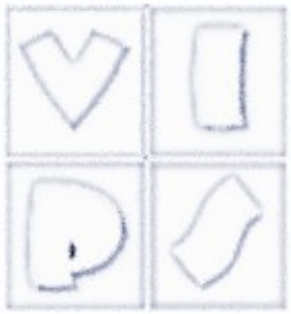


Grafica al calcolatore

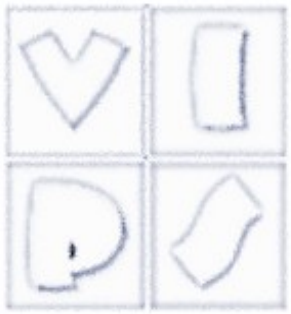
-

Computer Graphics

Andrea Giachetti

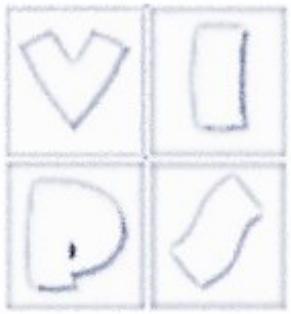
Department of Computer Science, University of Verona, Italy

andrea.giachetti@univr.it



Orario

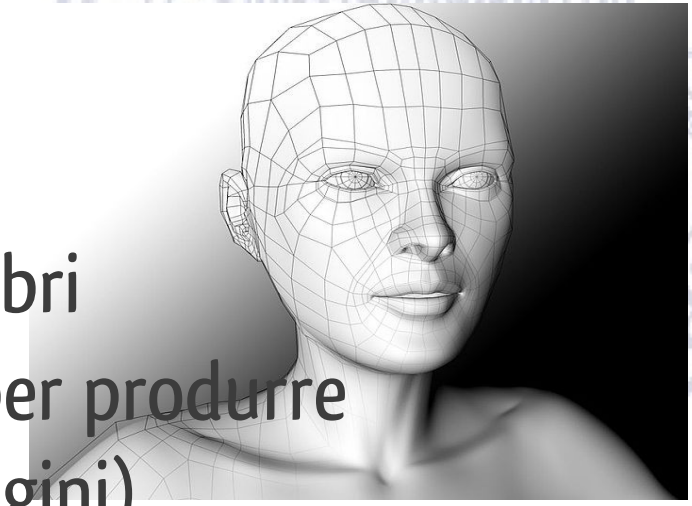
- Decente
- Andrea Giachetti – stanza 1.86
  - andrea.giachetti@univr.it
- Orario
  - 12/10 14-18
  - 19/10 14-18
  - 16/11 14-18
  - 23/11 14-18



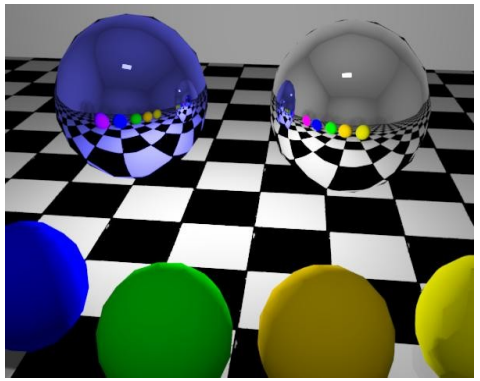
Testi di riferimento

- **Lucidi del corso**
- R. Scateni, P. Cignoni, C. Montani, R. Scopigno, Fondamenti di graca tridimensionale interattiva, McGraw-Hill, 2005
- E. Angel, Interactive Computer Graphics with OpenGL, 3rd edition, Addison Wesley 2003
- S.R. Buss, 3D Computer Graphics, Cambridge University Press, 2003.

Grafica al calcolatore

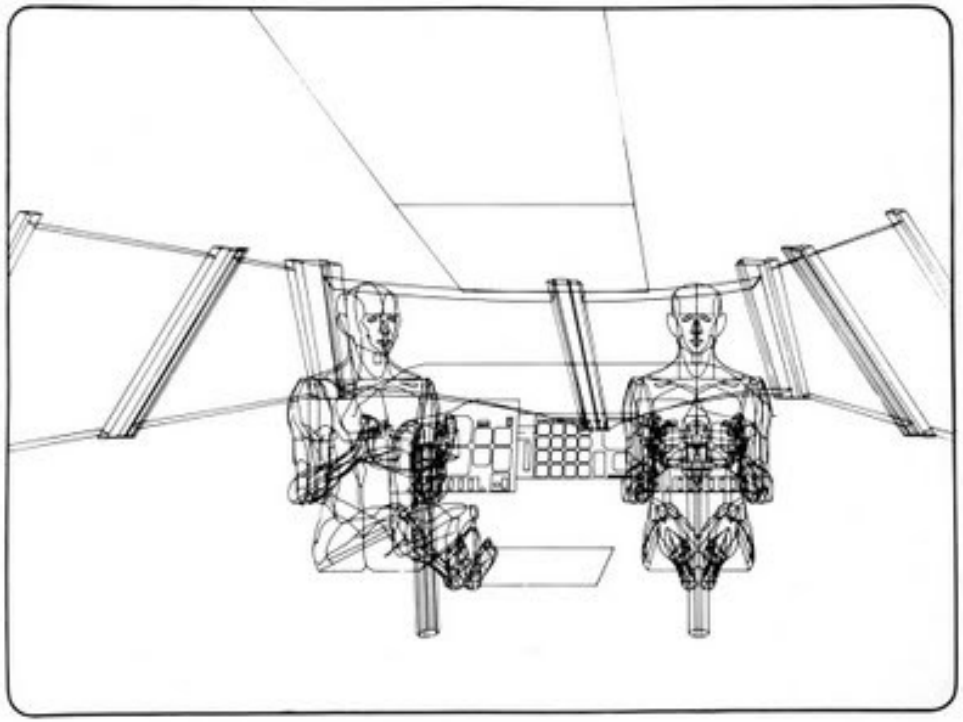


- Che cos'è?
  - Risposta meno ovvia di quanto sembri
- Intuitivamente: uso di un calcolatore per produrre un'immagine (o una sequenza di immagini)
  - Non necessariamente realistica o 3D
  - Non necessariamente interattiva
  - Ma le cose sono più complicate
- Per capire meglio vediamo un po' di storia
  - Nasce con i primi display per i calcolatori



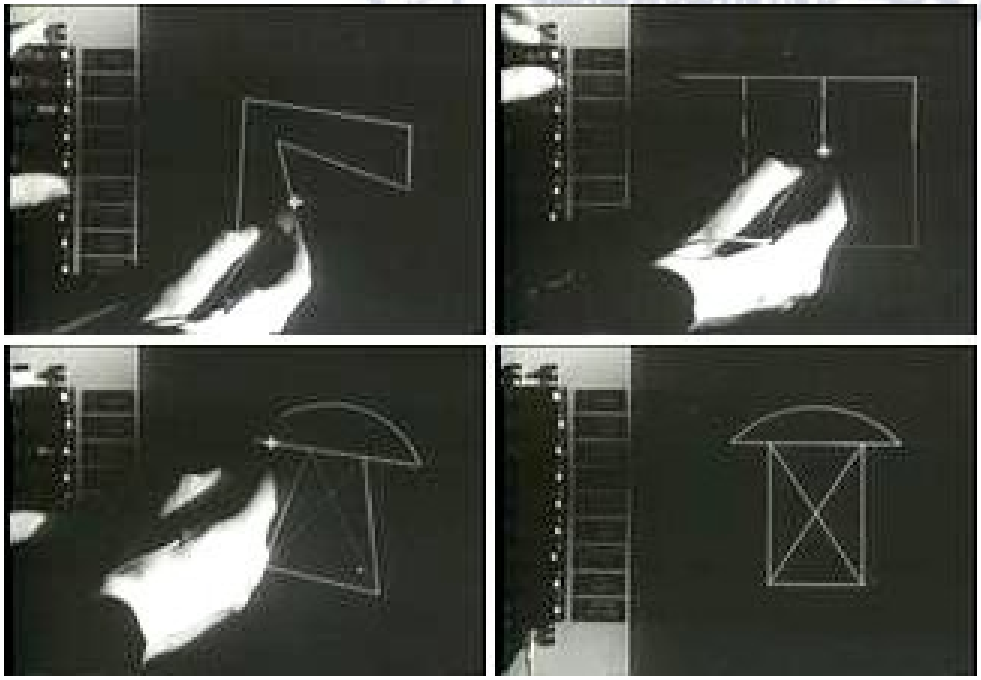
Storia

- 1960
  - William Fetter introduce il termine Computer Graphics per descrivere la ricerca che stava conducendo alla Boeing. Modello 3D del corpo umano per progettare la carlinga degli aerei.
  - C'è quindi l'idea della **modellazione 3D**
  - **E' una parte rilevante della moderna CG**



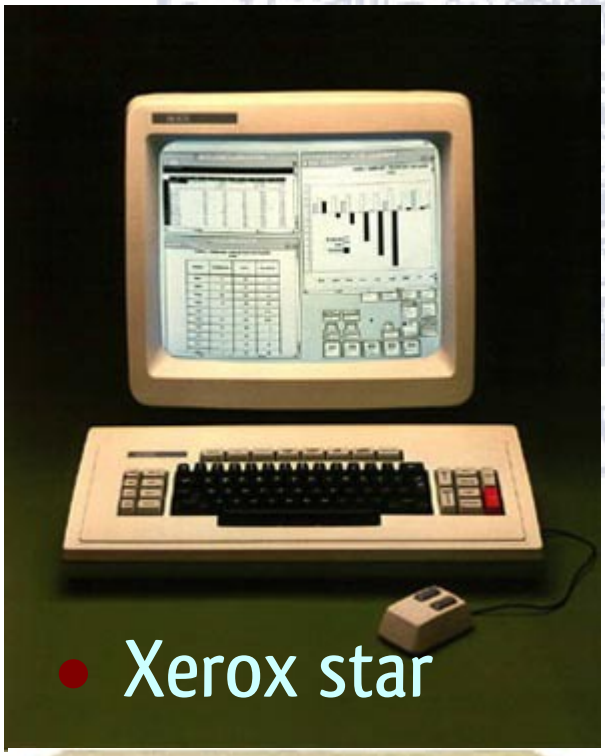
Storia

- 1963
  - Nascita della Computer Grafica interattiva: sistema sketchpad di Ivan Sutherland
  - In questo caso si tratta della prima **interfaccia grafica** interattiva
- Negli anni sessanta nascono i primi terminali grafici e giochi, si impara a disegnare sullo schermo 2D



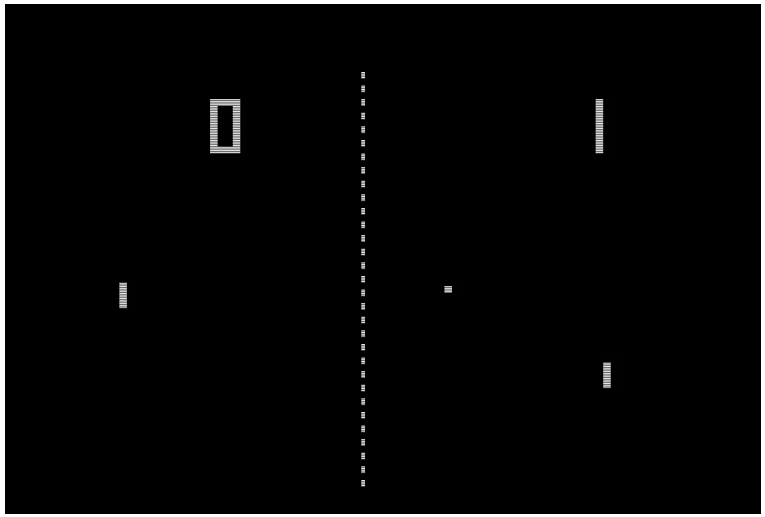
Storia

- Negli anni settanta nascono le moderne interfacce grafiche interattive dei computer (WIMP)
- La grafica interattiva, in questo caso 2D diventa parte integrante del sistema di interazione uomo-macchina



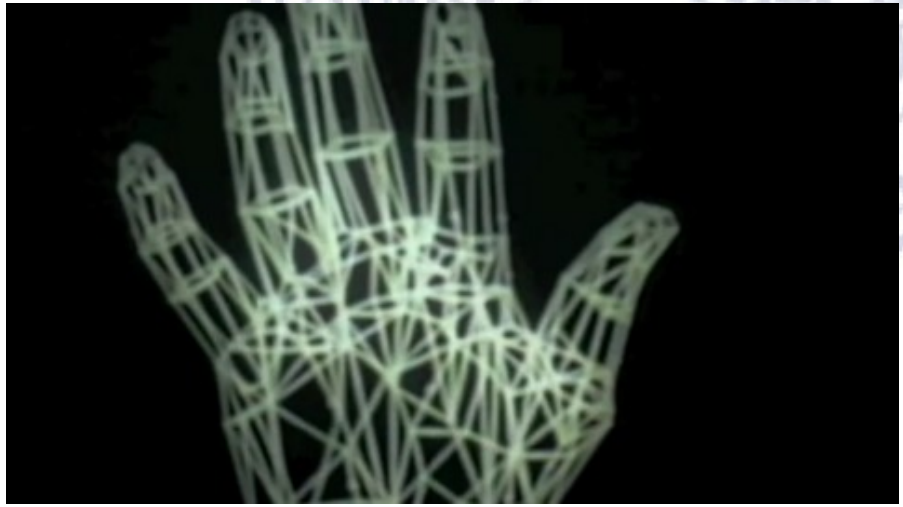
Storia

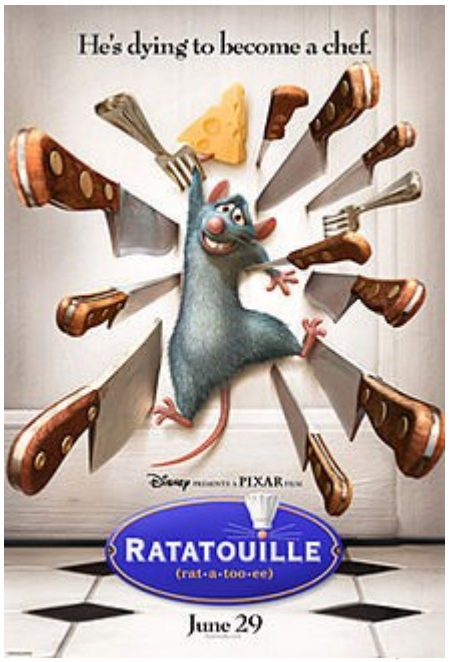
- 1961
  - Steve Russell at MIT crea il primo video game, Spacewar
- 1972
  - Nasce il videogioco Pong (Atari).
- Anche oggi una delle maggiori applicazioni della grafica interattiva è nel mondo dei videogiochi

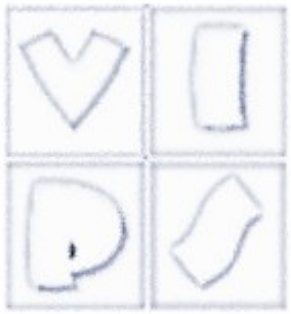


Storia

- Negli anni settanta nascono gli algoritmi per creare immagini da modelli 3D (rendering)
- 1972
  - Catmull (Univ. Utah) crea la prima animazione di grafica 3D
  - Modello della sua mano, 350 poligoni
  - Catmull diventerà un cofondatore della Pixar



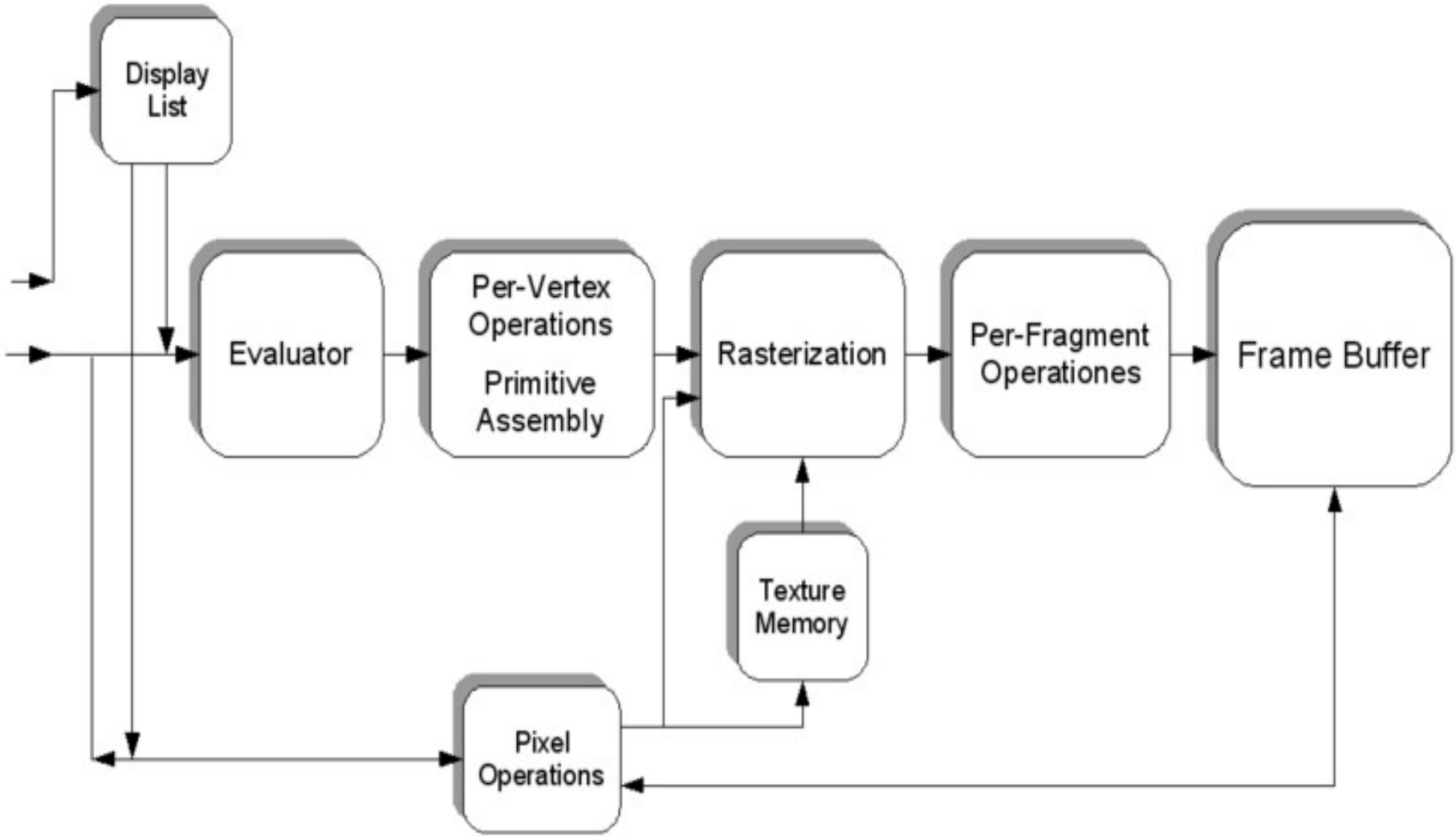




Storia

- Gli algoritmi per creare linee raster, riempire poligoni, proiettare oggetti 3D su telecamere virtuali vengono via via sviluppati negli anni '60-70-80
- Cuore della grafica 3D e di questo corso
- Si creano le pipeline di **rendering** per creare velocemente immagini sullo schermo a frame rate interattivi
- Si creano standard e implementazioni di sistemi grafici e si arriva alla situazione attuale
  - 1992 Silicon Graphics crea lo standard OpenGL
  - 1995 Microsoft rilascia Direct 3D

