filter arrays
  - Matrice di prismi davanti al LCD
  - Pixel pari sull'occhio destro, pixel dispari sull'occhio sinistro
  - L'osservatore deve essere in una zona fissa

Evoluzioni

- Autostereoscopici
  - Sfruttano barriere per far vedere immagini differenti ai due occhi
  - Stanno diventando comuni (es. Nintendo 3DS)
  - Posizione/i fisse

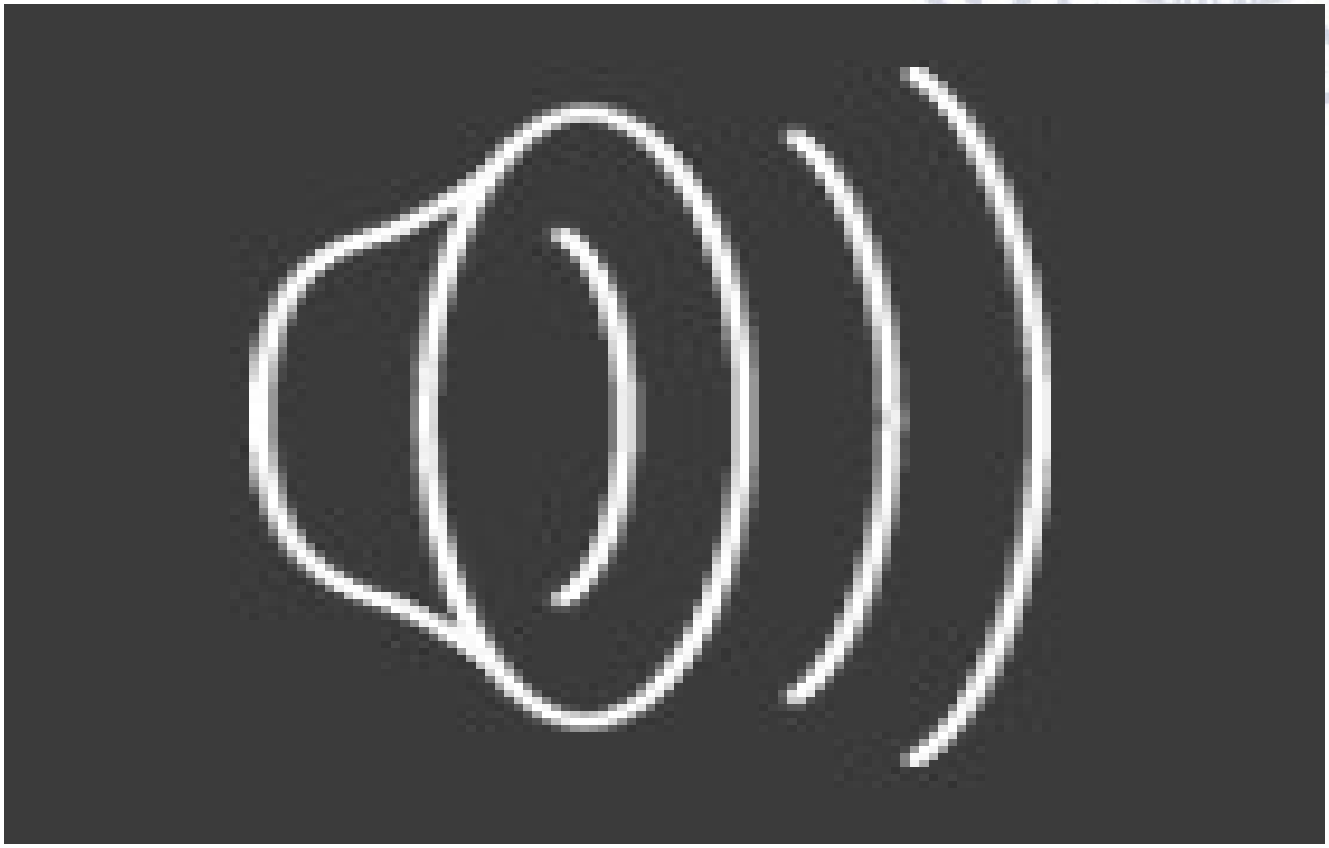


Immersività e realtà virtuale

- Naturalmente per sentirsi immersi in una scena 3D occorrerebbe che cambiando il punto di vista cambi la scena opportunamente.
- Coll'autostereoscopico addirittura si perde l'effetto 3D
- Con lo stereo si ha distorsione
- Soluzioni:
  - Tracking della posizione e generazione di una nuova scena (va bene per un utente, complesso
  - Monitor a parallasse continua (c'è comunque distorsione)

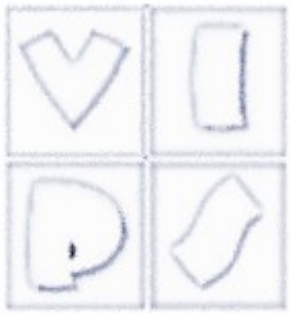
Display a parallasse continua

- Es holografika, in pratica molte direzioni con variazione continua



Complicazioni per i sistemi grafici

- Maggiore risoluzione accresce complessità
- Display stereoscopici richiedono 2 telecamere virtuali
- Display a parallasse continua ne richiedono molti
  - Array di schede grafiche per il controllo



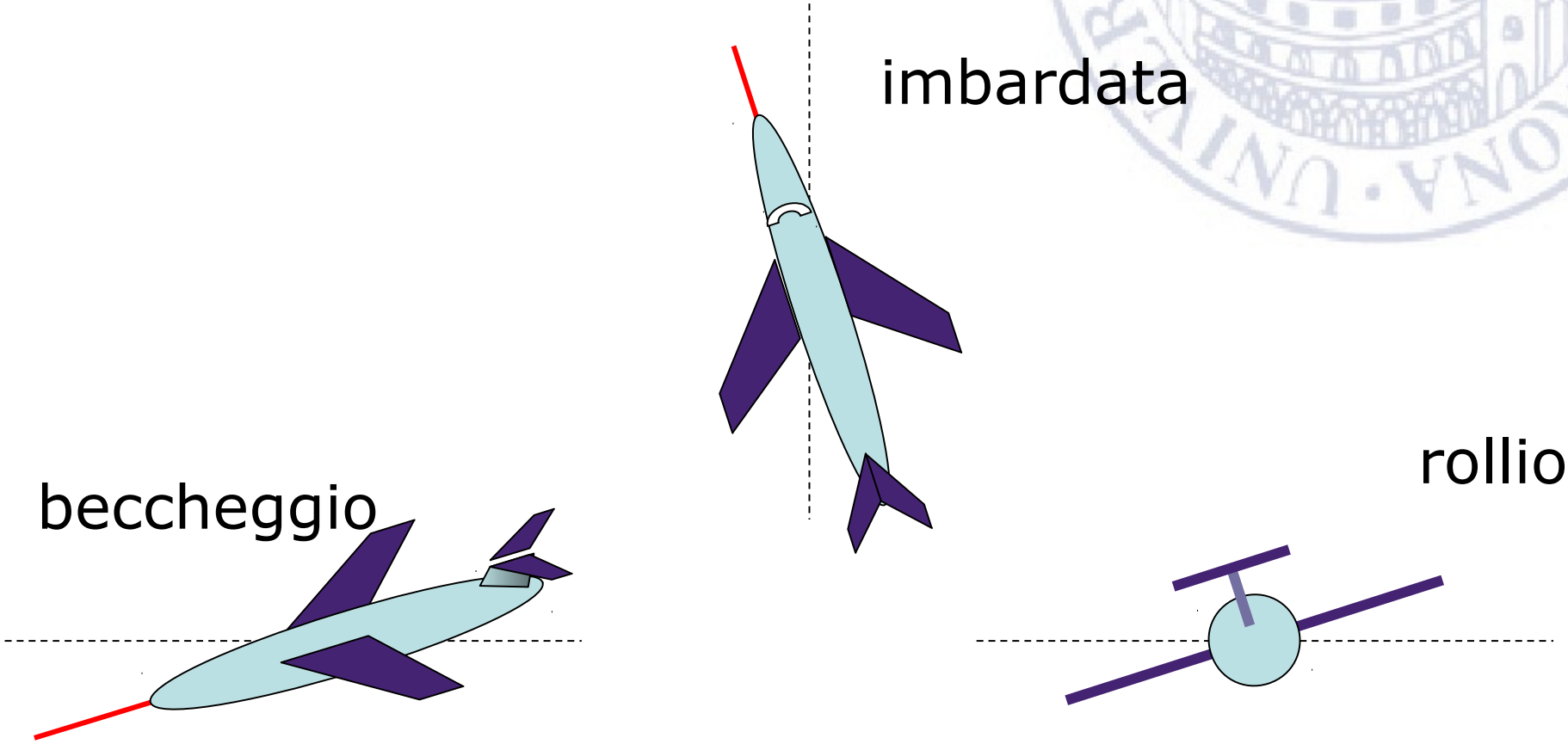
Dispositivi di input

- L'interazione col mondo 3D virtuale è tipicamente indiretta (tastiera, joystick)
- Si può pensare di rendere più “naturale” l'interazione?
- Sì ma non è ovvio che sia più usabile
  - Dispositivi di interazione 3D: possono acquisire orientazione, traslazione 3D di arti, punti, seguire la posizione dell'utente
- Anche per il gaming può essere fondamentale trovare il giusto meccanismo di input per l'interazione