Pipeline grafica



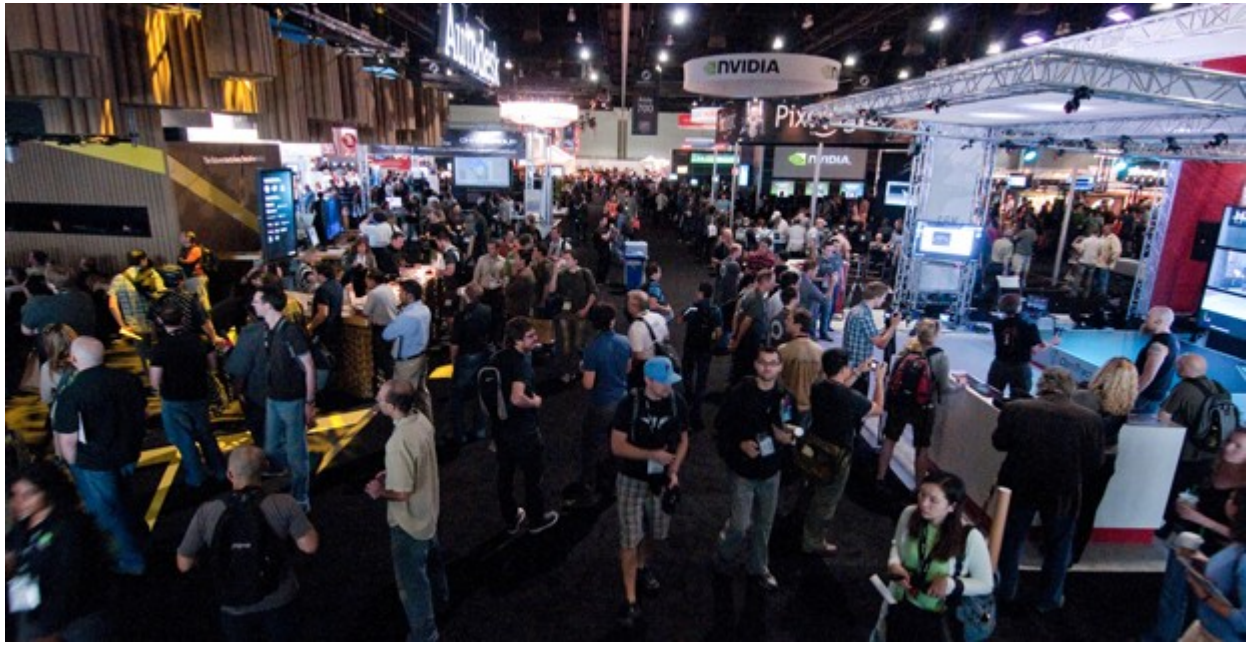
Storia

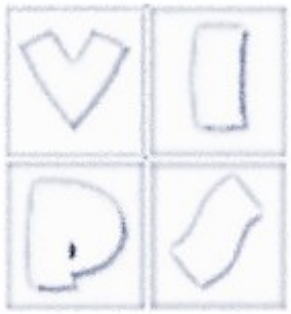
- Le operazioni grafiche vengono implementate su hardware specifico
- Inizialmente grafica raster calcolata su CPU, poi (doppio) buffer per mantenere le immagini (doppio perché il calcolo può essere lento rispetto al refresh dello schermo)
- 1985 Commodore Amiga, uno dei primi home computer con una GPU
- 1987 primo PC Ibm con operazioni 2D hardware
- 1995: prime schede video per PC con pipeline grafica 3D (S3 Virge)

- 1999 Nvidia GeForce 256 prima scheda con transform & lightning engine

Storia

- 1969, the ACM initiated A Special Interest Group in Graphics (SIGGRAPH)
- 1973 SIGGRAPH organizza la prima conferenza internazionale



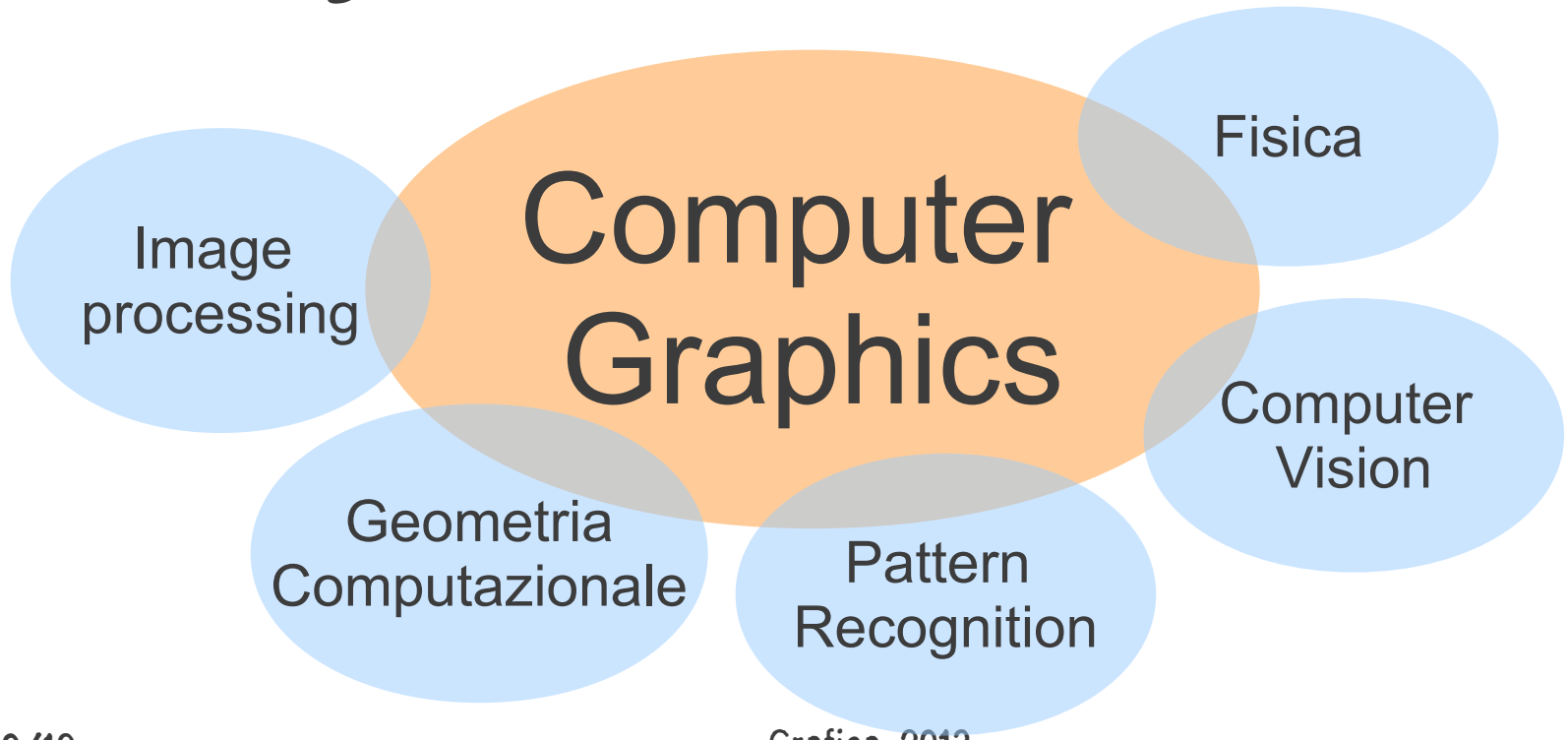


Quindi?

- Cos'è la “computer graphics”? Un po' tutto questo:
  - Creazione immagini 2d sintetiche e animazioni
  - Modellazione 2D, 3D, anche con comportamenti fisici
  - Computer Aided Design
  - Rendering delle scene, cioè creazione delle immagini simulando la proiezione ottica delle scene sulla camera
  - Animazione
  - Interfacce grafiche dei computer
  - Realtà virtuale
  - Enhancement video televisivo
  - Visualizzazione scientifica e dell'informazione

Una definizione semplice

- La disciplina che studia le tecniche e gli algoritmi per la rappresentazione visuale di informazioni numeriche prodotte o semplicemente elaborate dai computer (da Scateni e al.)
- E' quindi legata a molte altre discipline

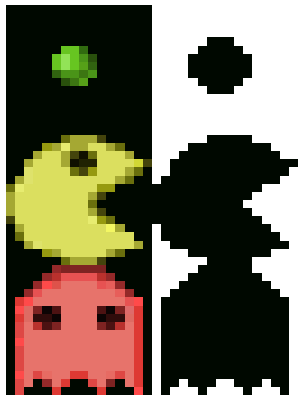


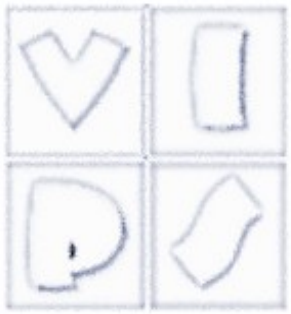
Computer Graphics vs Computer Vision

- In senso generale grafica è l'opposto di “image understanding” o “computer vision”
  - Nel primo caso si passa da immagini a parametri, a interpretazione
  - Nel secondo si crea un'immagine da un input parametrico
  - Quindi sono grafica tutti i sistemi informatici che creano e usano immagini sintetiche

2D vs 3D

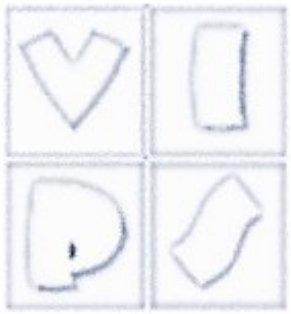
- 2D
  - Generazione e miglioramento di immagini, disegno tecnico al PC, cartografia: occorre generare e rappresentare modelli 2D e visualizzarli su schermo raster. Anche grafica vettoriale
- 3D
  - Generazione di immagini 2D (o altre forme di visualizzazione per l'occhio umano, es. light fields) a partire da scene (rendering)
  - Progettazione al calcolatore di manufatti reali (CAD)
  - Display di dati volumetrici (es. medici TAC, MRI)





Interattività

- La grafica generata con un calcolatore pu`o o meno essere interattiva, ovvero pu`o o meno permettere ad un operatore esterno di interagire in tempo reale con uno qualsiasi (o tutti) dei parametri della rappresentazione grafica
- Nel caso di grafica interattiva si richiede una risposta in tempo reale ai comandi dell'operatore (frame rate 10fps); questo implica
  - necessità di hardware particolari (schede grafiche acceleratrici, processori potenti, molta memoria)
  - un modello semplificato di resa grafica (magari non "fotorealistica")



Fotorealismo

- Uno dei principali scopi della grafica al calcolatore sta nel creare algoritmi per creare dai modelli di oggetti reali immagini che sembrano le foto degli oggetti reali. Per ottenere il fotorealismo occorre:
  - Simulare numericamente l'interazione luce materia e la formazione dell'immagine
  - Oppure usare “trucchi” per “dare l'impressione” realistica. Nella grafica interattiva è tipico

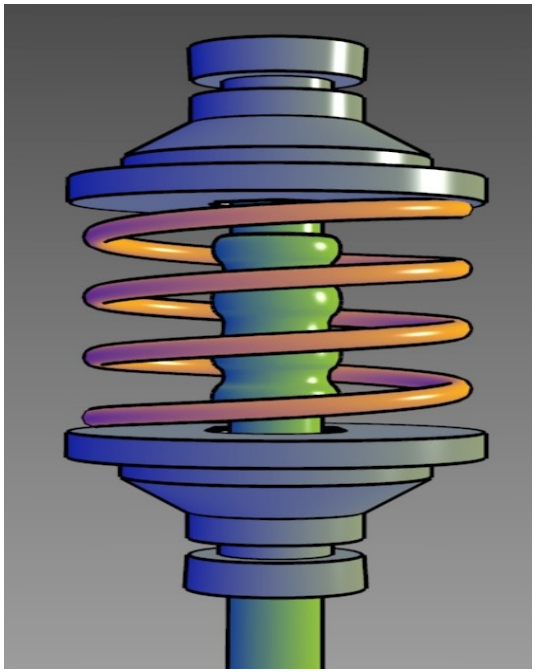
Fotorealismo

- Richiede come vedremo alogritmi complessi
- Ed anche modelli complessi

Non sempre si cerca il fotorealismo

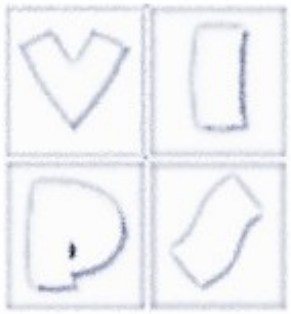
- Non sempre lo scarso fotorealismo è un difetto dovuto a limitate risorse computazionali, interattività, ecc.
- Può essere utile evidenziare contorni, silouhette per il disegno tecnico, ad esempio
- O simulare tratteggio artistico
- O evitare il confronto con la realtà proponendo caratteristiche fantasiose, come nel cinema di animazione
- Si parla di tecniche NPR (Non-Photorealistic Rendering) per generare automaticamente effetti particolari

Non sempre si cerca il fotorealismo



Non-photorealistic = Off

Non-photorealistic = On



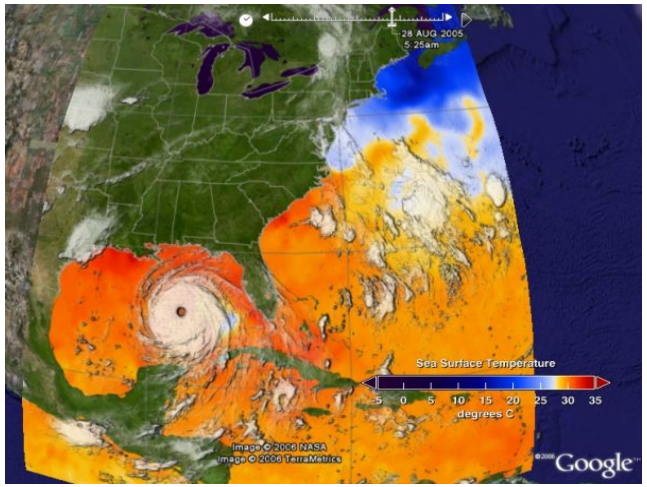
In questo corso

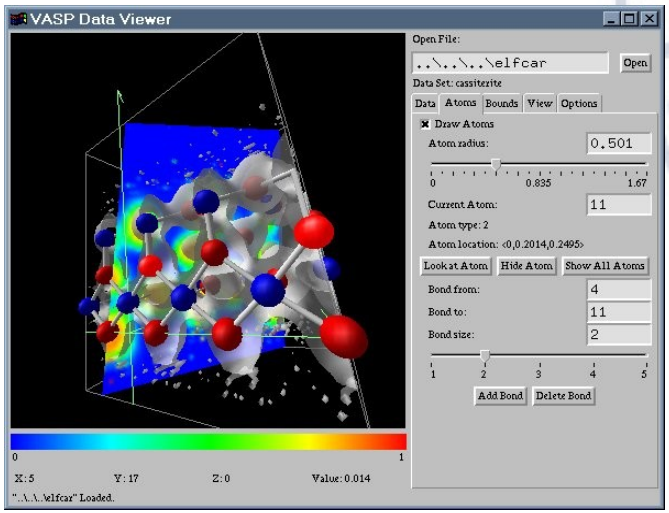
- Ci occuperemo quasi esclusivamente di 3D
- Vedremo un po' di modellazione e geometria di trasformazione
  - Mesh triangolate
  - Calcolo trasformazioni
  - Proiezione ortografica/prospettica
- Vedremo il processo di rendering, la creazione delle immagini
  - Modello fisico di illuminazione/telecamera
  - Approssimazioni
  - Trucchi per simulare fotorealismo
- Vedremo la pipeline grafica (di OpenGL)
  - Rasterizzazione, operazioni parallelizzate in hardware