Dispositivi di input particolari

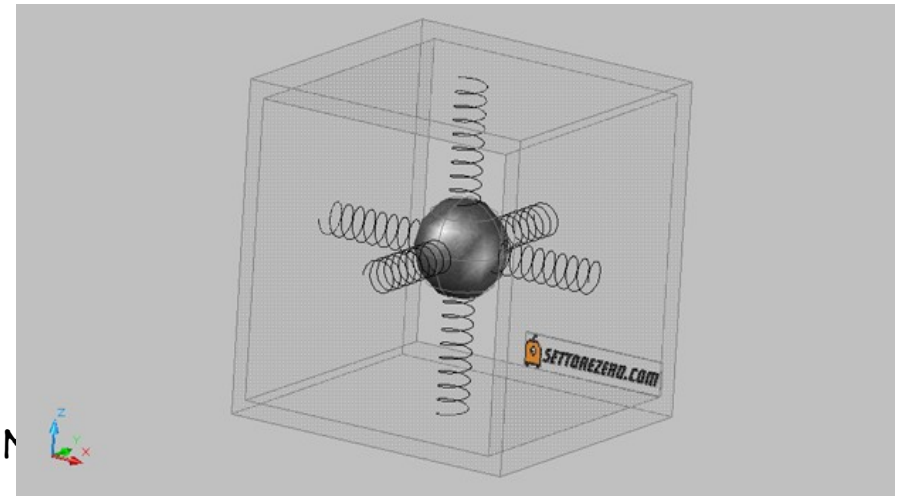
- mouse 3D
  - Strumento con 6 gradi di libertà (DOF): x, y, z + rollio, beccheggio, imbardata, non diretti
- Accelerometro
- Giroscopio
- Visione stereo
- Sensori attivi con visione IR (es. Microsoft Kinect), telecamere a tempo di volo
- guanto interattivo
  - fibre ottiche per rilevare la posizione delle dita

Orientazione in 3D



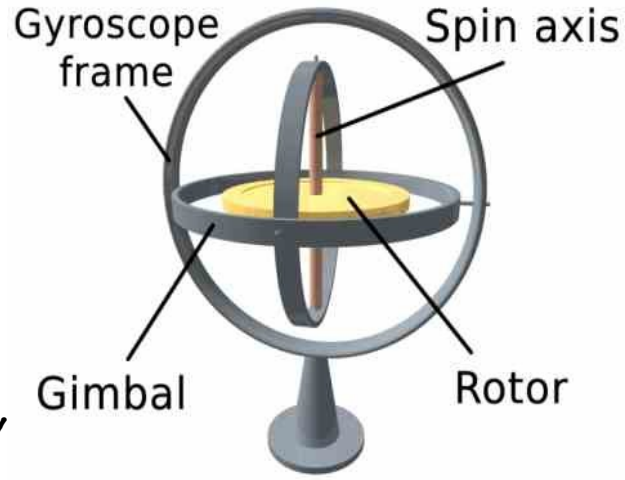
Accelerometro

- Può rilevare l'accelerazione su assi selezionati
- Idea di fondo semplice: pensate a un sistema come quello in figura massa-molla (estensimetro)
- Dato che siamo nel campo gravitazionale, possiamo calcolare l'orientazione di un oggetto in base alla direzione del campo rispetto all'accelerometro (specie se con più assi)
- Problema: non possiamo proprio calcolare l'angolo di imbardata: non cambia infatti il campo gravitazionale sull'oggetto



Giroscopio

- Permette di calcolare anche l'angolo di imbardata
- Il principio di funzionamento è quello della conservazione del momento angolare: se un oggetto ne ha, si oppone al tentativo di cambiare orientamento dell'asse di rotazione
- Anche dei giroscopi esistono versioni miniaturizzate e a basso costo
  - Incluse in smartphone, controlli videogiochi e non solo

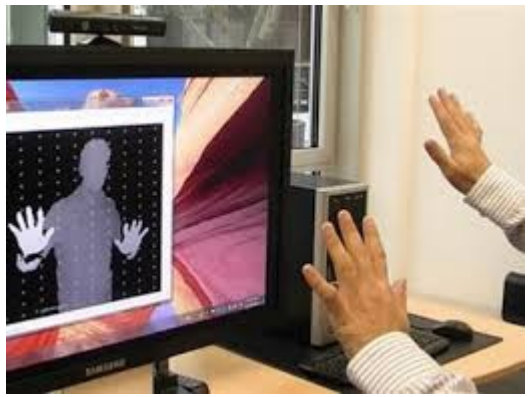


Nintendo WII

- Nintendo WII, controllo 3D con accelerometro e telecamera a IR:
  - Orientamento 3D (parziale)
  - Posizione mediante telecamera IR
  - Barra con led IR a distanza nota, SW che fa detezione di blob e triangolazione
  - Sottoprodotto: si può usare per tracking di penne a IR
  - Esempio: whiteboard
- Wii Plus: aggiunge giroscopio, può fare tracking posizione 3D assoluto

Microsoft Kinect

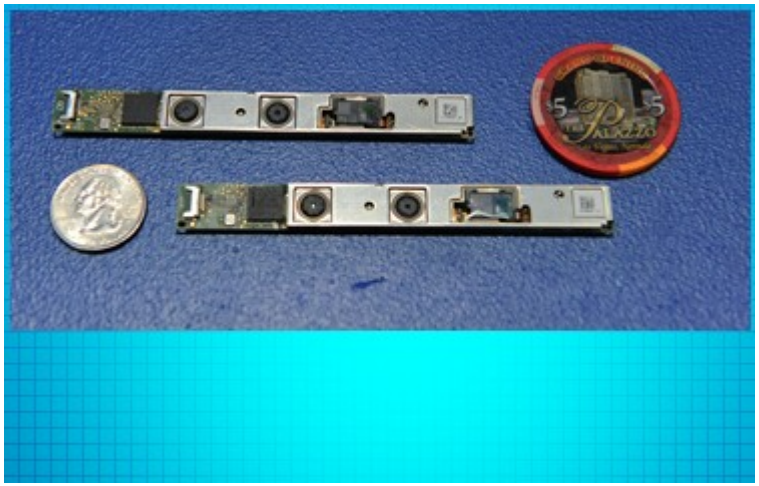
- Cattura mappa di profondità (oltre che immagini RGB) a frame rate interattivo
- In più acquisisce audio, immagine IR con pattern, ha modulo integrato per riconoscere modelli umani e fare tracking di scheletro.
- Basso costo, drivers e SDK disponibili (Microsoft, OpenNI)

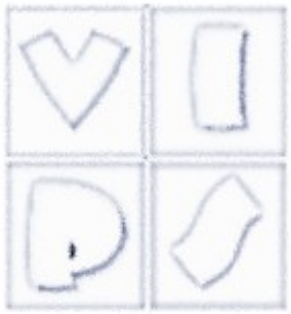


LeapMotion controller

- Sempre basato su visione 3D
- Sistema stereo con illuminatore a infrarosso
- Il software stima posizione di mani e dita (puntatori)
- Dichara accuratezza nel posizionamento molto elevata
- Latenza bassissima
- Costo basso (100 Euro)

Intel RealSense





Gestione dell'input

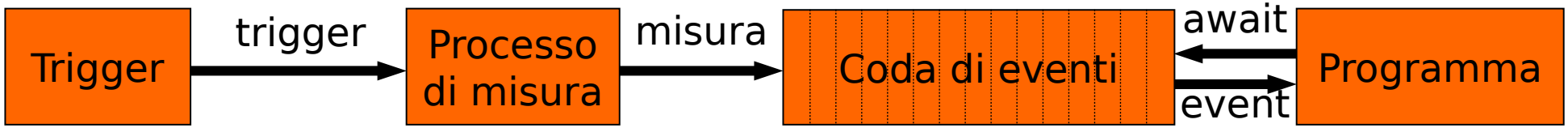
- API come openGL non prevedono gestione completa dispositivi di input, già nei sistemi a finestre (es. X11)
- Librerie specifiche per interfacciare gli ambienti dei sistemi a finestre con il sistema grafico (GLUT, GLFW, Qt)

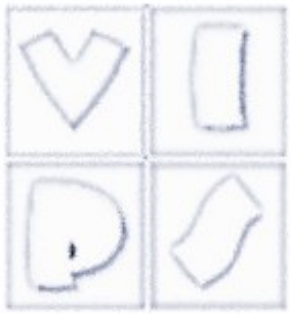
Gestione: modello ad eventi

- Gestione dell'input disaccoppiata dal suo uso da parte del programma applicativo
- L'applicazione opera in un ambiente con molteplici dispositivi di input, ciascuno con un proprio processo di misura e un trigger
- Ogni volta che si attua il trigger di uno di questi dispositivi, viene generato un evento
- La misura, insieme all'identificatore del dispositivo che l'ha eseguita, viene memorizzata in maniera asincrona in una event queue (coda di eventi)

Il modello ad eventi

- Il programma utente può esaminare l'evento in testa alla coda se esiste, oppure può attendere che un evento si verifichi
- Il programma applicativo consuma gli eventi al momento in cui ha disponibile il time-slot per processarli