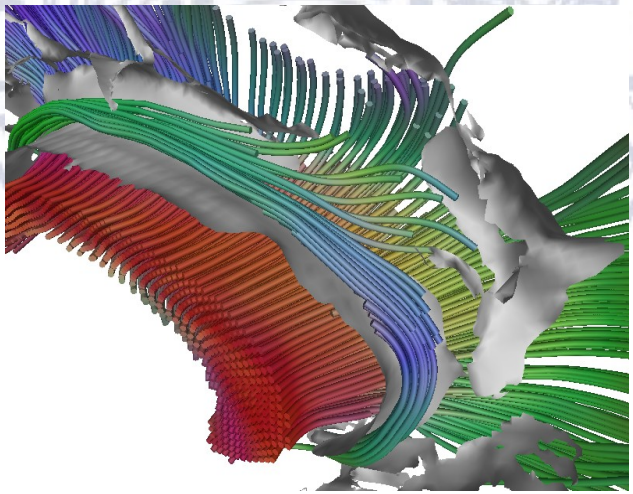
Altre applicazioni di interesse

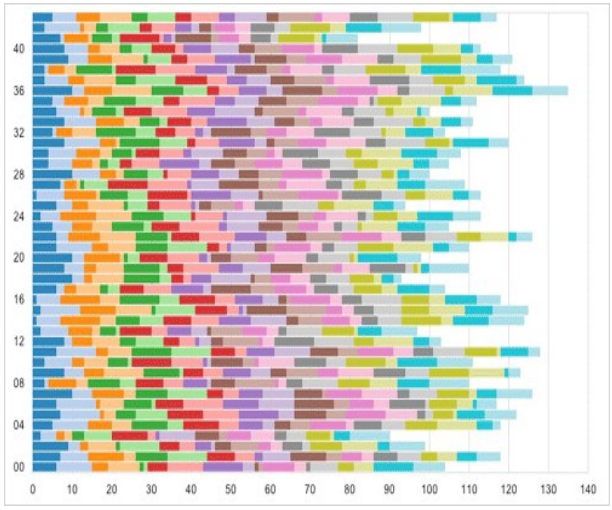
- Visualizzazione scientifica
  - Uso della grafica (2d-3D) per comunicare efficacemente informazione di misure o simulazioni
- Visualizzazione dell'informazione:
  - creazione di modelli “mentali” utili per rappresentare nello spazio dati astratti
- Realtà virtuale o aumentata e interazione uomo macchina
  - Interfacce naturali per comunicare con i computer o simulare attività reali

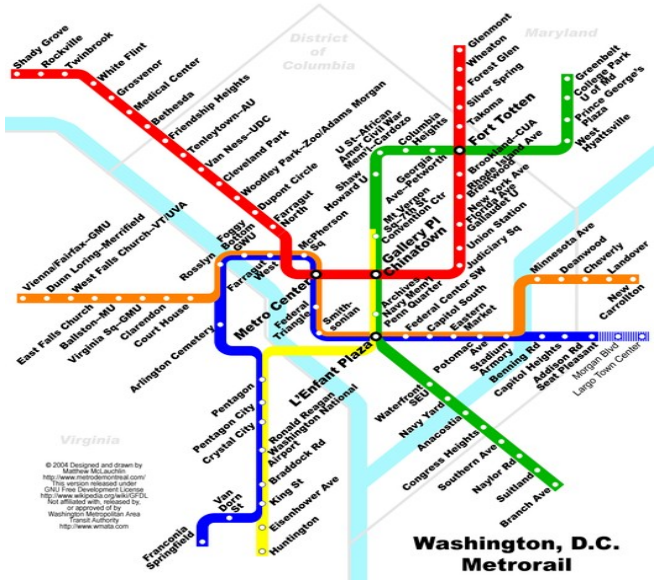
Visualizzazione

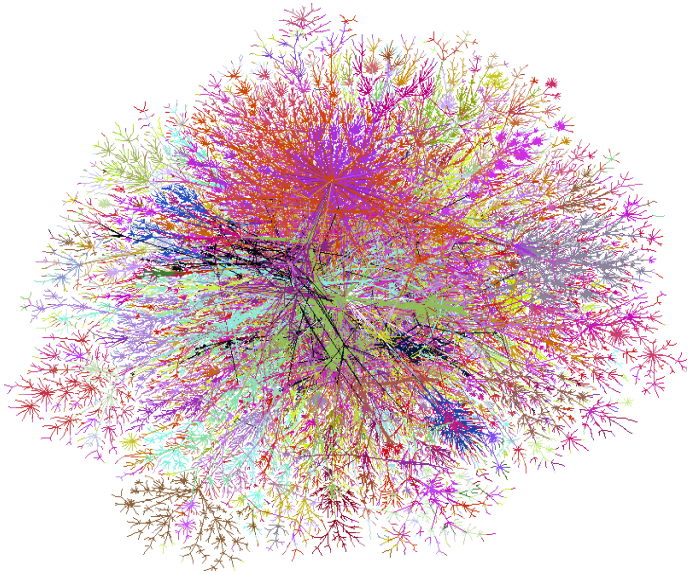










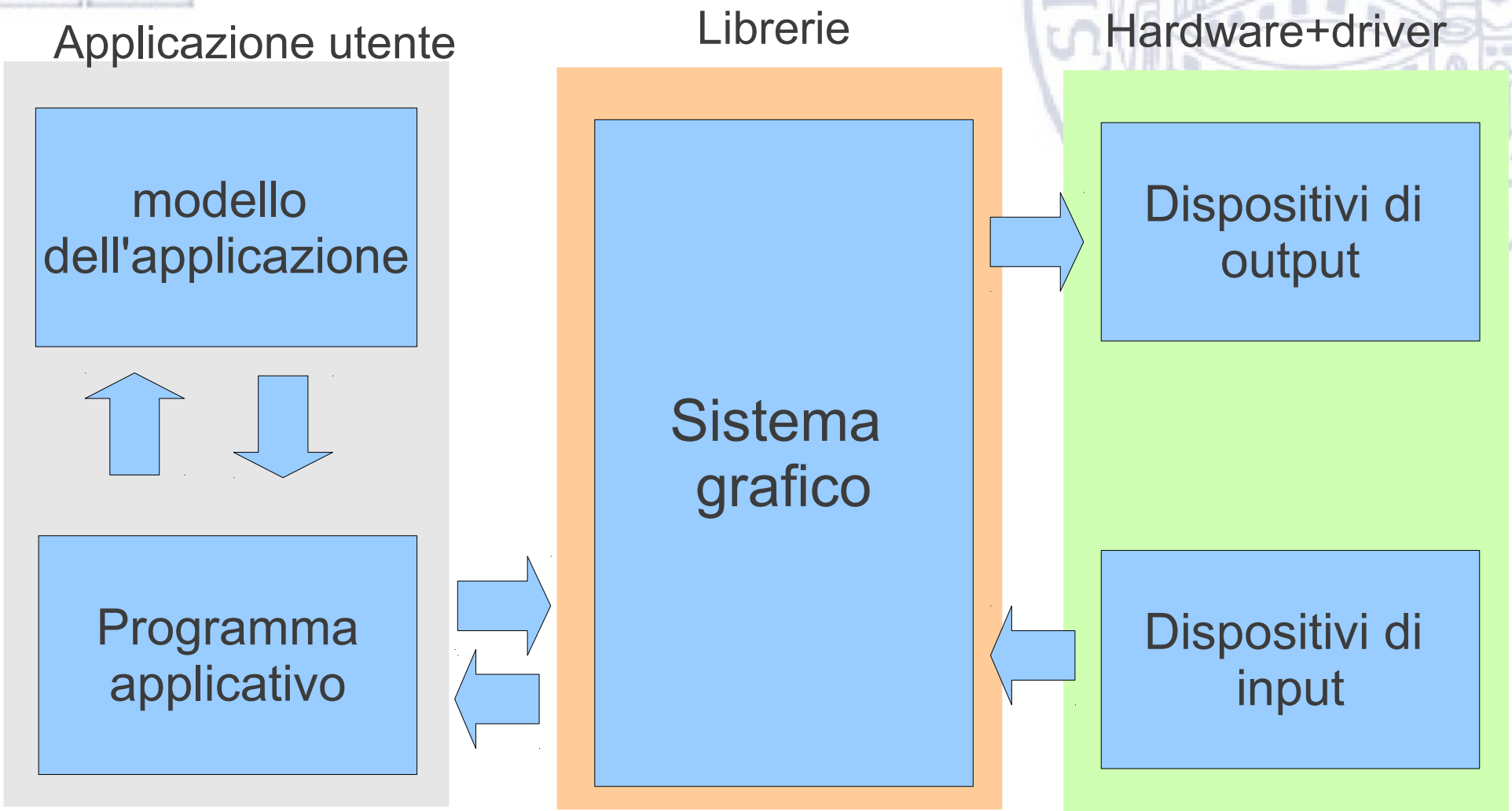


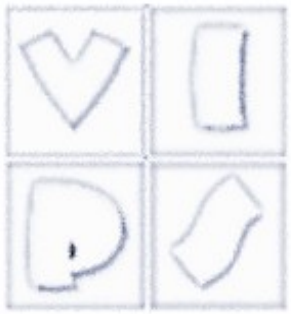
Realtà virtuale e aumentata

Schema di un'applicazione grafica

- Vi è una descrizione di qualche tipo (procedurale o meno) del mondo che deve essere rappresentato. La produzione di tale descrizione (modello) prende il nome di modellazione.
- Da tale descrizione si ottiene una immagine bidimensionale; tale processo è chiamato globalmente rendering
- La sequenza di procedure ed algoritmi che implementano il rendering prende il nome di pipeline grafica; la studieremo nel dettaglio nel seguito
- L'immagine ottenuta viene quindi visualizzata sullo schermo (in applicazioni interattive, per esempio) o salvata su file
- Se c'è l'interazione occorre naturalmente progettarla!

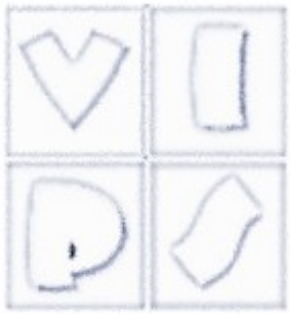
In pratica


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



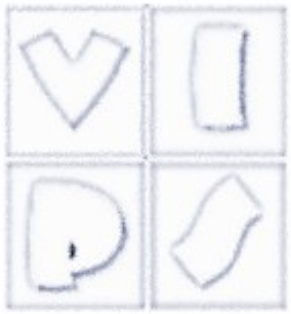
Applicazione

- L'applicazione utente contiene il modello dell'applicazione e la sua implementazione che usa le librerie per accedere all'input e all'output dai relativi dispositivi
  - Mouse, controller, ecc
  - Display raster
- La pipeline di rendering che vedremo è implementata a livello di librerie e sostanzialmente demandata all'hardware grafico



Note

- OpenGL si occupa solo del rendering grafico. Altre librerie per le finestre, l'input, funzioni accessorie
  - GLX, X11, GLU, GLUT
- Oggi si usano spesso librerie di livello superiore, per esempio OpenSceneGraph, Java3D, ecc per modellare a oggetti sistemi grafici interattivi
- Concetto di Scene Graph
  - Struttura dati con schema logico e spaziale della scena
- Standard X3D, evoluzione di VRML per la realtà virtuale

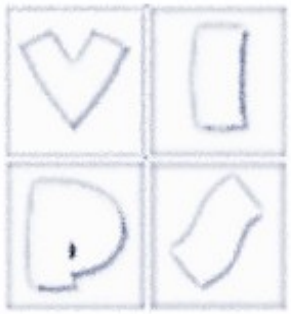


Rendering

Visualizzazione della scena (o *rendering*)

Requisiti dipendenti dalla applicazione di interesse:

- Applicazioni ***interattive***, real-time:
  - Frame rate alto (>10 fps)
  - Tempo di rendering del singolo frame prefissato
  - Si sacrifica la qualità per garantire l'interattività
- Applicazioni ***non interattive*** (computer animation, grafica pubblicitaria)
  - l'obiettivo primario: massima qualità delle immagini di sintesi
  - non si hanno vincoli sul tempo di generazione del singolo frame
  - animazioni calcolate *frame by frame* da PC cluster, ricomposte successivamente nella successione temporale corretta

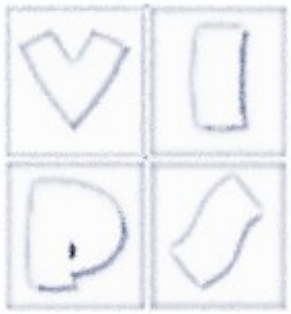


Rendering

Come si implementa la fase di rendering?

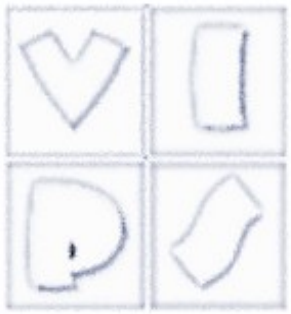
- **Applicazioni interattive**:
  - si avvalgono pesantemente delle moderne **schede grafiche** (HW dedicato al processing di dati 3D)

- **Applicazioni non interattive**:
  - fanno uso di ambienti di rendering più sofisticati e flessibili (ad es. RenderMan), spesso eseguiti SW su cluster di PC



Rendering - API

- API (Application Programming Interface) per la grafica 3D
  - OpenGL, DirectX, ...
- Progettate per grafica 3D interattiva, organizzazione logica funzionale ad una efficiente implementabilità HW
- Efficienza direttamente dipendente dalla possibilità di elaborare in parallelo le diverse fasi del processo di rendering
- Soluzione vincente: suddivisione del processo in fasi indipendenti, organizzabili in pipeline
  - Maggiore parallelismo e velocità di rendering
  - Minore memoria (fasi indipendenti ⇒ memoria locale, non necessario conoscere la rappresentazione dell'intera scena ma solo della porzione trattata)
  -

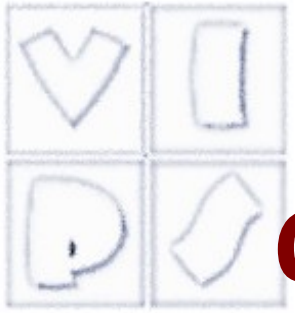


Evoluzione

- GKS (Graphics Kernel System) primo standard europeo per grafica 2D. Poi estensione 3D
- PHIGS Programmer Hierarchical Interactive Graphics System appoggiatto dall'ANSI. Separazione modelling/rendering e modello applicazione/programma applicativo
- Intanto Silicon Graphics crea API proprietaria IrisGL direttamente legata all'architettura, efficiente
- OpenGL versione “aperta” di IrisGL (1992)
  - Fornisce un'interfaccia uniforme all'hardware grafico
  - Emulazione software
- Alternativa: Microsoft Direct 3D (DirectX)

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)

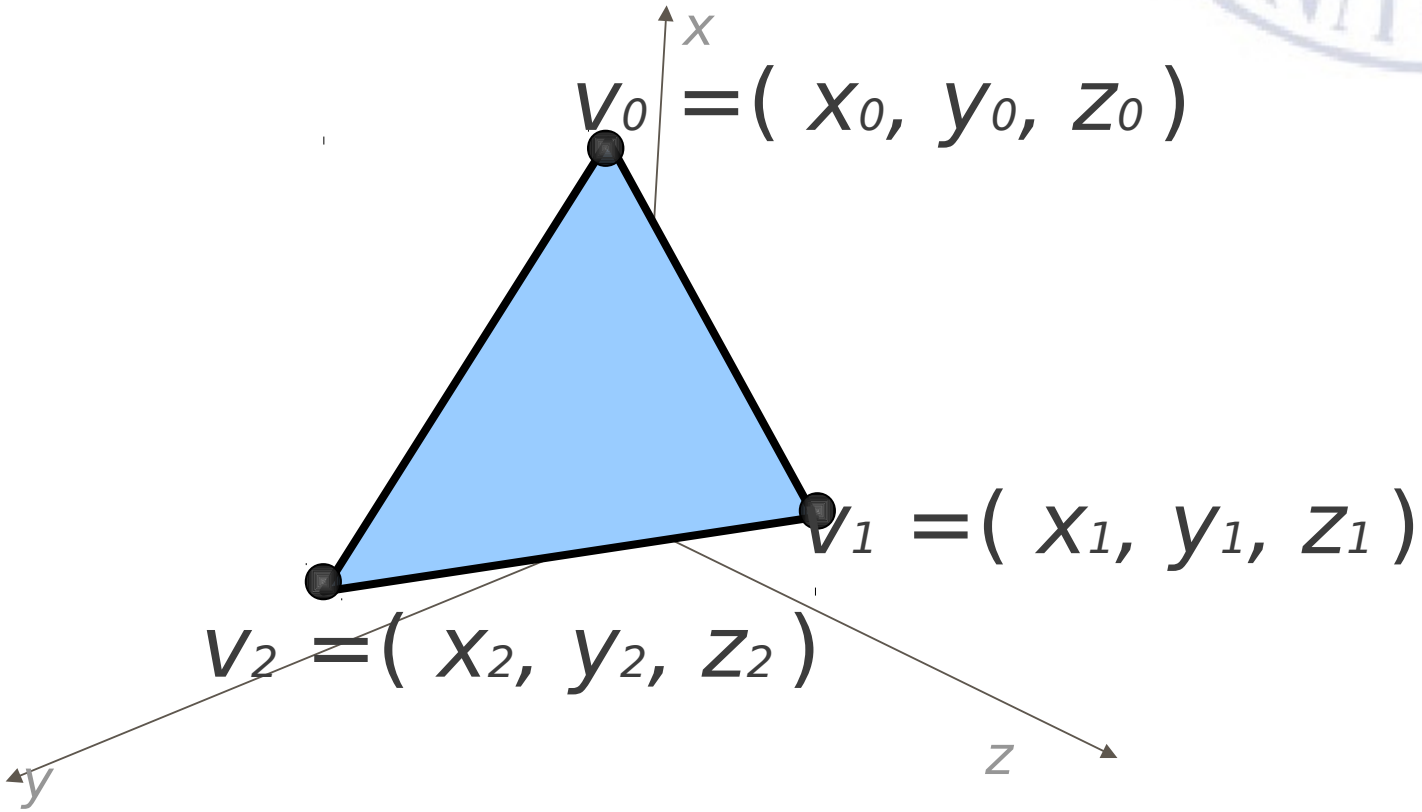


Modello dell'applicazione/programma

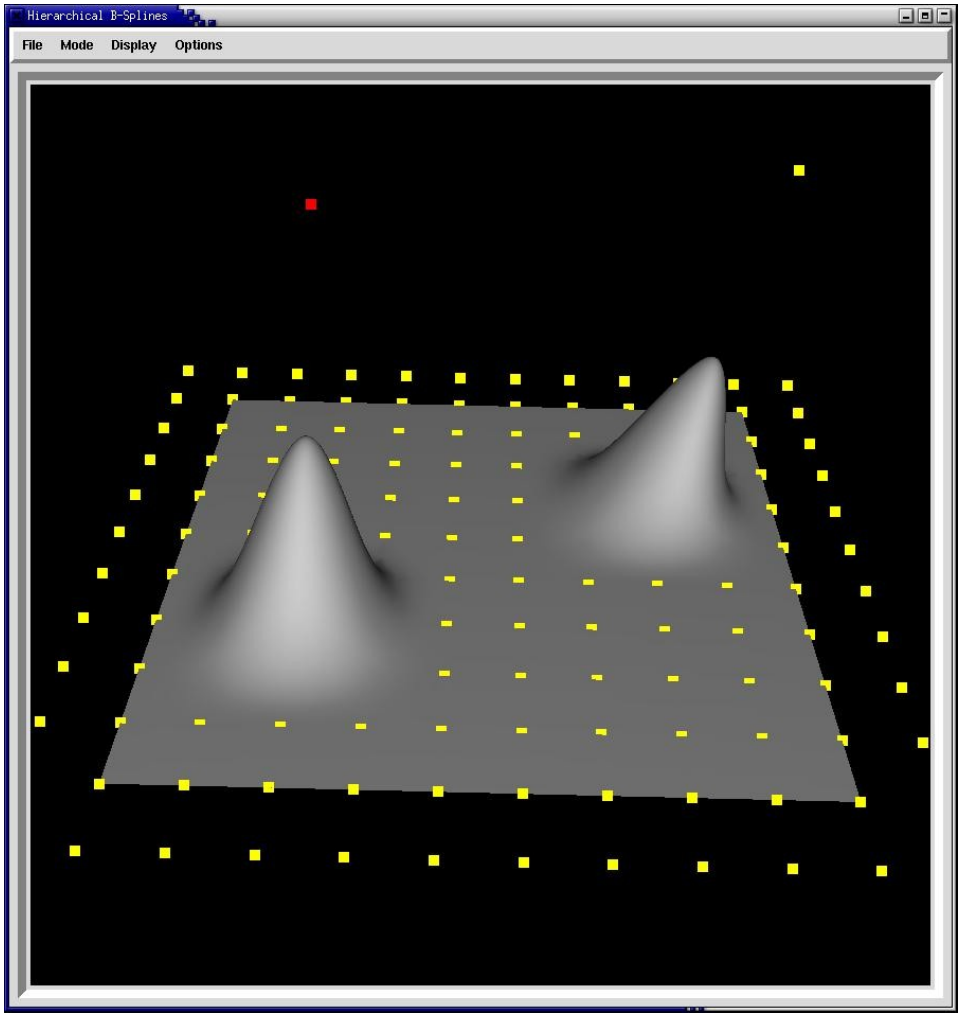
- Nell'applicazione ci saranno le scene (sistemi di riferimento spaziali) e gli oggetti per generare le imamgini
  - Modelli geometrici 3D, tipicamente mesh di triangoli, con coordinate punti e connettività
  - Alternativa: rappresentazioni volumetriche. Tipicamente cubetti pieni/vuoti (voxel) con caratteristiche 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.

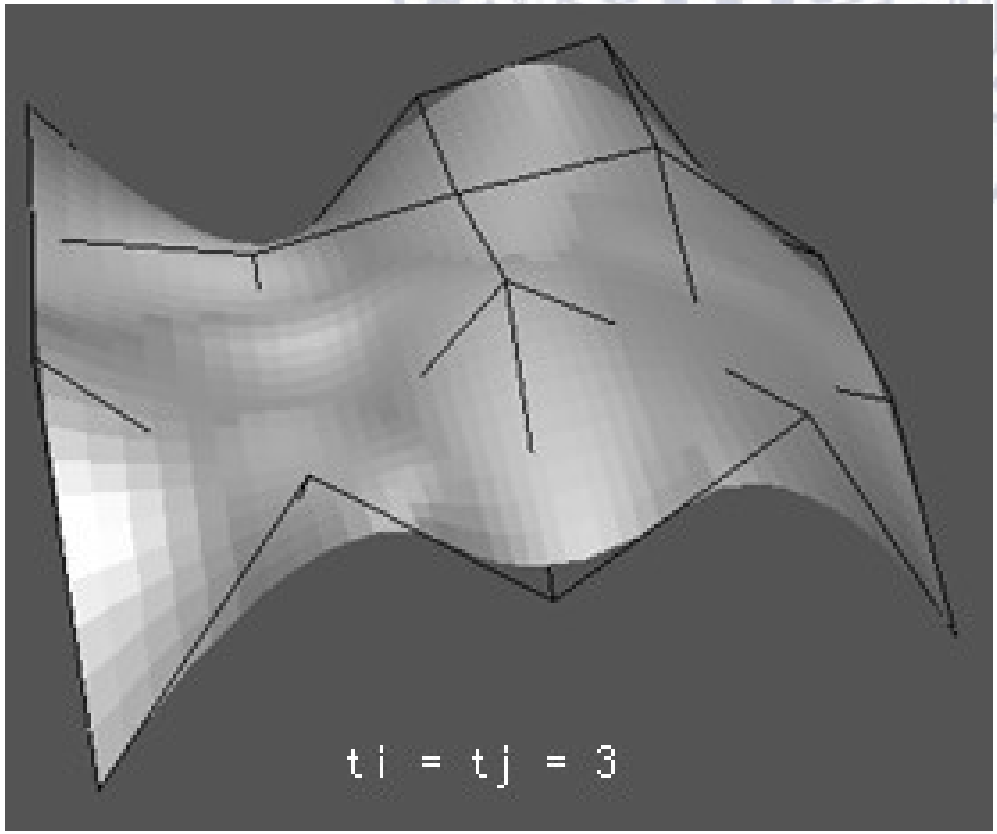
Facce triangolari

- Rappresentazione: **geometria**, ossia coordinate dei vertici



Superfici parametriche - Esempi



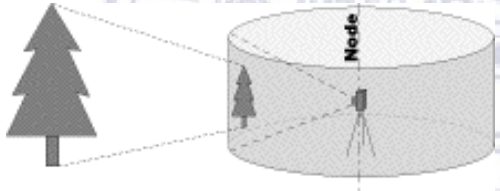


Enumerazione spaziale (volumetrica)

- Codificano direttamente il volume di spazio occupato dall'oggetto, anzichè la sua frontiera
  - Ad es. i modelli basati su voxel (volume element) tipici delle applicazioni medicali (prodotti da TAC e RM)
- Rendering di modelli voxel-based:
  - algoritmi ad hoc (tecniche direct volume rendering)
  - conversione da voxel a rappresentazione per superfici (triangle-based)

Immagini panoramiche

- Una scena 3D o un singolo oggetto rappresentati elaborando una serie di immagini 2D, in modo che sia garantita continuità nella rappresentazione della scena e la scelta interattiva della posizione di osservazione
- Uno dei formati piu' comuni è il QuickTimeVR
- Dal punto di vista centrale l'utente puo' guardare (ma senza spostarsi) in ogni direzione e puo' zoomare a piacimento



pre-processing

real-time

Light field - Esempio

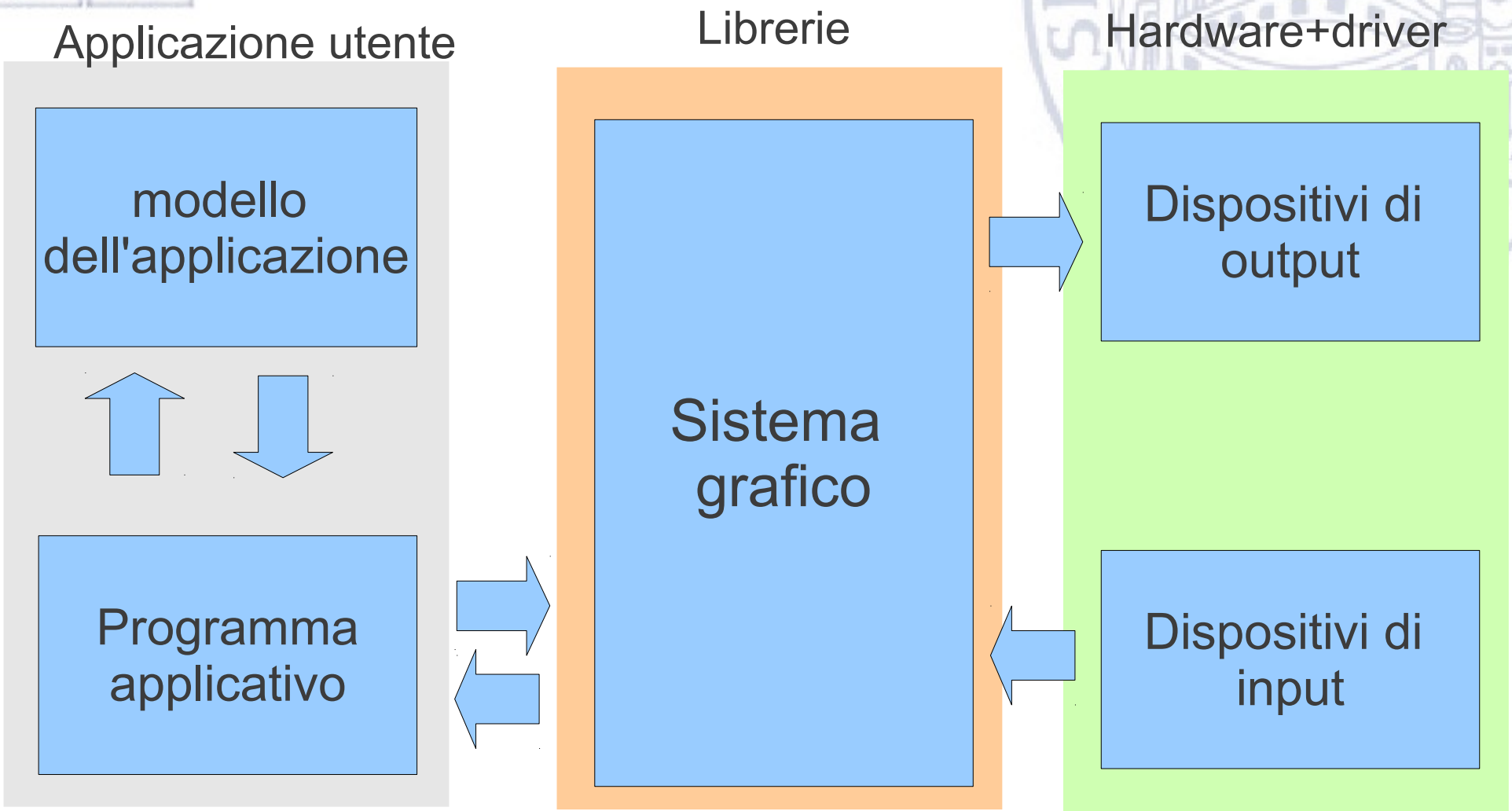
- Esempio di strumentazione per acquisizione della radianza (light field) e immagine di sintesi ottenuta dai dati acquisiti

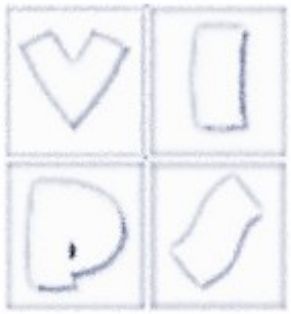
[Immagini da P. Debevec, USC, USA]

Luce, materiali, punti di vista

- Dati gli oggetti della scena vedremo che lo scopo della pipeline da implementare è "simulare" la fisica della formazione delle immagini
- Le scene dovranno pertanto contenere le sorgenti luminose, le telecamere virtuali e le proprietà di interazione tra luce e materiali
- Tutto questo si inserisce nelle strutture dati del programma cercando di ottimizzare le operazioni che si dovranno svolgere
  - Ad esempio trovare gli oggetti che contribuiscono a formare l'immagine e come questi siano colpiti dai raggi luminosi

In pratica


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



Quindi

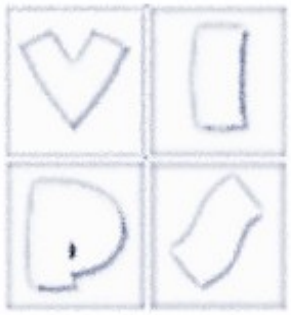
- L'applicazione grafica, per il momento una “black box” prenderà la descrizione dello stato della scena, modificabile dall'input nelle applicazioni interattive e genererà le immagini, demandando quanto possibile alle librerie grafiche e all'hardware
- In teoria potremmo usare molte metodologie per la generazione delle immagini, implementanto 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

- 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.

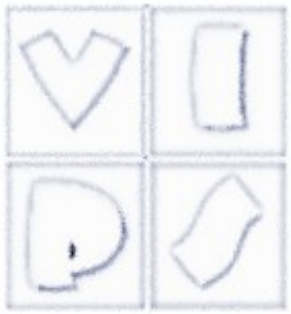
Output

- 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

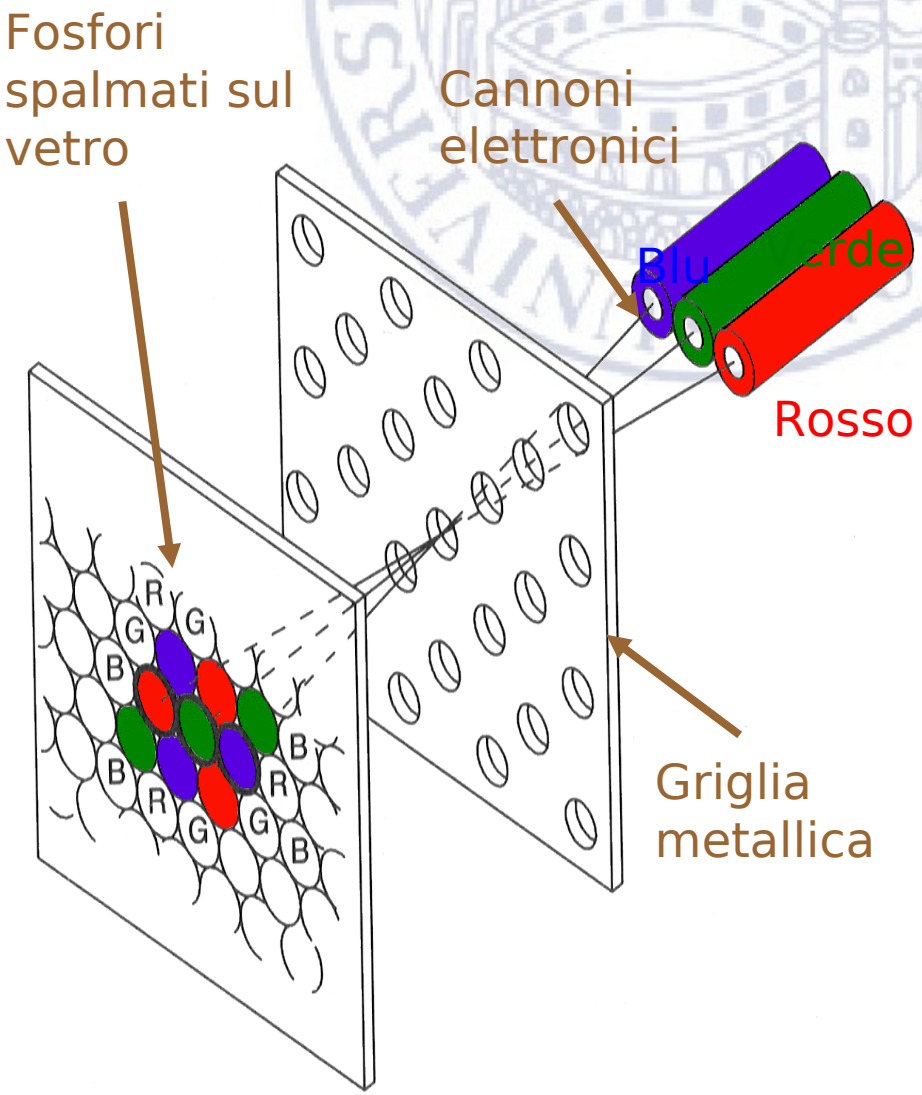
- I display attuali: tecnologia *raster*
- Spazio di output discreto bidimensionale
- Il termine risoluzione video indica il numero di pixel (*picture element*) dello schermo
- Importante oltre alla risoluzione la dimensione fisica del singolo pixel (il *dot pitch* o *pixel pitch*)
- Minore è la dimensione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