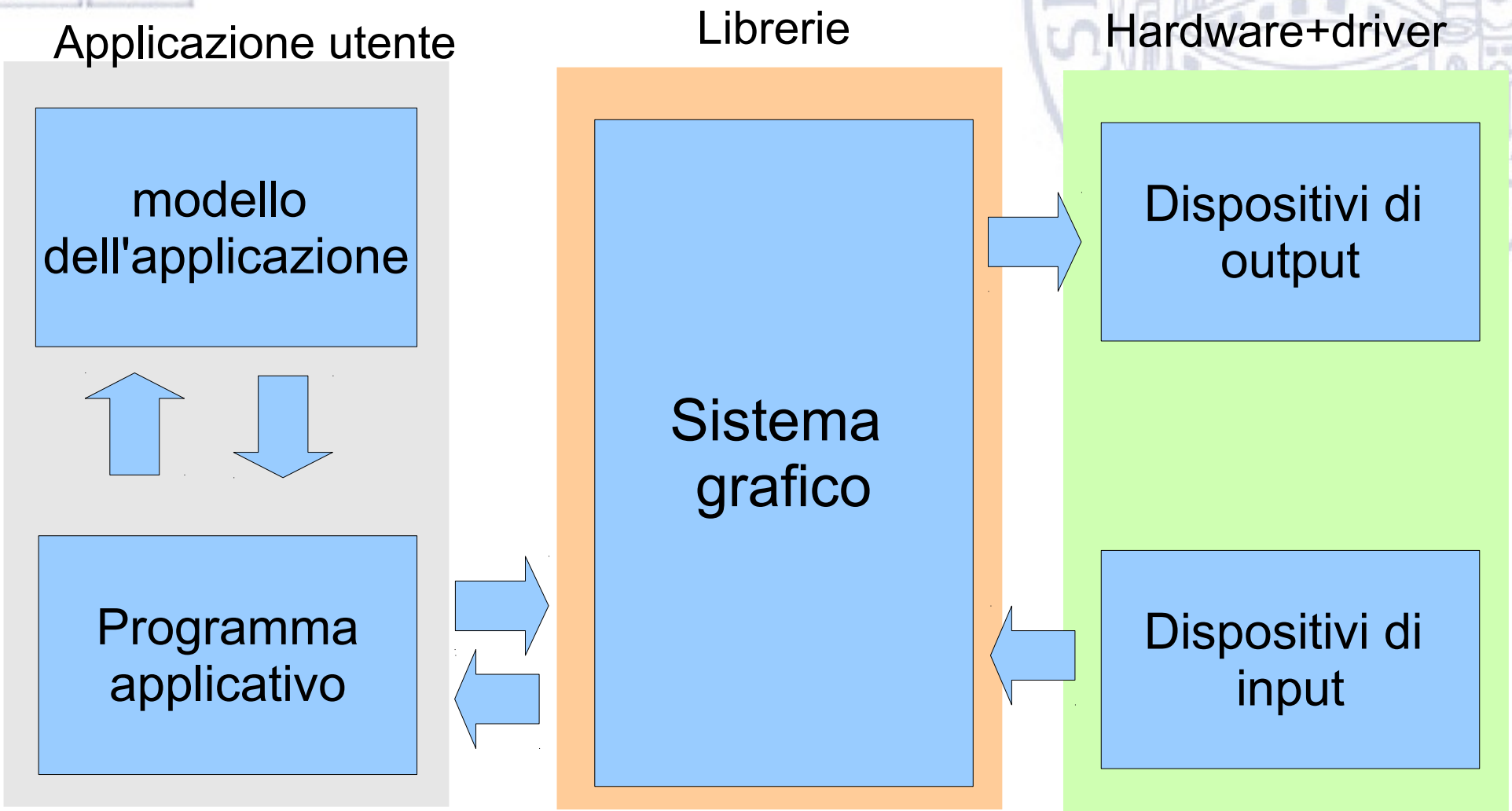




Callback

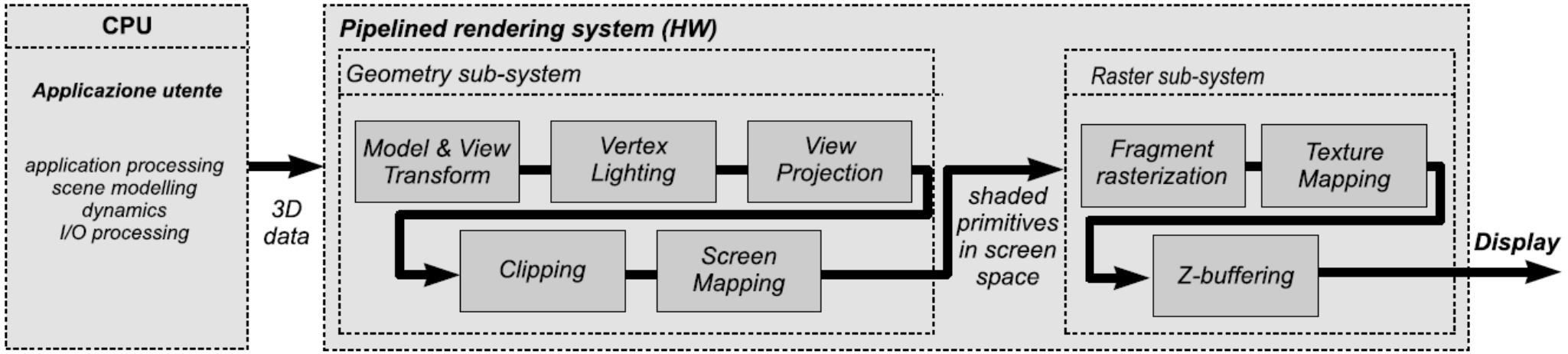
- Questo schema caratterizza gli ambienti client-server e le applicazioni interattive, in cui più processi concorrenti controllano le varie componenti del sistema
- Un modo comune per gestire dispositivi in questo contesto è quello di fornire una opportuna funzione (callback) per la gestione di ogni specifico tipo di evento
- Tale funzione e attivata dal gestore degli eventi del sistema a finestre senza che vi sia un esplicito controllo di attivazione da parte del programma applicativo
- È molto più semplice progettare e controllare sistemi complessi con interfaccia costituita da molte componenti indipendenti

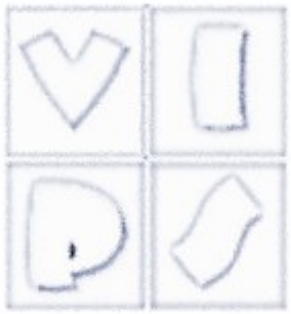
Torniamo all'applicazione



Hardware grafico

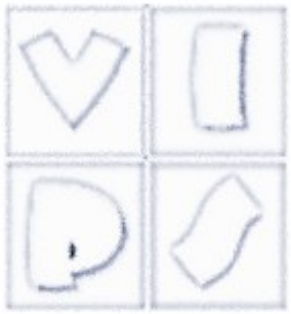
- Pipeline adottata dalle comuni API grafiche 3D e, conseguentemente, caratterizzante l'architettura dei sottosistemi grafici 3D hardware
- Le singole primitive 3D fluiscono dall'applicazione al sottosistema grafico, e sono trattate in modo indipendente dai vari stadi del sottosistema grafico
- Approccio proposto inizialmente da Silicon Graphics (1982) e poi adottato da tutti i produttori di HW grafico





Pipeline di rendering

- Tre principali fasi elaborative:
  - gestione e trasmissione della rappresentazione tridimensionale (a cura dell'applicazione)
  - gestione della geometria (Geometry Subsystem)
  - gestione della rasterizzazione (Raster Subsystem)
    - Queste ultime normalmente su GPU

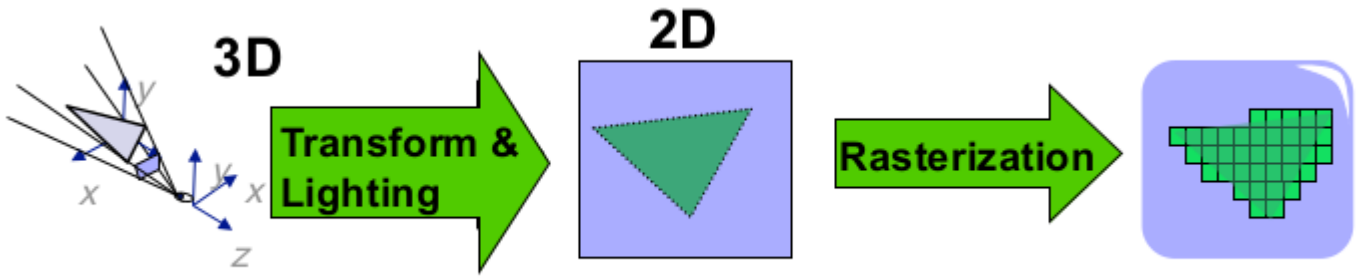


Primitive geometriche

- Le primitive geometriche effettivamente visualizzate in questa pipeline sono poligoni (modelleremo usando questi), di fatto solo triangoli
- L'applicazione ad alto livello avrà la scena definita con strutture dati varie, ma sceglierà ad ogni istante
  - Quali poligoni mandare alla pipeline grafica
  - I parametri relativi alla vista
- Da qui tutto avviene sull'hardware grafico
- La pipeline lavora quindi in object order, cioè si parte dalle primitive e non dai pixel dell'immagine che si vuole ottenere (image order)