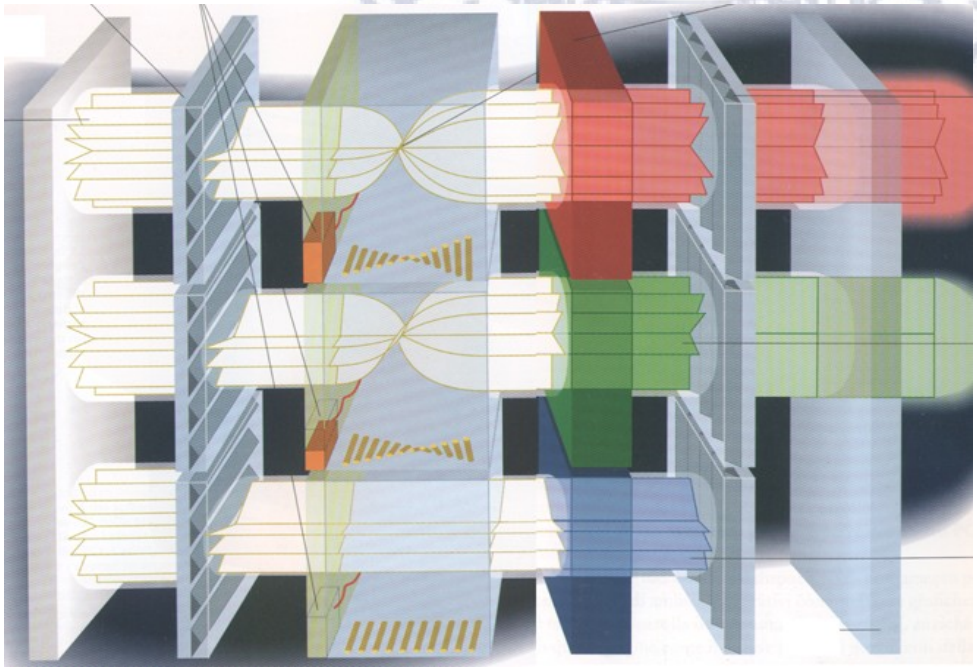
Display CRT

- 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

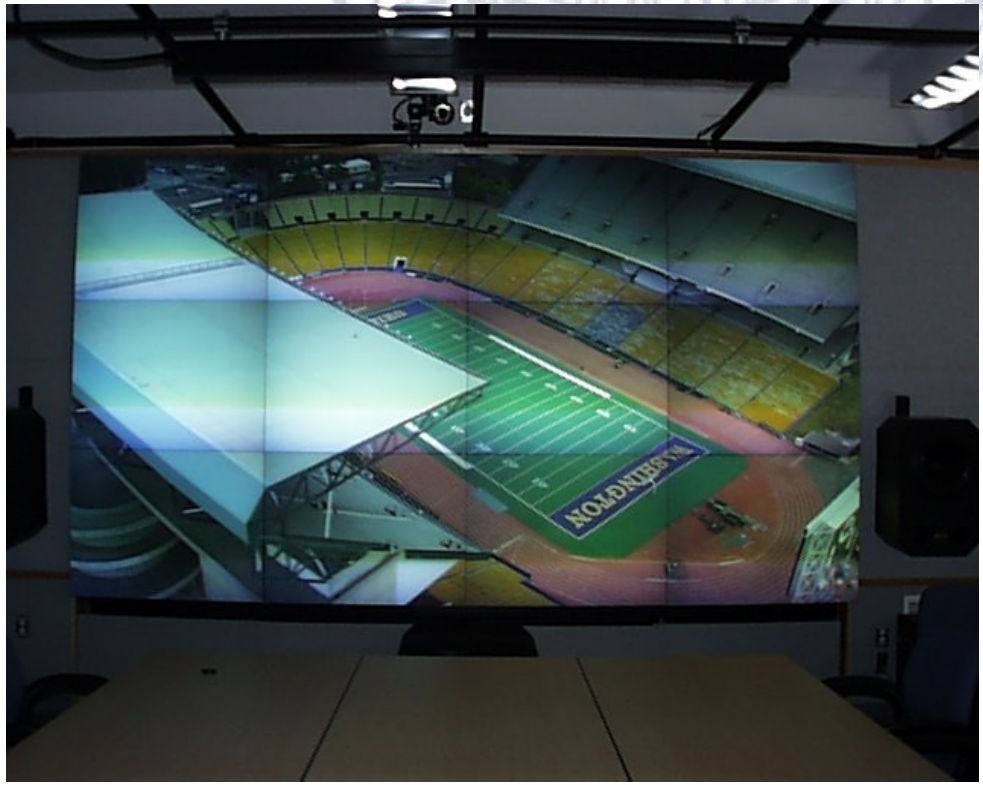


Sistemi a proiezione

- 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

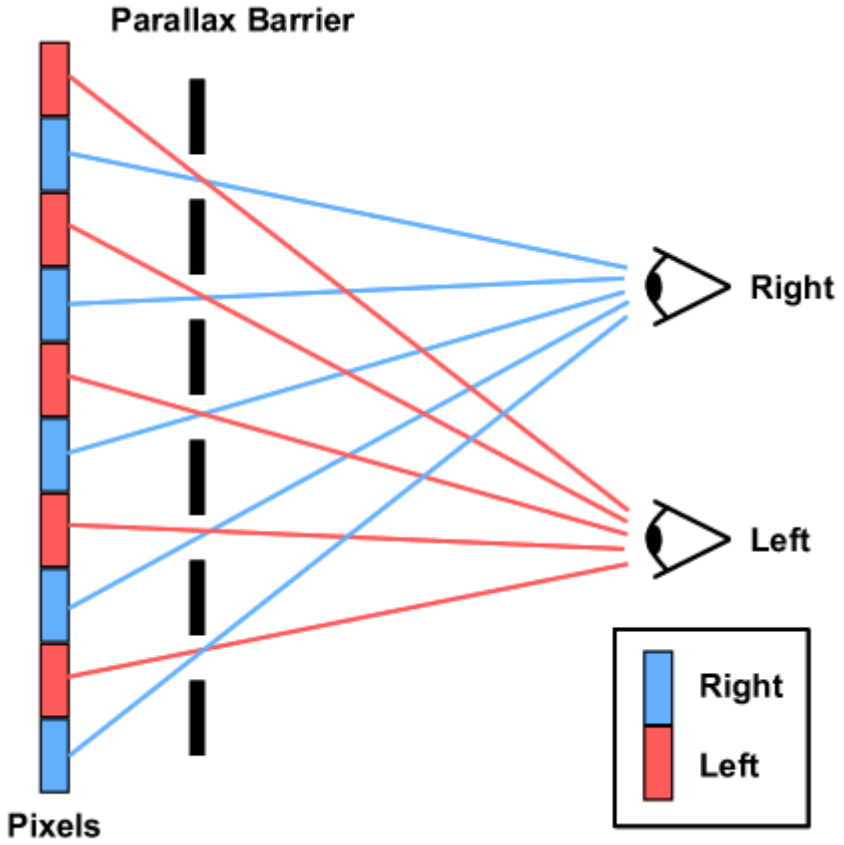
- Principio:
  - fornire due immagini leggermente diverse ai due occhi, in modo da ottenere la percezione di profondità (stereopsi)
- Stereoscopio (fine 19o secolo)
- Anaglifo
  - contiene due immagini sovrapposte, che rappresentano il punto di vista dei due occhi
  - occhiali con filtri cromatici

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

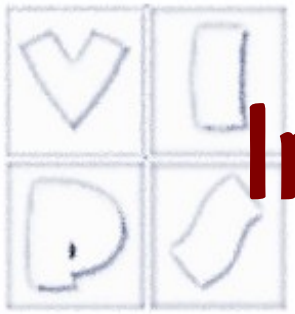
- Naturalmente per sentirsi immersi in una scena 3D occorrerebbe che cambiando il punto di vista cambi la scena opportunamente.
- Coll'autostereoscopico addirittura si perde l'effetto 3D
- Con lo stereo si ha distorsione
- Soluzioni:
  - Tracking della posizione e generazione di una nuova scena (va bene 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

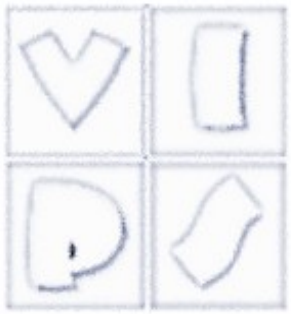
Complicazioni per i sistemi grafici

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



Input per la grafica interattiva

- Coi sistemi 3D e i paradigmi correlati hanno naturalmente acquisito importanza i dispositivi di puntamento
- Ora la situazione è cambiata
- L'input modifica con



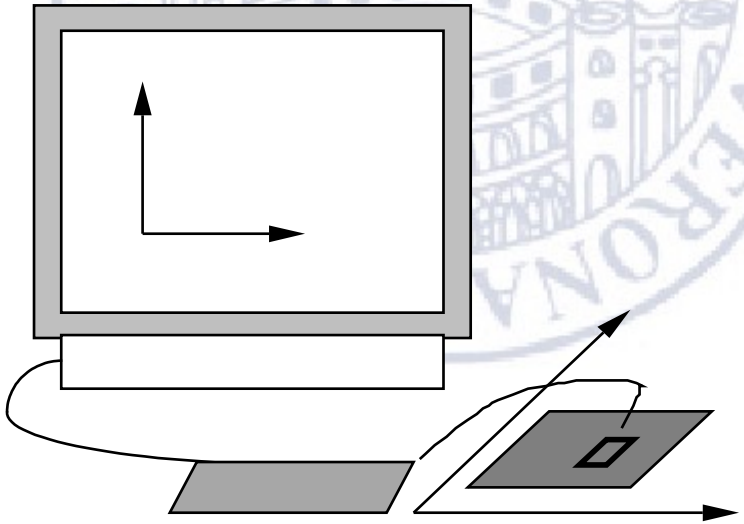
Dispositivi di puntamento

- Se l'interazione è la classica “wimp” in 2D lo strumento base con cui diamo l'input al PC è un dispositivo di “puntamento” che muove il cursore sullo schermo 2D
- Tipicamente indiretto (mouse)
- Ma anche diretto (touchscreen)
- Vi sono molti tipi di devices che si possono però utilizzare allo scopo

Il mouse

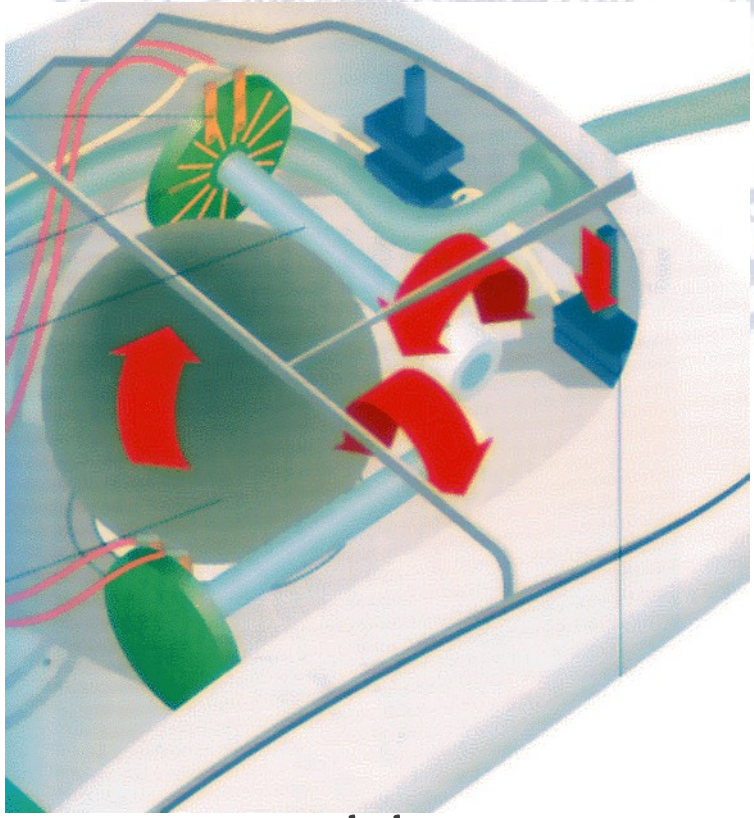
- Dispositivo di puntamento tascabile
  - molto comune
  - facile da usare
- Due caratteristiche
  - funzionamento sul piano
  - pulsante
    - (di solito da 1 a 3 pulsanti sul lato superiore, usati per eseguire una selezione, indicare un'opzione o per iniziare un disegno ecc.)

Mouse e paradigma scrivania



- Mouse localizzato su scrivania reale
  - richiede spazio fisico
  - nessuna fatica delle braccia
- Rileva movimento relativo.
- Movimento del mouse sposta cursore sullo schermo (scrivania virtuale)
- Cursore sullo schermo orientato nel piano (x, y), movimento del mouse nel piano (x, z)
- Dispositivo di manipolazione indiretta.
  - il dispositivo stesso non oscura lo schermo, è preciso e veloce.
  - problemi di coordinamento mano-occhi per utenti inesperti. Ma la metafora funziona bene

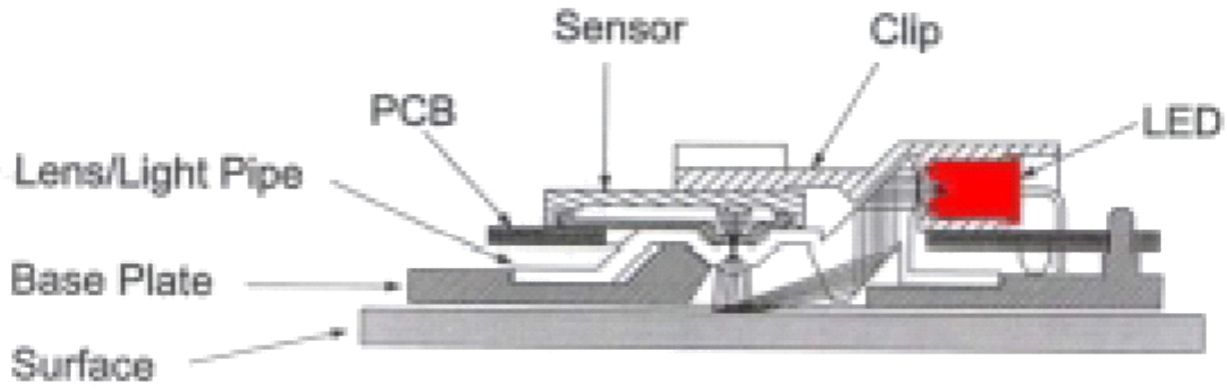
Funzionamento



- Meccanico
  - La pallina posta sulla parte sottostante del mouse gira mentre si sposta il mouse.
  - Ruota potenziometri ortogonali.
  - Può essere usato quasi su qualsiasi superficie piatta.
- Ottico
  - Diodo a emissione luminosa sulla parte sottostante del mouse.
  - Può usare piattaforma speciale a griglia o il piano della scrivania.
  - Meno sensibile alla polvere.
  - Usa le fluttuazioni nell'intensità della luce riflessa per calcolare il movimento relativo nel piano (x, z).

Mouse ottico

- Le ultime versioni non utilizzano il mousepad grigliato
  - uso di diodo LED, lente, telecamera, e un DSP per determinare direzione e distanza di moto
  - le immagini acquisite sono sovrapposte per il 50% determinando una precisione elevata (400 punti per pollice)

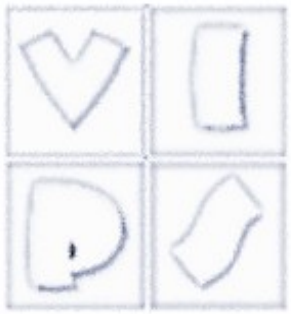


Il joystick

- Dispositivo di input principale per videogames, simulatori di volo (navigazione virtuale)
- Il joystick decodifica i movimenti impressi dalla mano ad una barra verticale tramite switch posti alla base del dispositivo
- La chiusura o l'apertura degli switch è tradotta in spostamenti

Keyboard nipple

- Minuscolo joystick isometrico
- Posizionato all'interno delle tastiera
- Buona sensibilità ma interferisce con la tastiera



Touch-screen

- Dispositivo di puntamento diretto
- Rileva la presenza del dito o dello stilo sullo schermo
  - Funziona con interruzione di raggi luminosi, in maniera capacitiva o con riflessione di ultrasuoni
- Vantaggi:
  - Veloce, non richiede puntatori specifici
  - Ottimo per selezione da un menù
  - Funziona bene in ambienti non protetti, sicuro da danneggiamento
- Svantaggi :
  - Le dita possono sporcare lo schermo
  - Impreciso (le dita sono un device di puntamento grezzo!)
  - Tenere sollevato il braccio affatica
  - Occlusione!