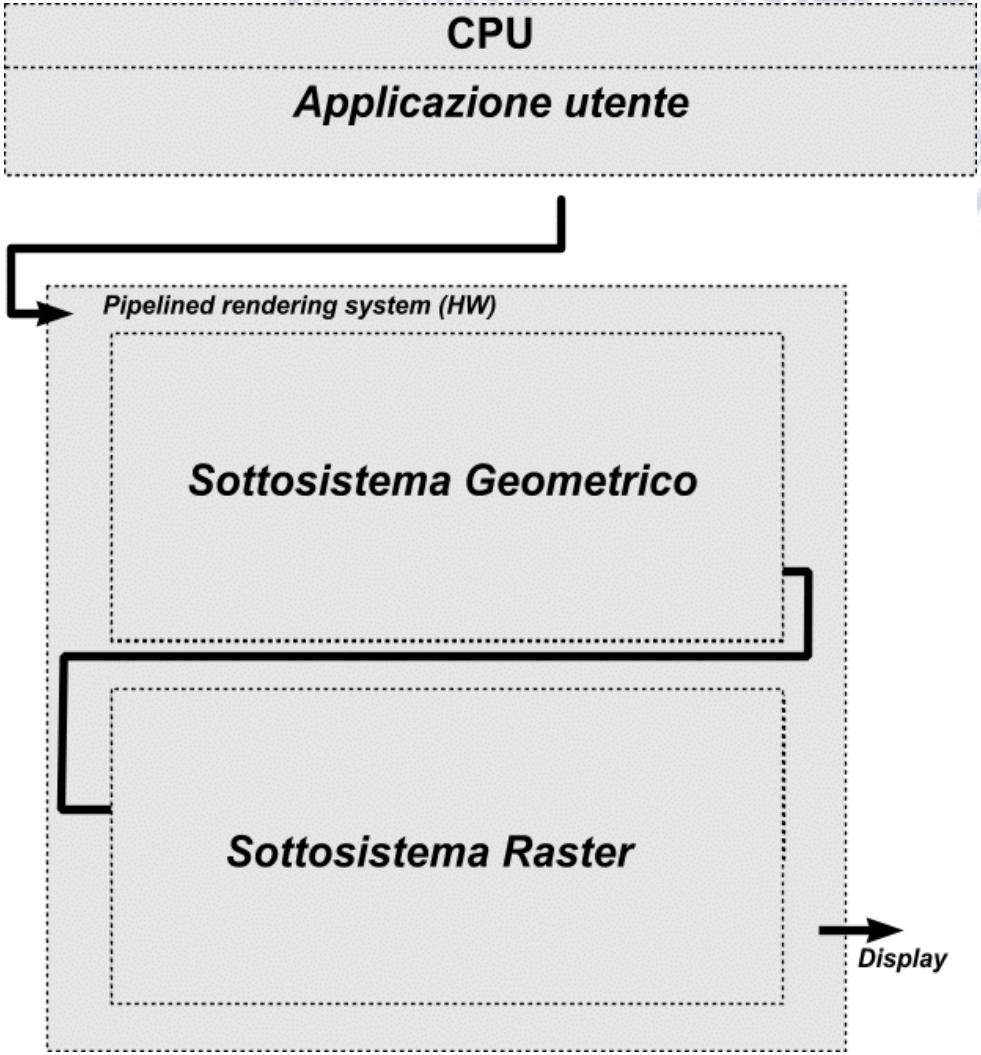
OpenGL Rendering Pipeline

- Nei prossimi lucidi ci concentreremo sul funzionamento della pipeline grafica di rasterizzazione riferendoci in generale alle librerie OpenGL
  - Basata sul concetto di rendering di una scena tramite la proiezione e rasterizzazione in object order di tutte le primitive della scena
  - 2 stadi: geometrico e rasterizzazione



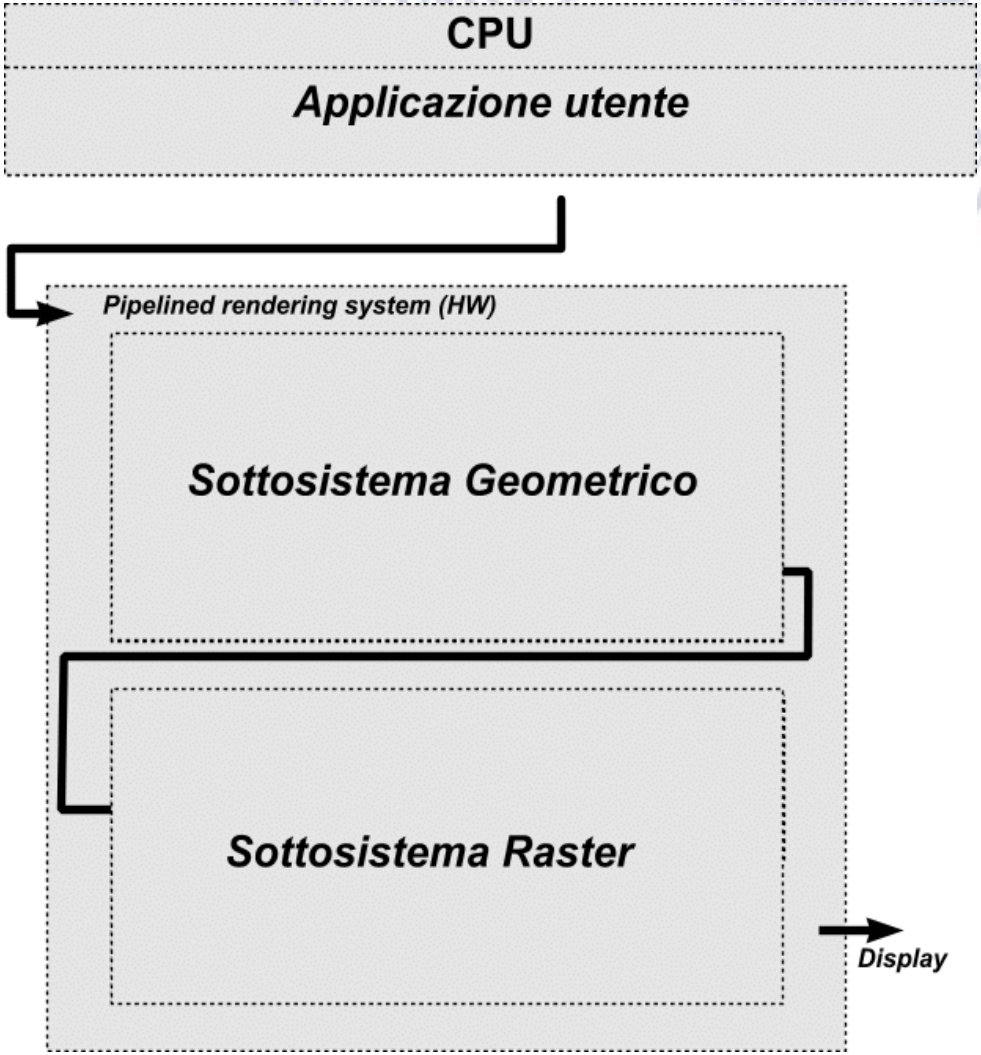
Basic Pipeline

- Tre principali attori in gioco
  - L'applicazione
    - che gestisce la scena e decide quali delle molte primitive che la compongono è necessario mandare agli stadi successivi della pipeline
  - Il sottosistema geometrico
  - Il sottosistema raster



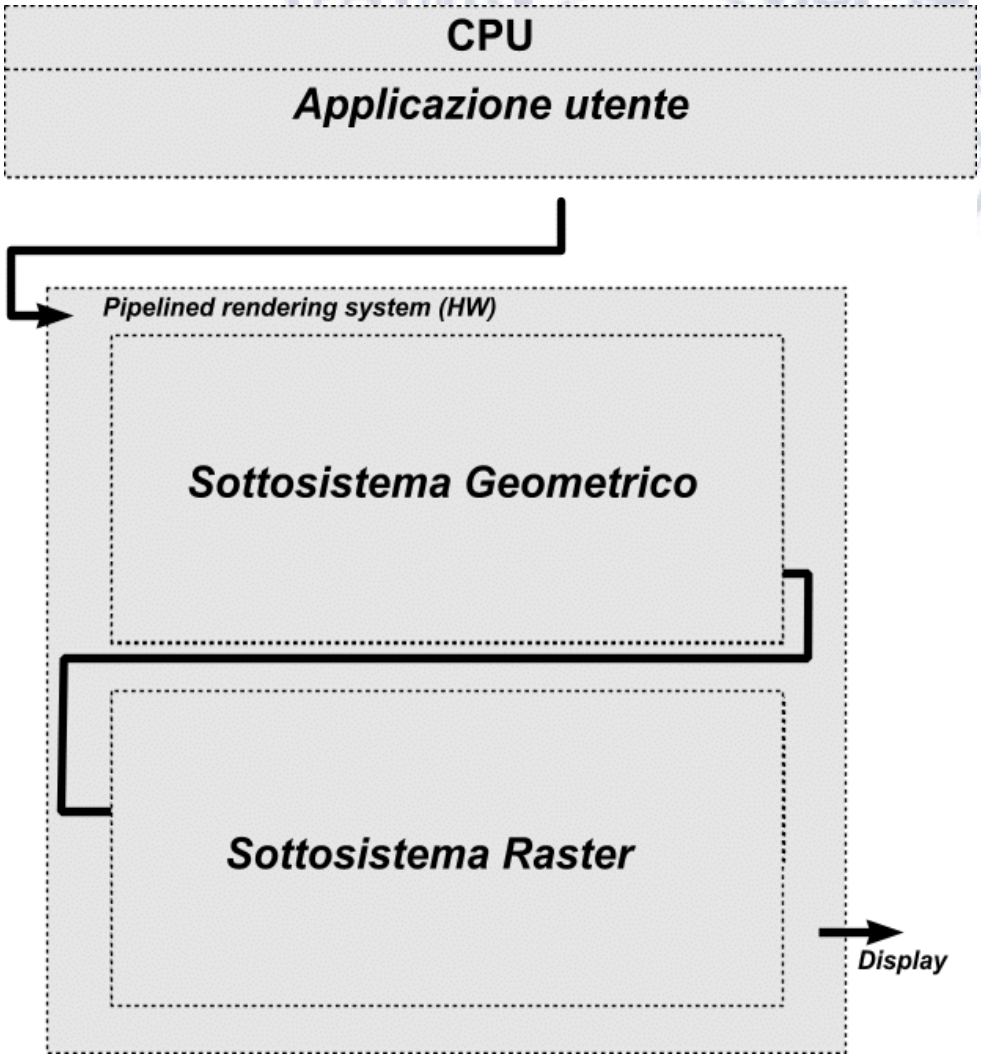
Basic Pipeline

- Tre principali attori in gioco
  - L'applicazione
  - Il sottosistema geometrico
    - che processa le primitive in ingresso e decide
      - se, (culling)
      - come, (lighting)
      - e dove (transf & proj)
    - devono andare a finire sullo schermo
  - Il sottosistema raster



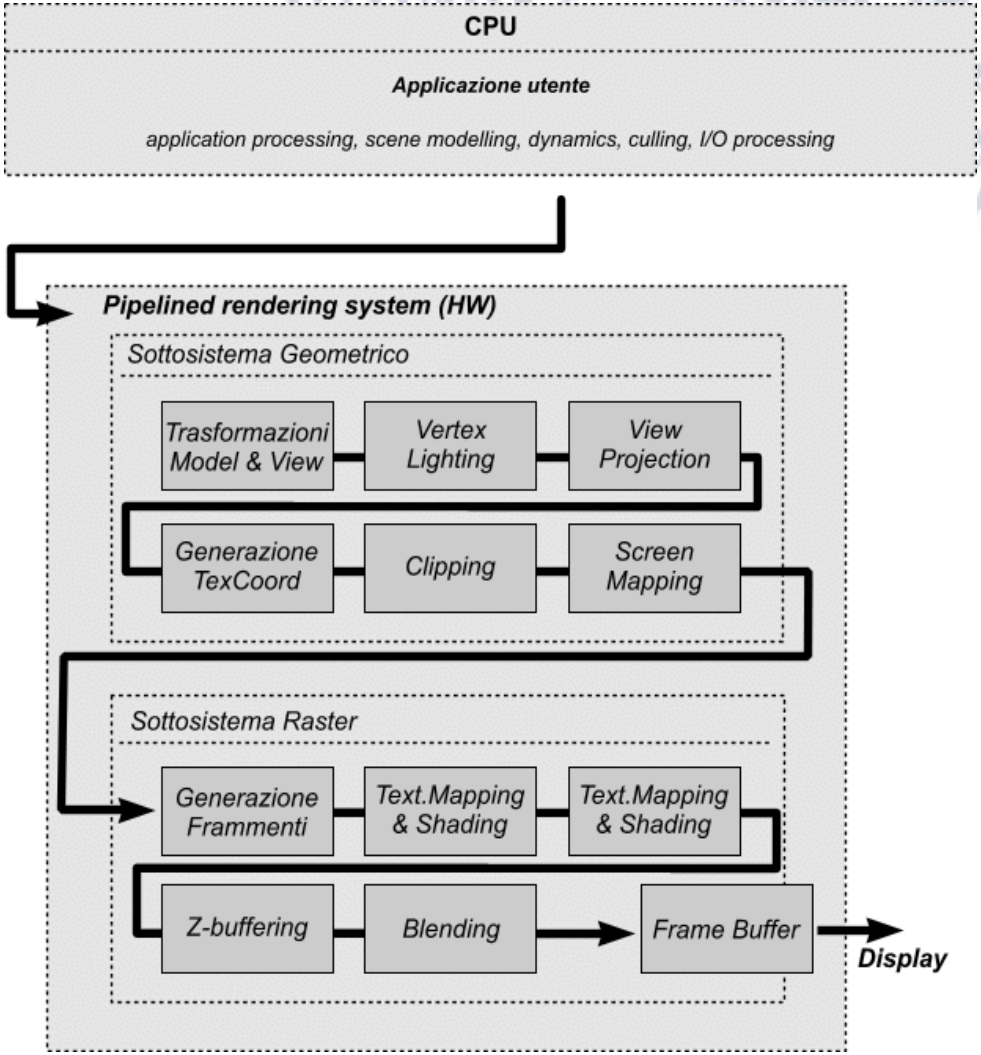
Basic Pipeline

- Tre principali attori in gioco
  - L'applicazione
  - Il sottosistema geometrico
  - Il sottosistema raster
    - che per ogni primitiva di cui ormai si conosce la posizione finale accende i pixel dello schermo da essa coperti in accordo a
      - Colore
      - Texture
      - profondità



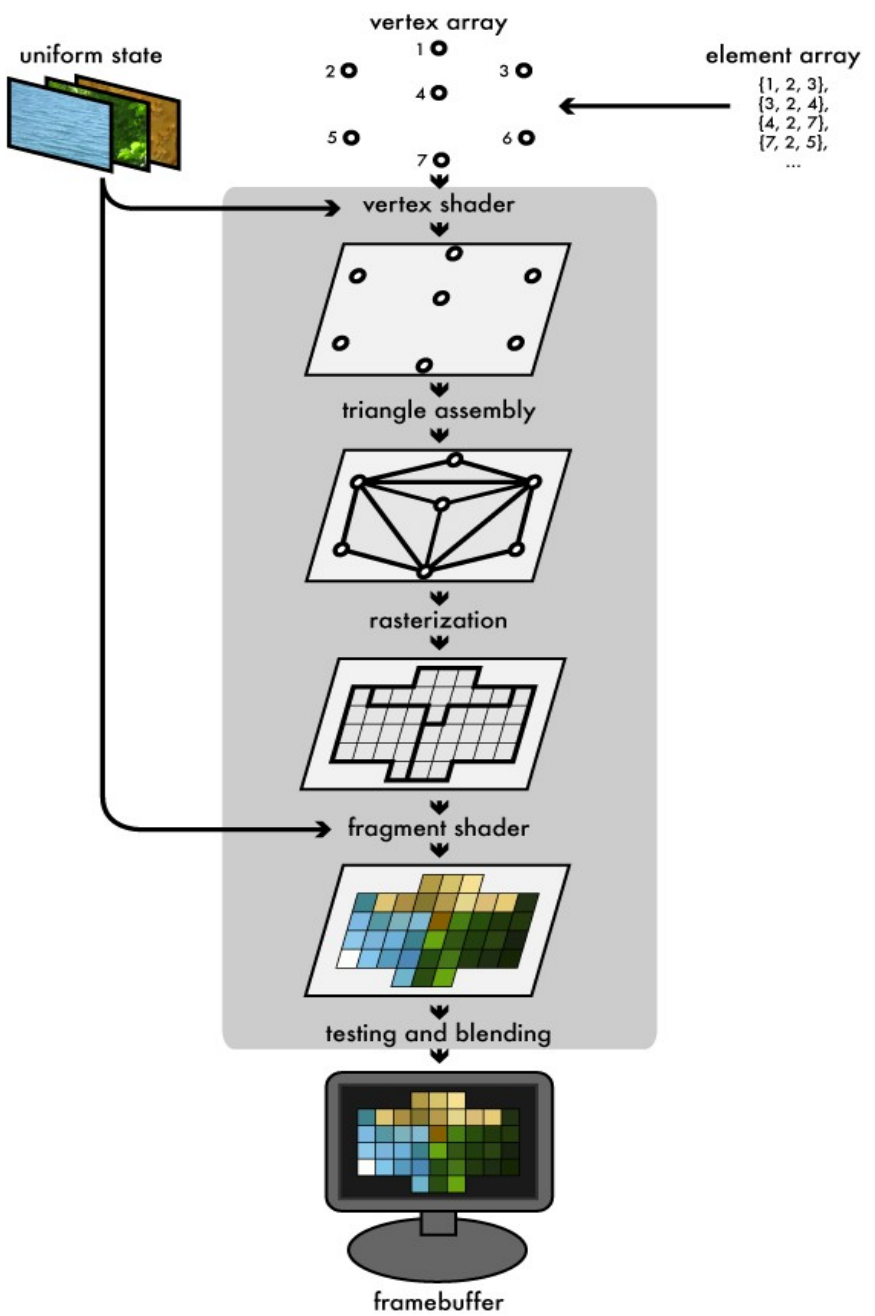
Basic Pipeline

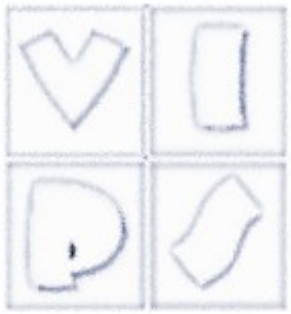
- Nelle prossime lezioni vedremo in dettaglio i vari stadi
- Prima però vedremo le informazioni teoriche preliminari su
  - Modellazione
  - Fisica della formazione delle immagini