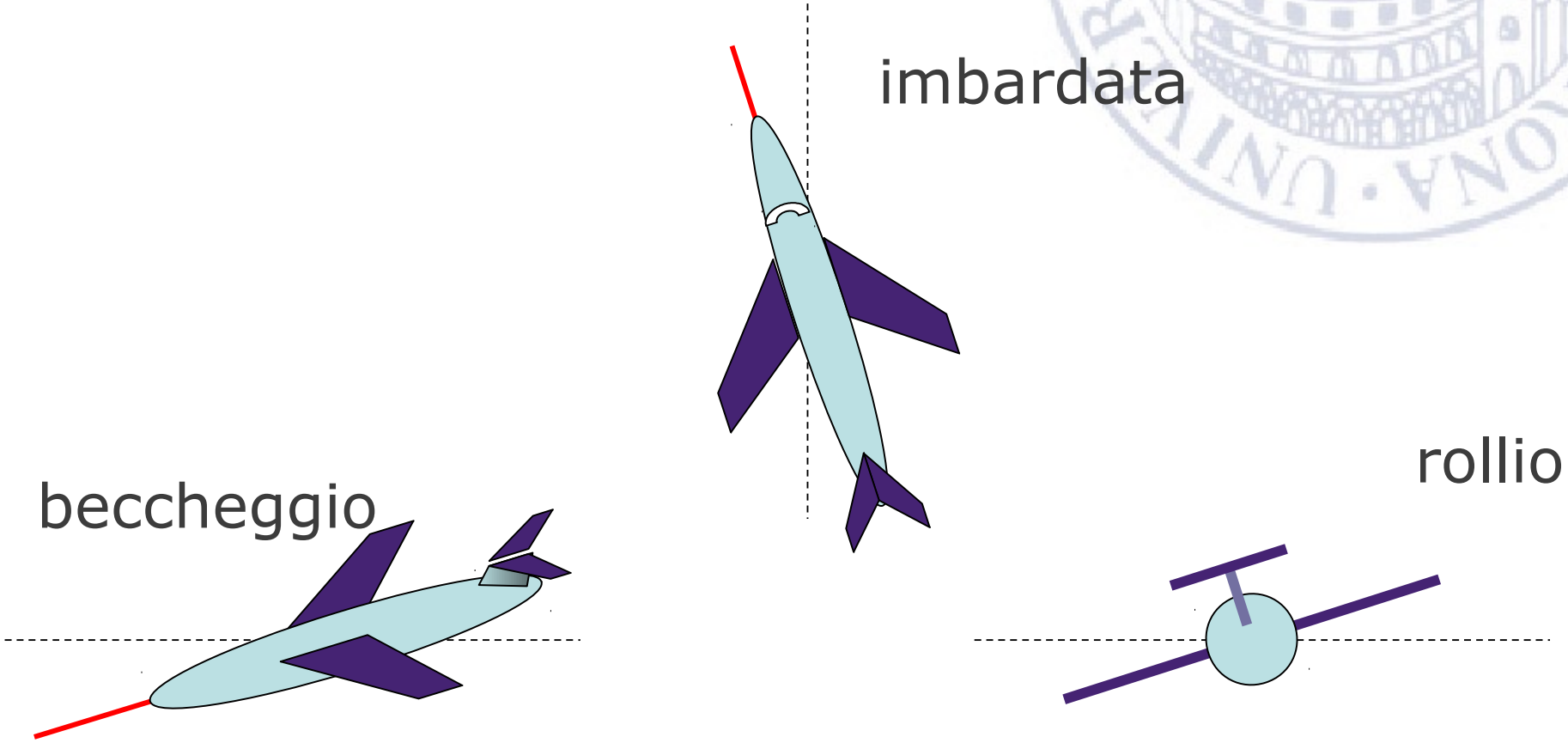
Device per posizionamento 3D

- mouse 3D
  - Strumento con 6 gradi di libertà (DOF): x, y, z + rollio, beccheggio, imbardata, non diretti
- Simulazione, es. cabina di pilotaggio e controlli virtuali
  - volanti, manopole e rotelle, come nella realtà!
- 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

Air mouse, ecc

- Esistono altri devices per catturare il movimento (es. giroscopi, vedi dopo)
- Utilizzati anche per costruire mouse, anche con controllo di posizione 3D

Orientazione in 3D

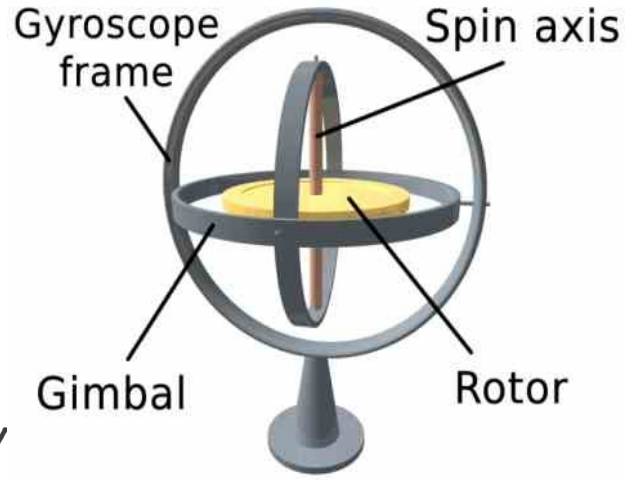


Accelerometro

- 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

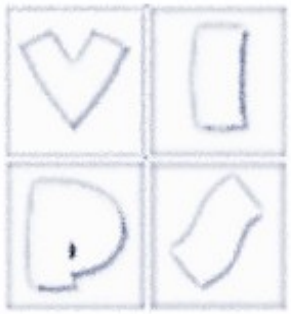


Nintendo WII

- 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

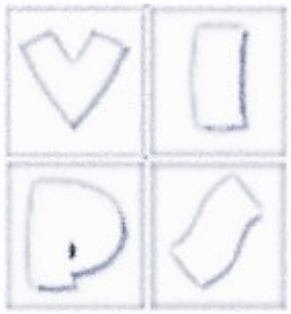
Microsoft Kinect

- 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)
- VIPS Interaction/3D scanning lab: abbiamo varie kinect e PC dedicato per stage/tesi e lavori relativi a ricostruzioni modelli 3D ed interazione



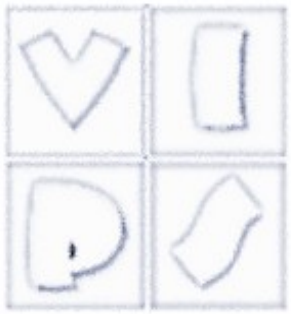
Gestione dell'input

- 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)



Modello OpenGL

- Gestione dell'input demandata in larga parte al sistema operativo, tramite window manager
- Esempi
  - apertura e chiusura di finestre
  - gestione dei dispositivi di input (mouse, tastiera ecc.)
  - dispositivi di I/O per il canale audio
- Librerie specifiche (GLUT) che interfacciano le API con l'ambiente a finestre e di gestione dell'input
- Ambiente trasparente al programmatore
- Altre librerie (es. QT) gestiscono le GUI vere e proprie (pulsanti, menù ecc.)

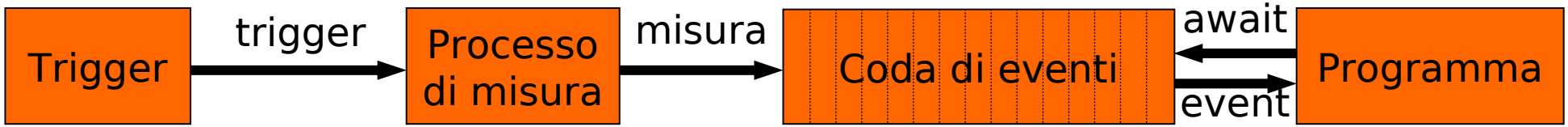


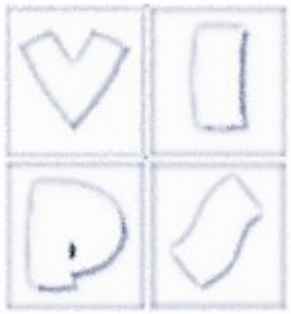
Il modello ad eventi

- 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

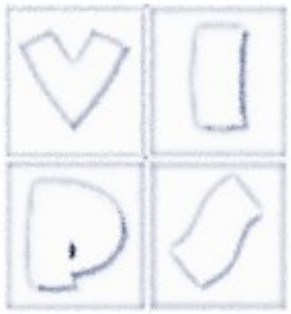
- Il programma utente può esaminare l'evento in testa alla coda se esiste, oppure può attendere che un evento si verifichi
- Il programma applicativo consuma gli eventi al momento in cui ha disponibile il time-slot per processarli





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

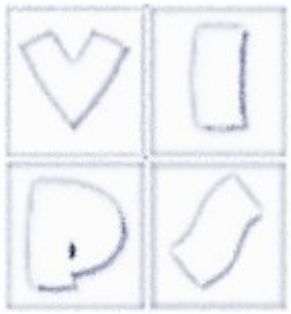


Callback

- 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

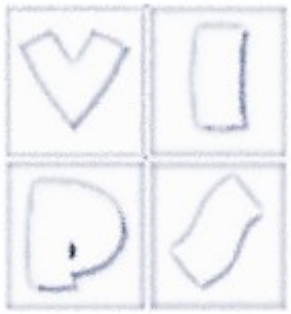
Interazione 2D-3D

- Interazione standard, es. per esaminare un oggetto consiste nel manipolarlo (traslarlo, ruotarlo)
- Se non si ha un dispositivo che cattura i movimenti in 3D, simuliamo la cosa con il mouse 2D, con modalità ormai standardizzate
  - Trackball virtuale
  - Panning
  - Zooming



Trackball virtuale

- Meccanismo di interazione comune in grafica tridimensionale: cercare di riprodurre la sensazione di esaminare un piccolo oggetto tenendolo in mano
- È molto semplice ruotarlo attorno ad un asse qualsiasi, avvicinarlo e allontanarlo da se
- Due possibilità:
  - usare un data-glove (altro dispositivo di input 3D)
  - simulare il meccanismo con un mouse

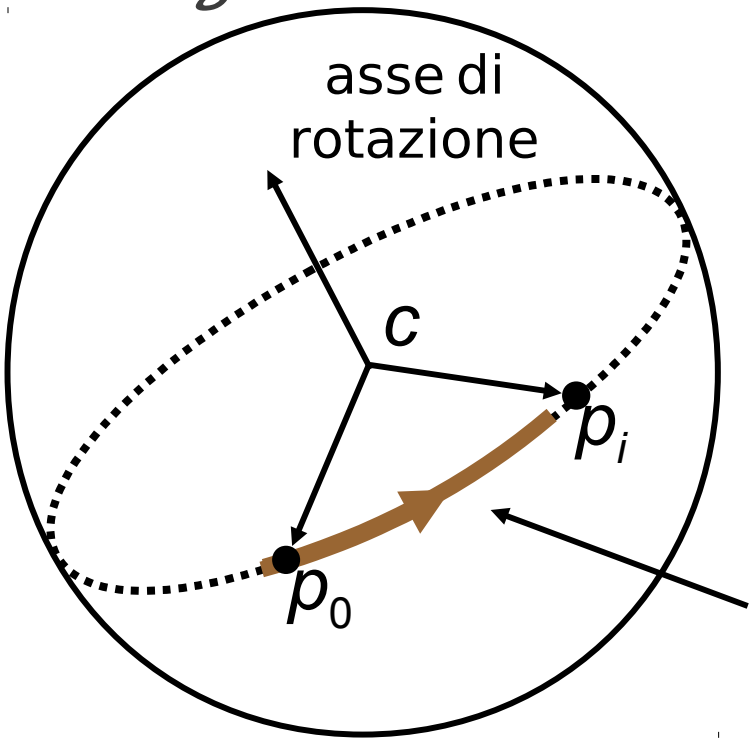


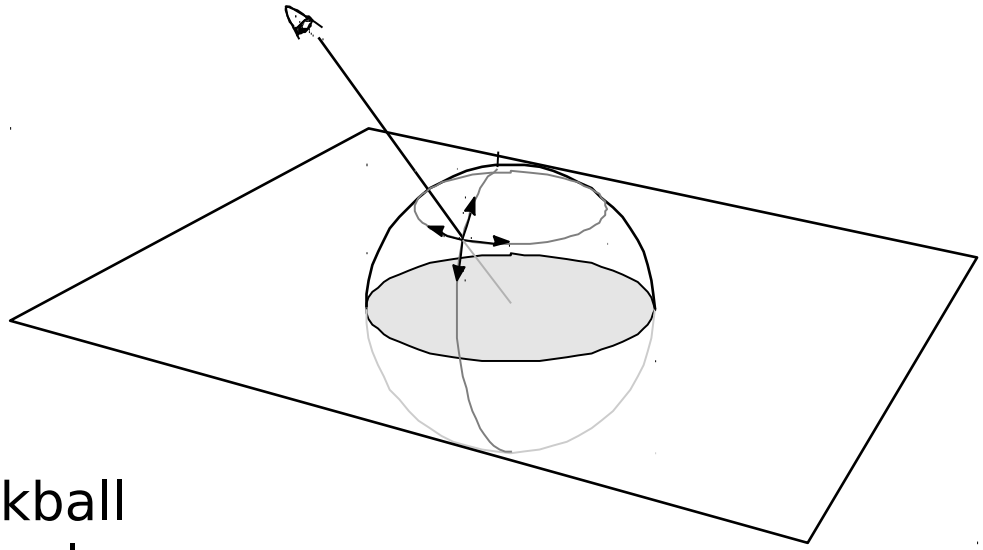
Trackball virtuale

- Pensiamo di avere una semisfera appoggiata sullo schermo la cui proiezione sullo schermo è un cerchio
- Ogni volta che posizioniamo il mouse all'interno del cerchio calcoliamo la proiezione normale al cerchio sulla semisfera (è unica!)
- Considerando la sfera in spazio di camera con il centro c sull'asse z e piazzato circa a metà tra i piani di clipping, si può calcolare l'asse di rotazione durante un'azione di trascinamento (*dragging*)

Trackball virtuale

- Tenendo traccia della posizione iniziale p0 del mouse, si può calcolare l'asse di rotazione, per ogni posizione pi come (p0−c)x(pi−c)
- L'angolo è l'arco-seno del modulo

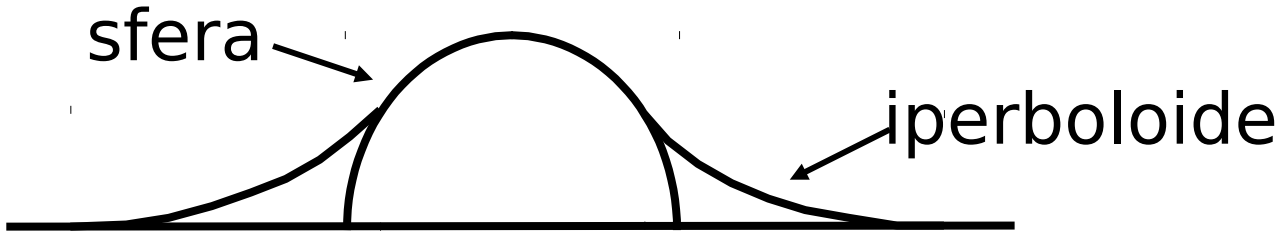


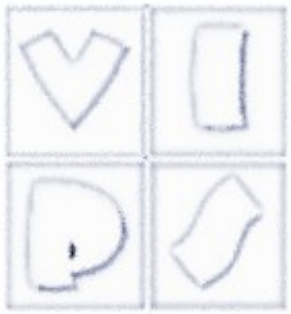


trackball
virtuale

Trackball virtuale

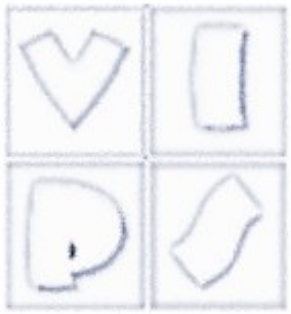
- Problema: estendere l'interazione al di fuori del cerchio per non bloccarla quando l'utente trascina il mouse fuori dal cerchio
- Soluzione: usare al posto di una semisfera l'unione tra una semisfera ed un iperboloide di rotazione (superficie che si estende all'infinito) che permette di trovare sempre proiezioni





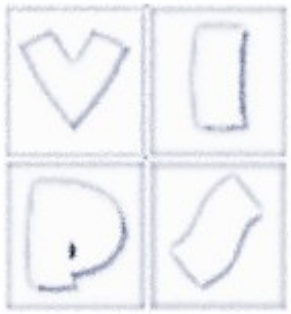
Trackball vincolata

- Estensione del meccanismo di interazione della trackball virtuale
- i movimenti del mouse in x e in y sono tradotti in movimenti dell'osservatore in coordinate polari: non si va mai oltre lo zenith o sotto il nadir
- Esempio: nell'esplorazione visiva di una statua o un edificio non si può capovolgere la scena

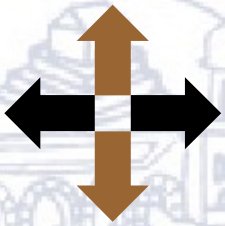


Panning e zooming

- Per rendere il meccanismo di trackball virtuale completo dobbiamo aggiungere la possibilità di fare panning e zooming
- Panning: spostare il centro della sfera (il punto attorno al quale ruota la scena)
- Zooming: variare la distanza dal modello all'osservatore

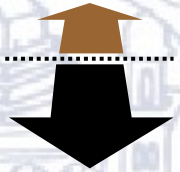


Panning



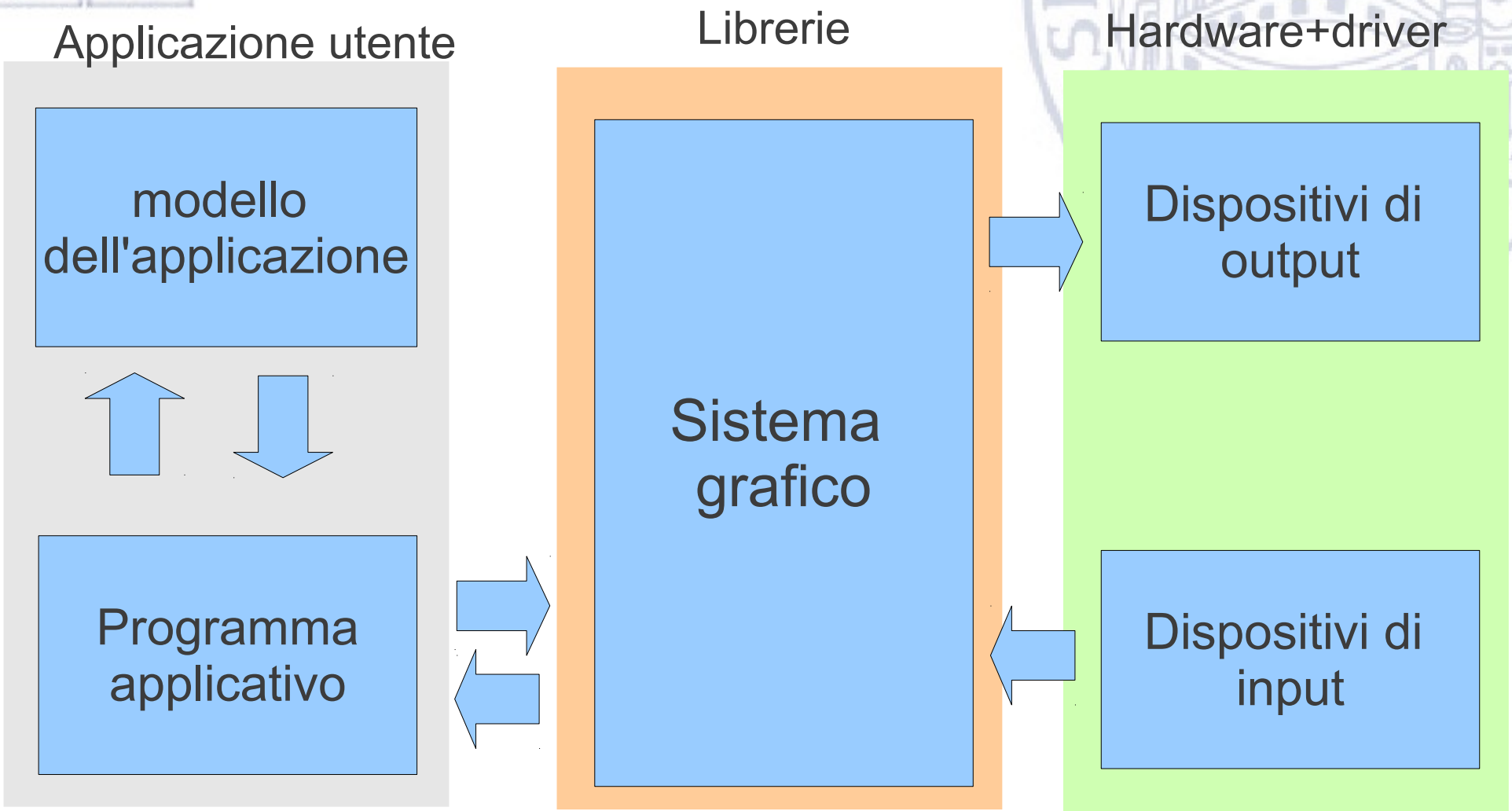
- Calcolare la distanza in coordinate schermo tra due posizioni successive
- Tradurla in uno spostamento in coordinate del modello sul piano ortogonale alla direzione da cui l'osservatore vede la scena
- Dov'è il piano? Non lo possiamo sapere (abbiamo perso la z in proiezione)
- Possiamo posizionarlo sul punto della superficie dell'oggetto su cui l'utente ha iniziato l'operazione di dragging

Zooming



- Variare esplicitamente la distanza dell'osservatore: pericoloso (varia il comportamento al variare dei parametri di vista)
- Molto meglio associare i movimenti del mouse a una scalatura della scena in spazio di camera
- Associazione non lineare poiché noi percepiamo come ingrandimento a velocità costante un oggetto che raddoppia la sua dimensione ogni k secondi
- Questo non è un fenomeno lineare nel tempo e nello spazio
- Usare una funzione esponenziale
- Ormai comune legare l'azione di zoom alla rotella del mouse

Torniamo all'applicazione



Il sistema grafico

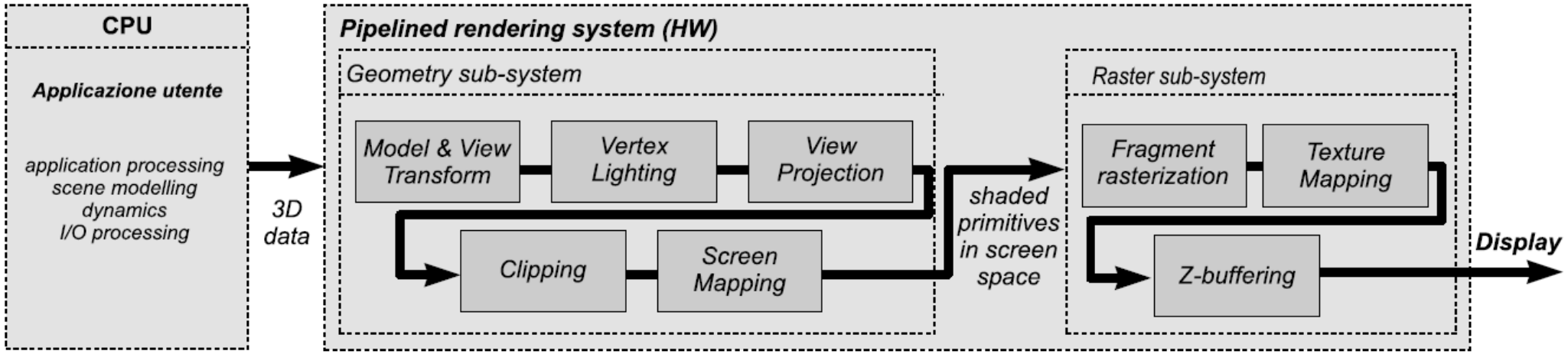
- In realtà si lavora su sistemi che hanno già un sistema grafico complesso, implementato su hardware
- Schede grafiche che implementano una procedura di generazione delle immagini a partire da primitive e descrizioni di scena standard, es. OpenGL
- Pipeline di rendering detta Rasterization: è una semplificazione del processo di formazione immagine fortemente parallelizzata
  - In realtà la rasterizzazione sarebbe solo la seconda parte della pipeline
- La vedremo bene nel corso dopo aver impostato il problema generale, parlando di modellazione e rendering
- Ma vediamo un'anticipazione per capire il contesto

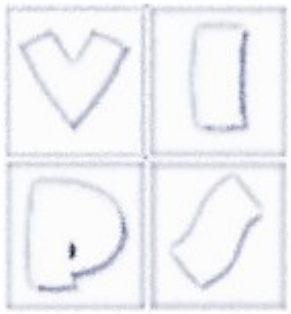
Visualizzazione della scena - API

- API (Application Programming Interface) per la grafica 3D
  - OpenGL, DirectX, ...
- Progettate per grafica 3D **interattiva**, organizzazione logica funzionale ad una efficiente implementabilità HW
- Efficienza direttamente dipendente dalla possibilità di elaborare in parallelo le diverse fasi del processo di rendering
- Soluzione vincente: suddivisione del processo in **fasi indipendenti**, organizzabili in **pipeline**
  - Maggiore parallelismo e velocità di rendering
  - Minore memoria (fasi indipendenti → memoria locale, non necessario conoscere la rappresentazione dell'intera scena ma solo della porzione trattata)

Pipeline di rendering

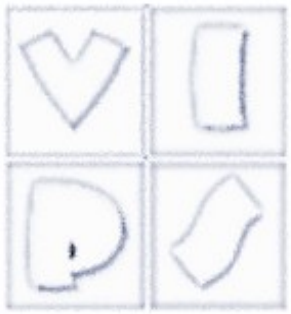
- Pipeline adottato dalle comuni API grafiche 3D e, conseguentemente, caratterizzante l'architettura dei sottosistemi grafici 3D hardware
- Le singole primitive 3D fluiscono dall'applicazione al sottosistema grafico, e sono trattate in modo indipendente dai vari stadi del sottosistema grafico
- Approccio 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)

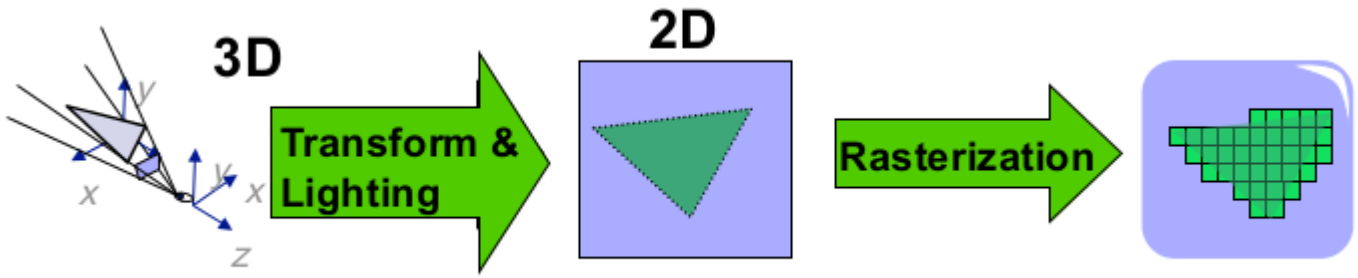


Primitive geometriche

- 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)

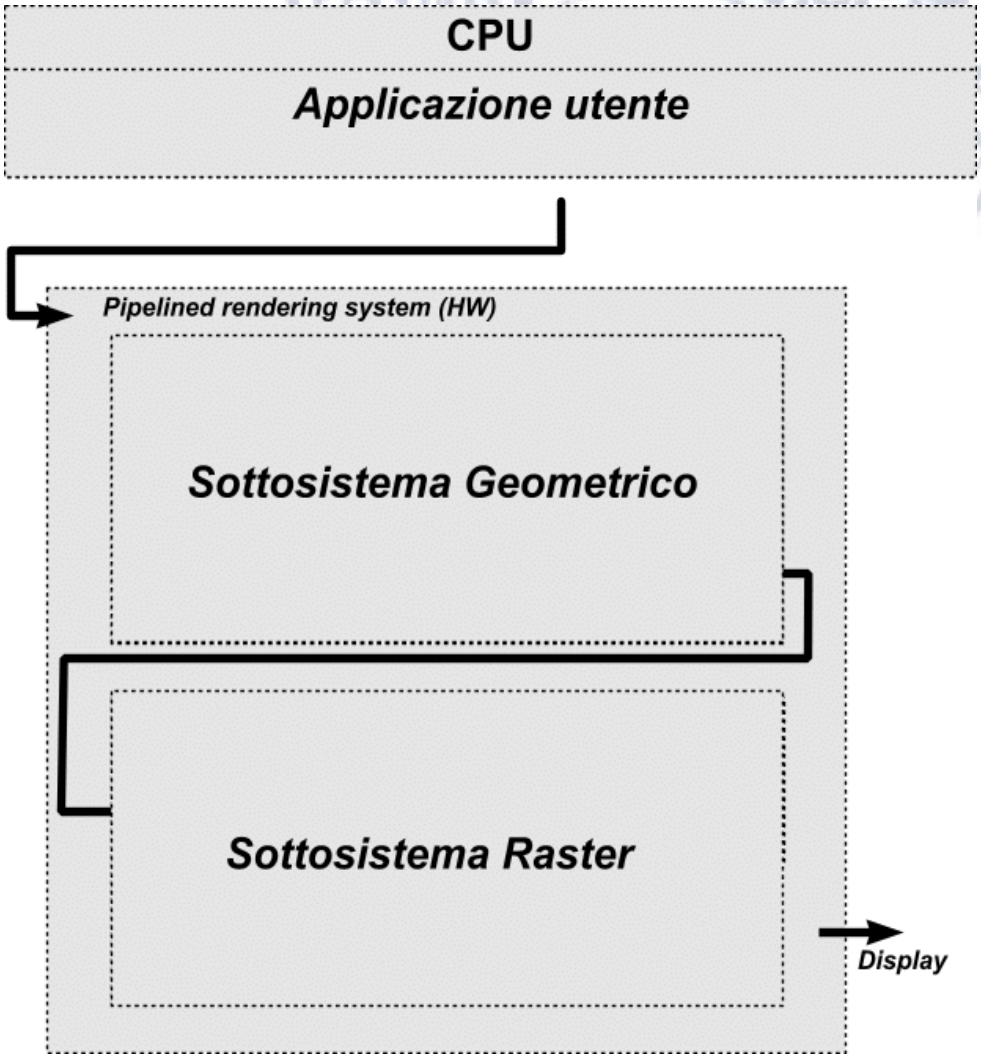
OpenGL Rendering Pipeline

- Nei prossimi lucidi ci concentreremo sul funzionamento della pipeline grafica standard
  - Utilizzata da tutto l'hardware grafico corrente
  - 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