Basic pipeline

- Vedremo in dettaglio i vari stadi
- Prima però vedremo le informazioni teoriche relative alla creazione e rendering delle scene
  - Modellazione
  - Fisica della formazione delle immagini
- Questi sono in pratica gli agomenti del nostro corso

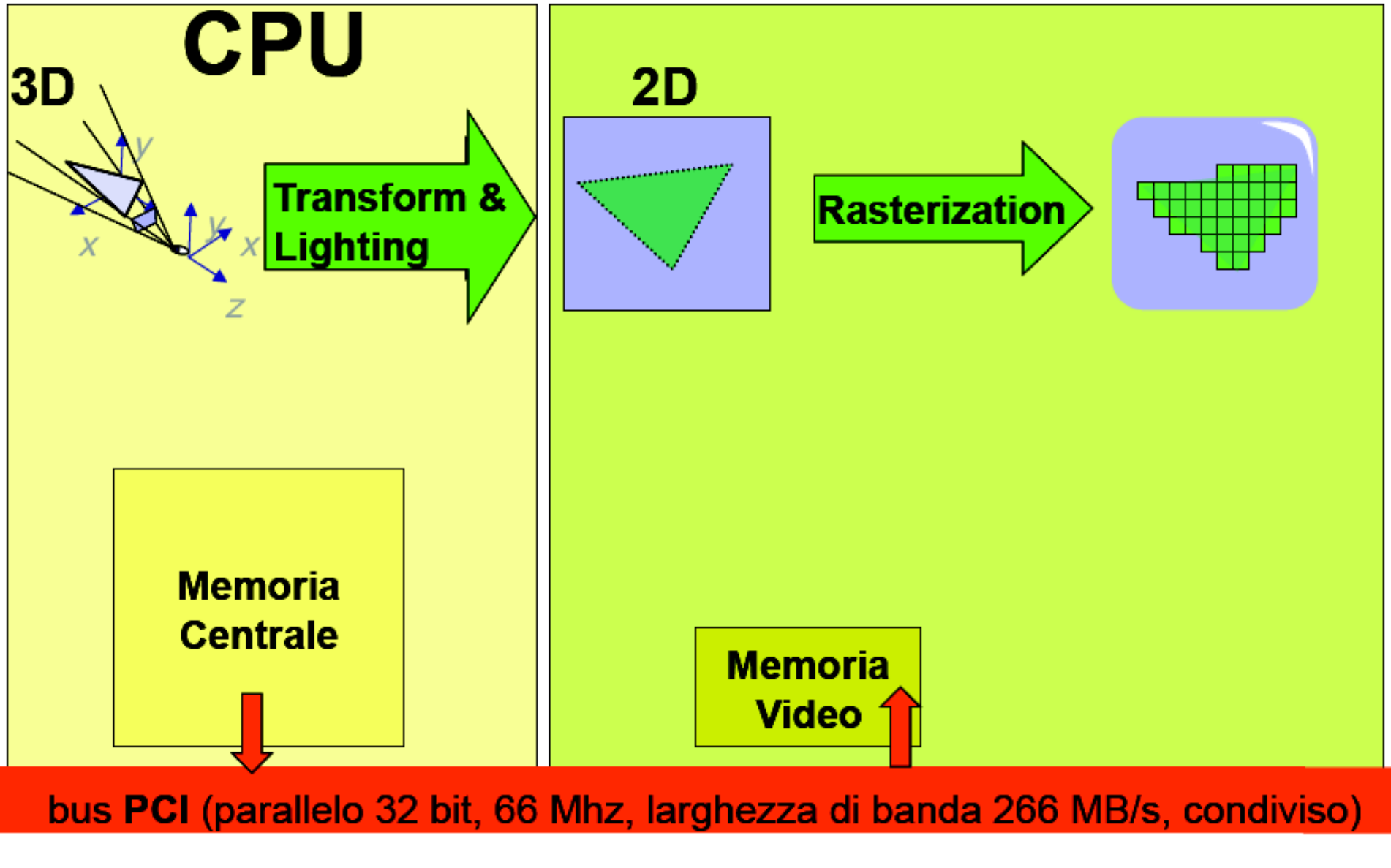




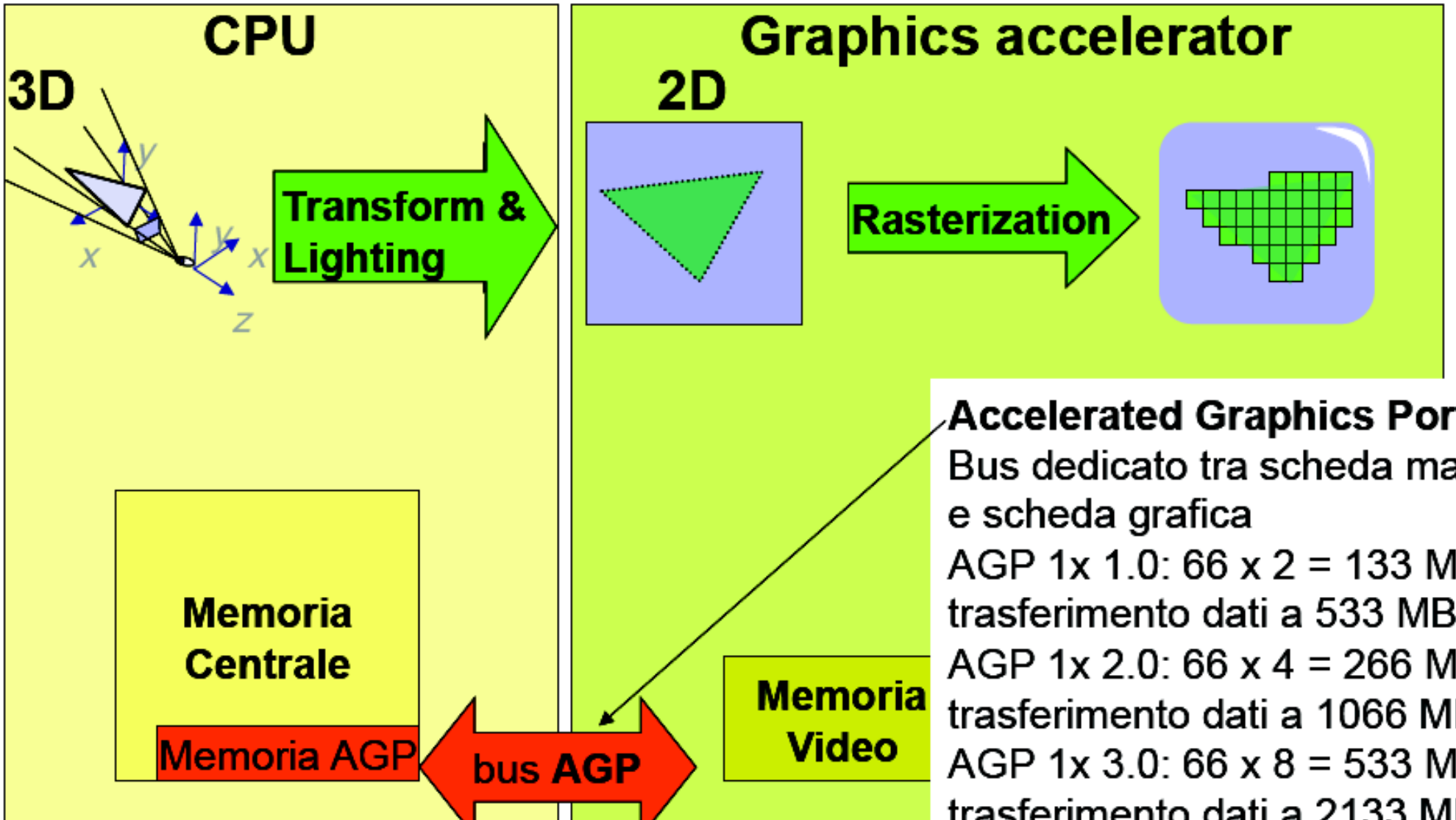
Schede grafiche

- Gestiscono nei sistemi moderni interattivi tutta la parte di pipeline del sottosistema raster+geometrico
- Non è sempre stato così
- Oggi possono fare molto di più e sono in pratica sistemi di calcolo parallelo
  - Si possono implementare algoritmi di rendering più complessi della pipeline standard
  - Si può fare calcolo generico (GPGPU)

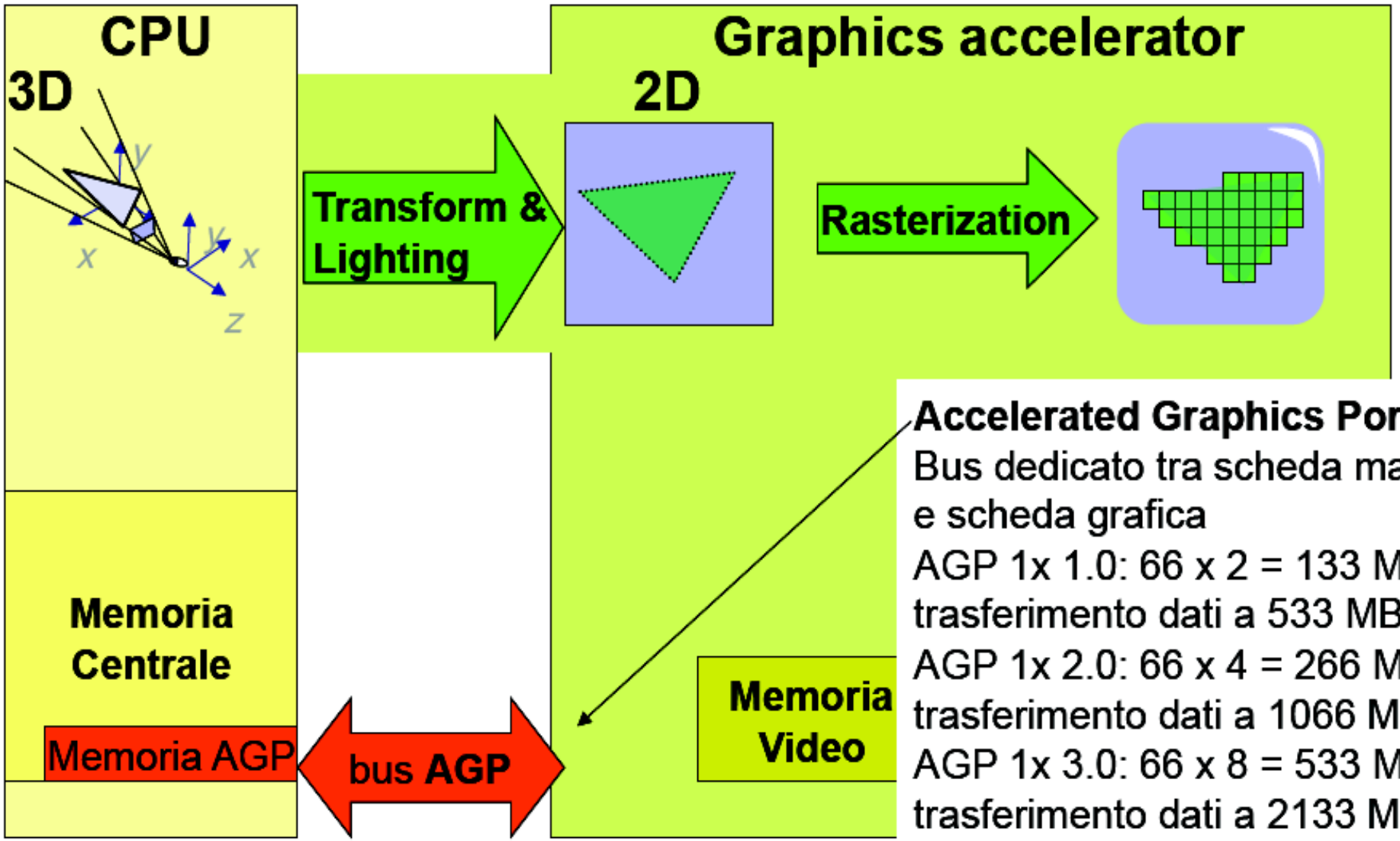
- 1995-1997: 3DFX Voodoo



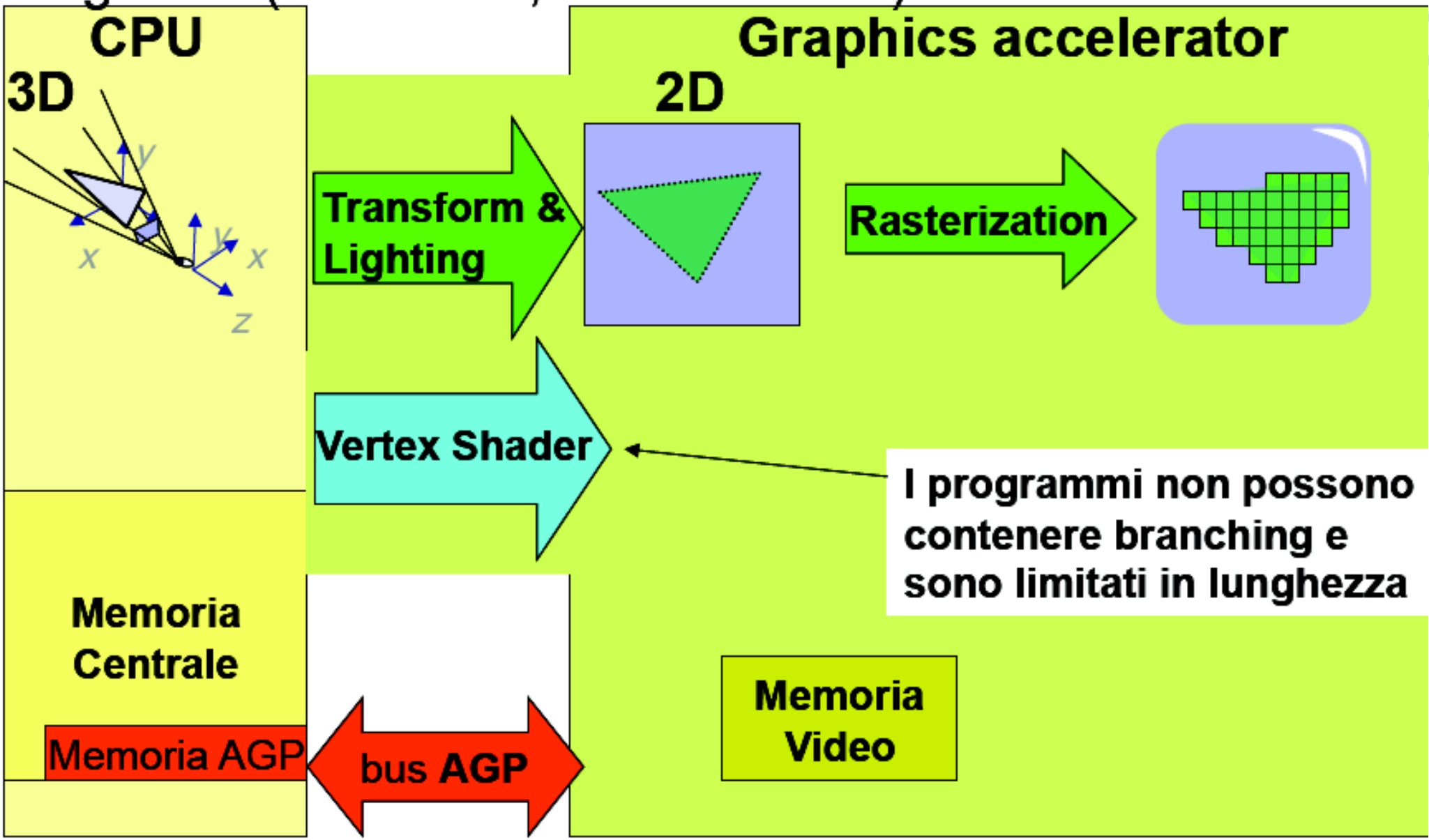
- 1998: NVidia TNT, ATI Rage



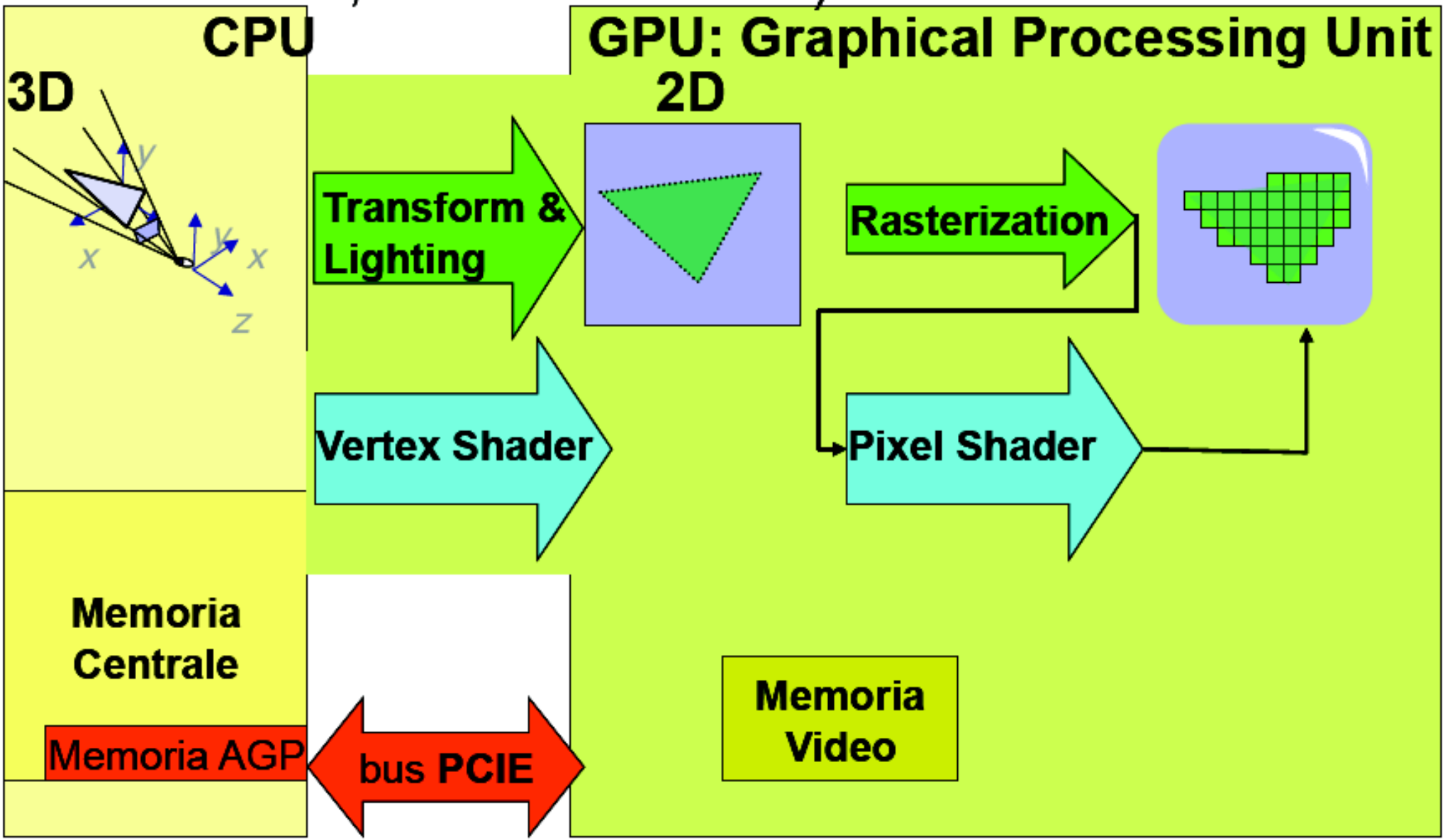
- 1998: NVidia TNT, ATI Rage



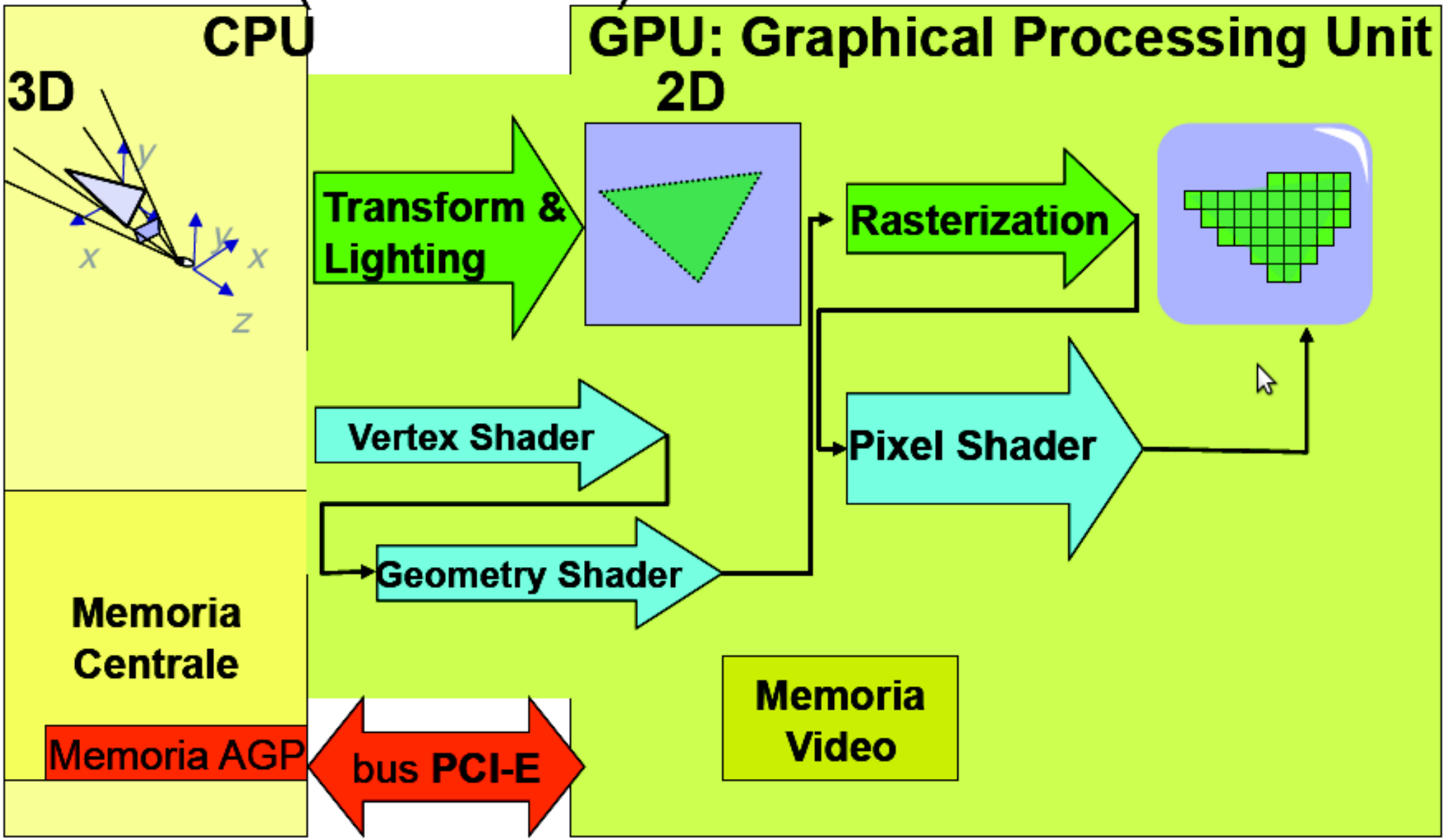
- 2001: Vertex Shader. Programmare il chip della scheda grafica (GeForce3, Radeon 8500)



- 2004-5: PCI-Express, branching negli shader (GeForce 6800-7800, ATI Radeon 9200)

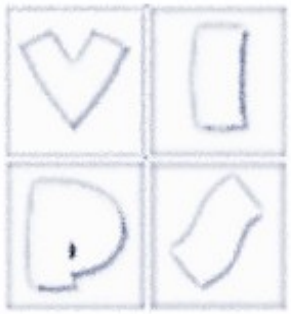


- 2007: geometry shader, stesso hardware per tutti gli shaders (NVidia 8800)



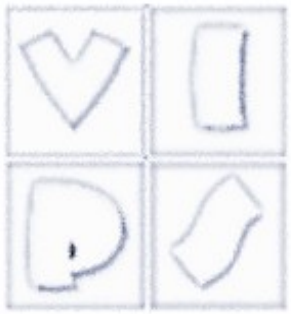
Livello applicazione

- La pipeline descritta che vedremo in dettaglio è controllata da un programma (videogioco, visualizzazione, ecc) che gestisce gli oggetti della scena, i punti di vista, le rappresentazioni geometriche, gli eventi
- Di solito tutto in CPU
- Si possono usare algoritmi e strutture dati opportune per lo svolgimento efficiente di operazioni e librerie che gestiscono le scene e molto altro
- Strumenti di authoring di alto livello che userete gestiscono tutto e utilizzano le librerie in modo trasparente per l'utente che può accedere alle chiamate relative dagli script



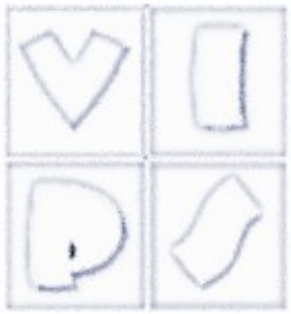
Livelli gerarchici/grafi

- Gli oggetti modellati possono essere inseriti in strutture gerarchiche per semplificare la gestione
- Questo è spesso implementato nelle librerie di più alto livello sopra openGL (es. OSG, java3D, ecc.).
- Distinguiamo
  - Gerarchie di oggetti: in tal caso gli oggetti vengono distribuiti su un albero o grafo diretto aciclico. Esplicitano relazioni di contenimento (la bottiglia nel frigo nella cucina nella casa nella città ...) oppure in animazione agevolano il calcolo della postura di oggetti composti
  - Gerarchia della scena: (scene graph) in tal caso la scena è descritta da una gerarchia che contiene sia gli oggetti da disegnare, che alcuni stati che determinano come disegnarli (trasformazioni geometriche, etc..). Il rendering si basa sulla visita del grafo



Riferimenti

- Scateni et al. Cap. 2,3



Domande di verifica

- Quali sono i possibili tipi di rappresentazione di una scena 3D
- Si indichino le principali tecnologie di display
- Si indichino le principali tecnologie di stampa 3D
- Quali problematiche esistono per l'interazione con scene 3D?
- Come è in genere strutturata un'applicazione interattiva 3D e come in genere sfrutta l'hardware grafico?