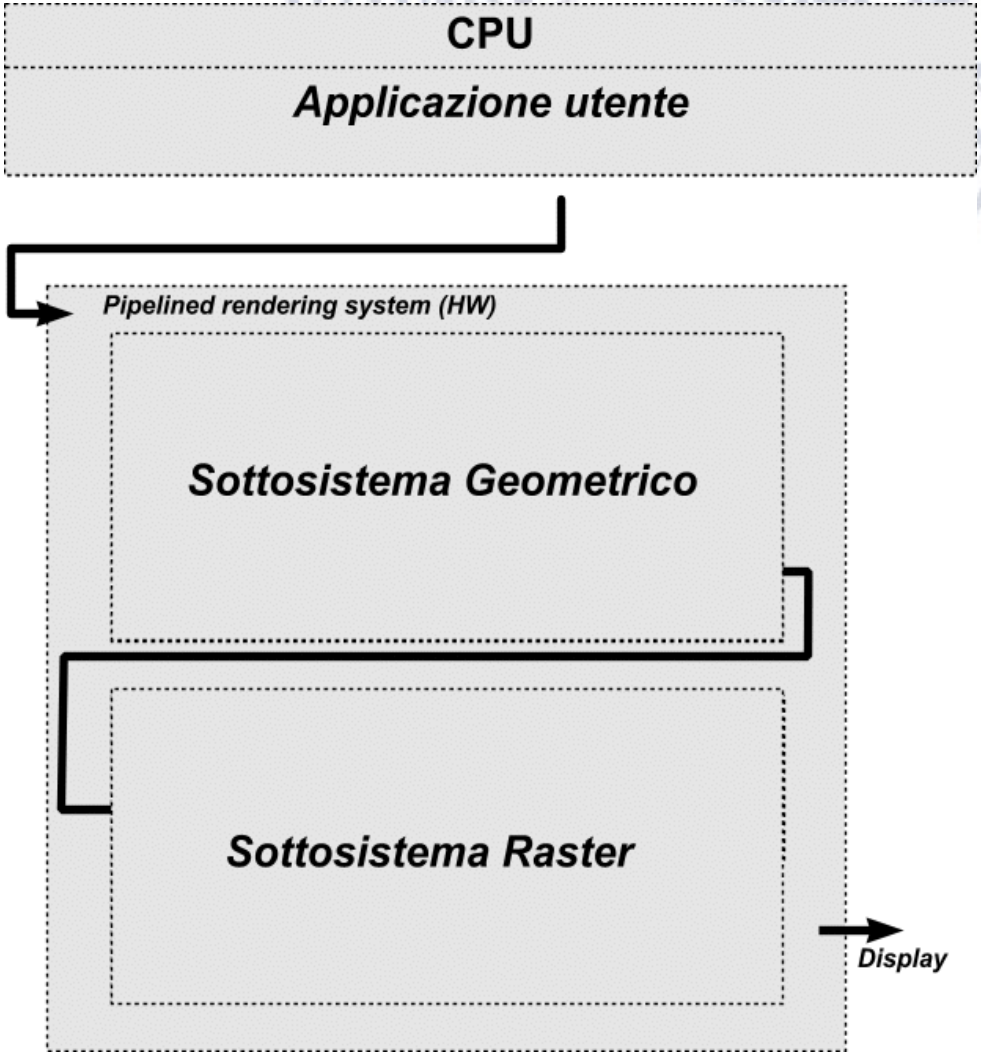
Basic Pipeline

- 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



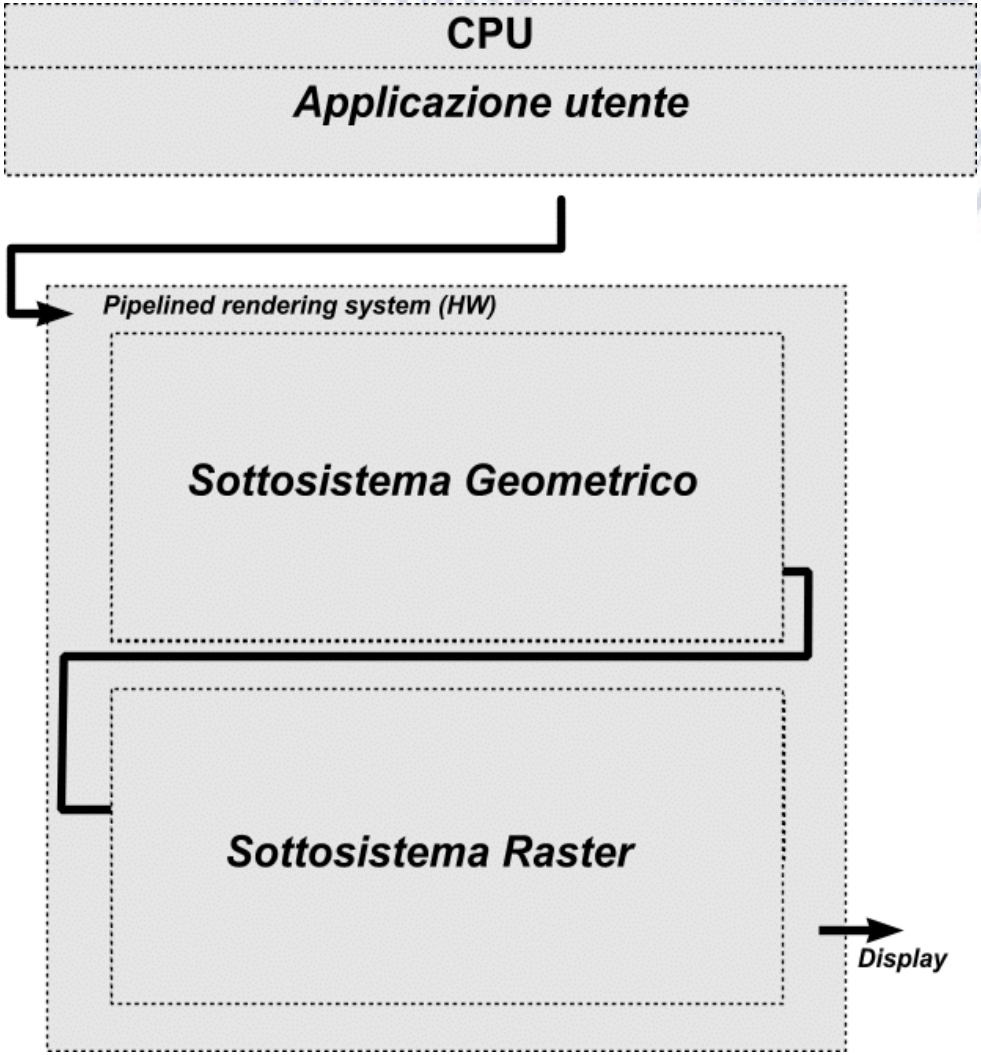
Basic Pipeline

- 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



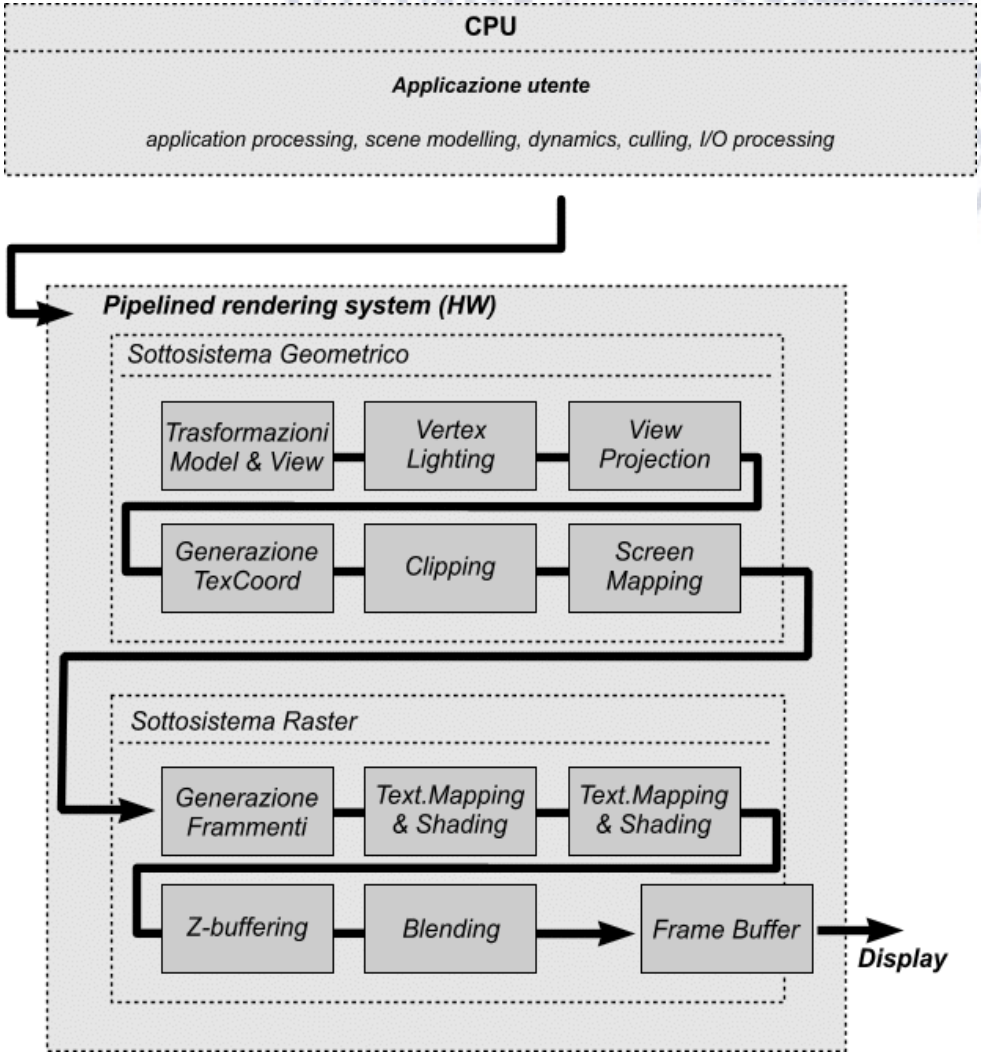
Basic Pipeline

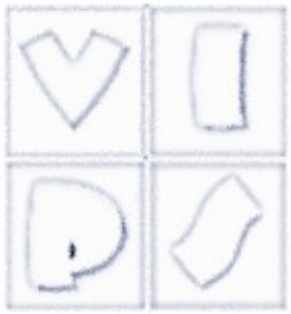
- Tre principali attori in gioco
  - L'applicazione
  - Il sottosistema geometrico
  - Il sottosistema raster
    - che per ogni primitiva di cui ormai si conosce la posizione finale accende i pixel dello schermo da essa coperti in accordo a
      - Colore
      - Texture
      - profondità



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

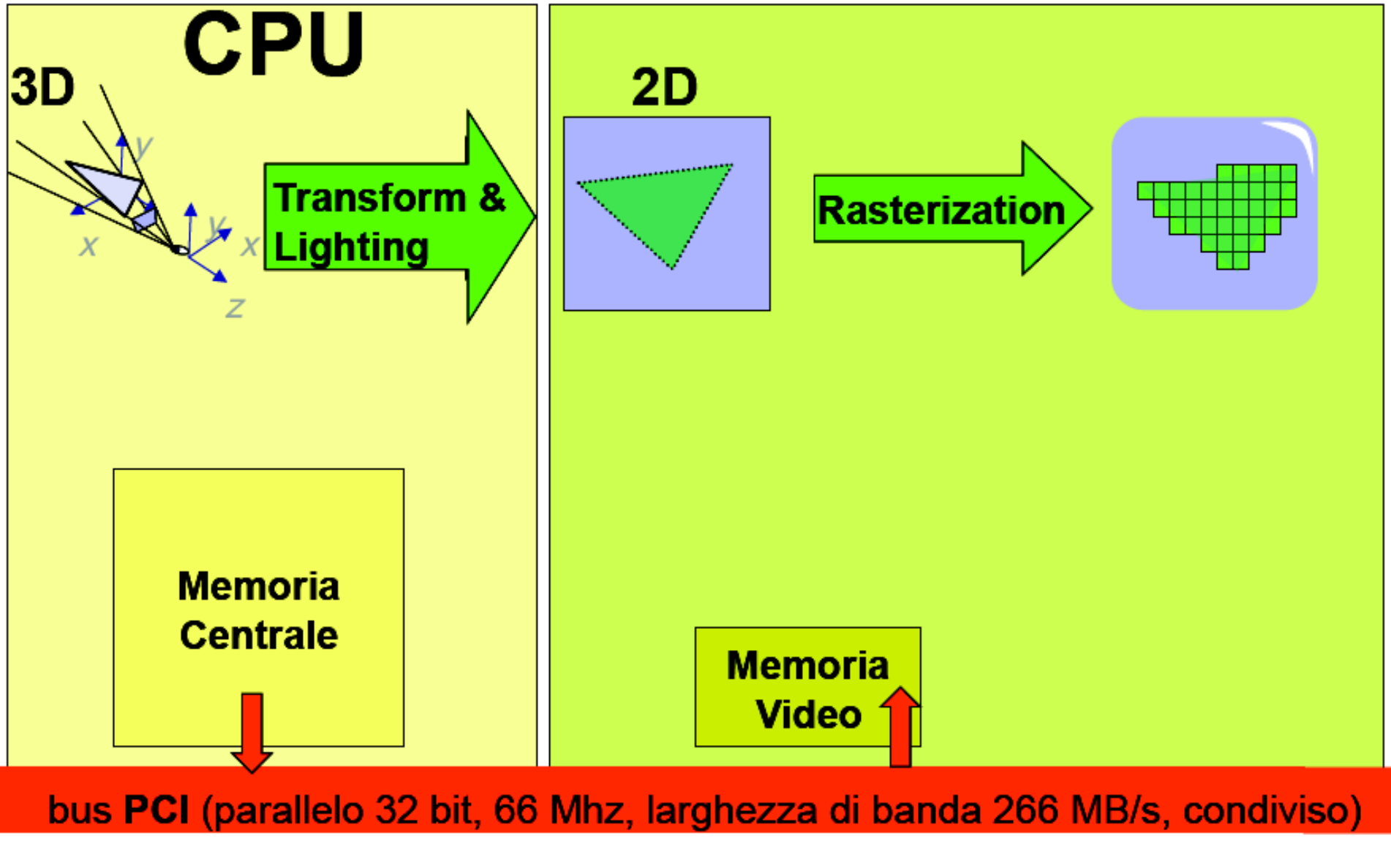




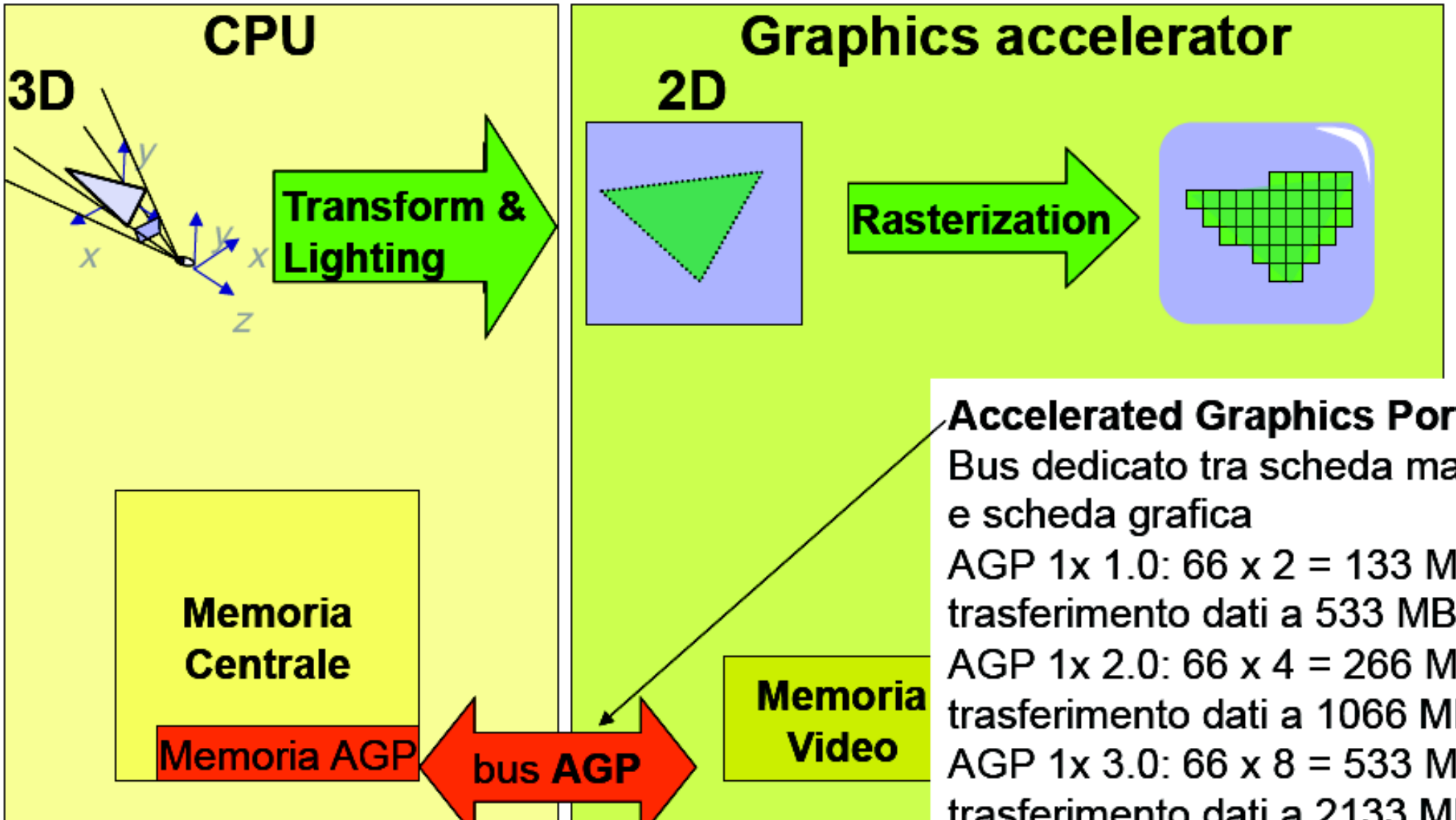
Schede grafiche

- 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)

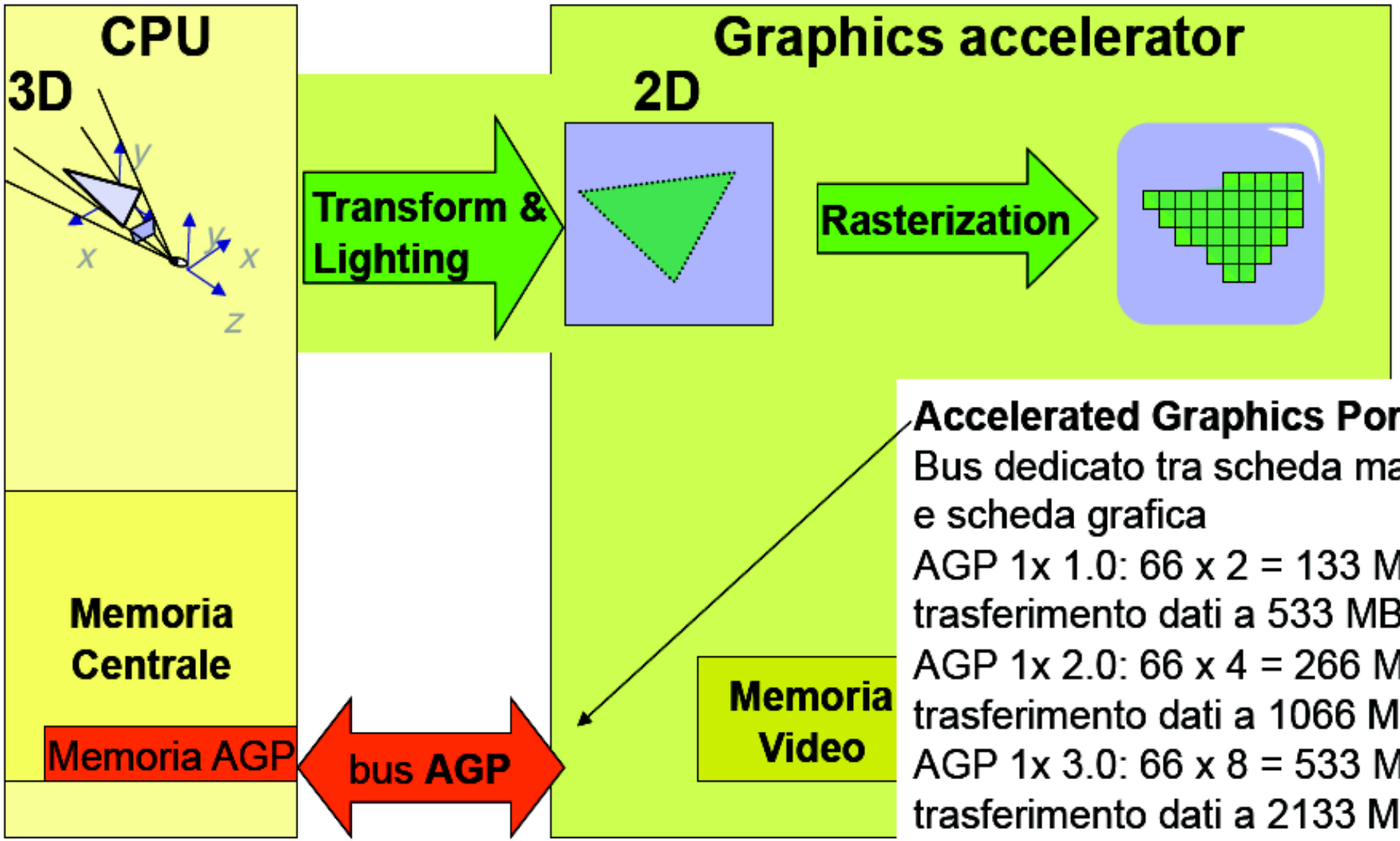
- 1995-1997: 3DFX Voodoo



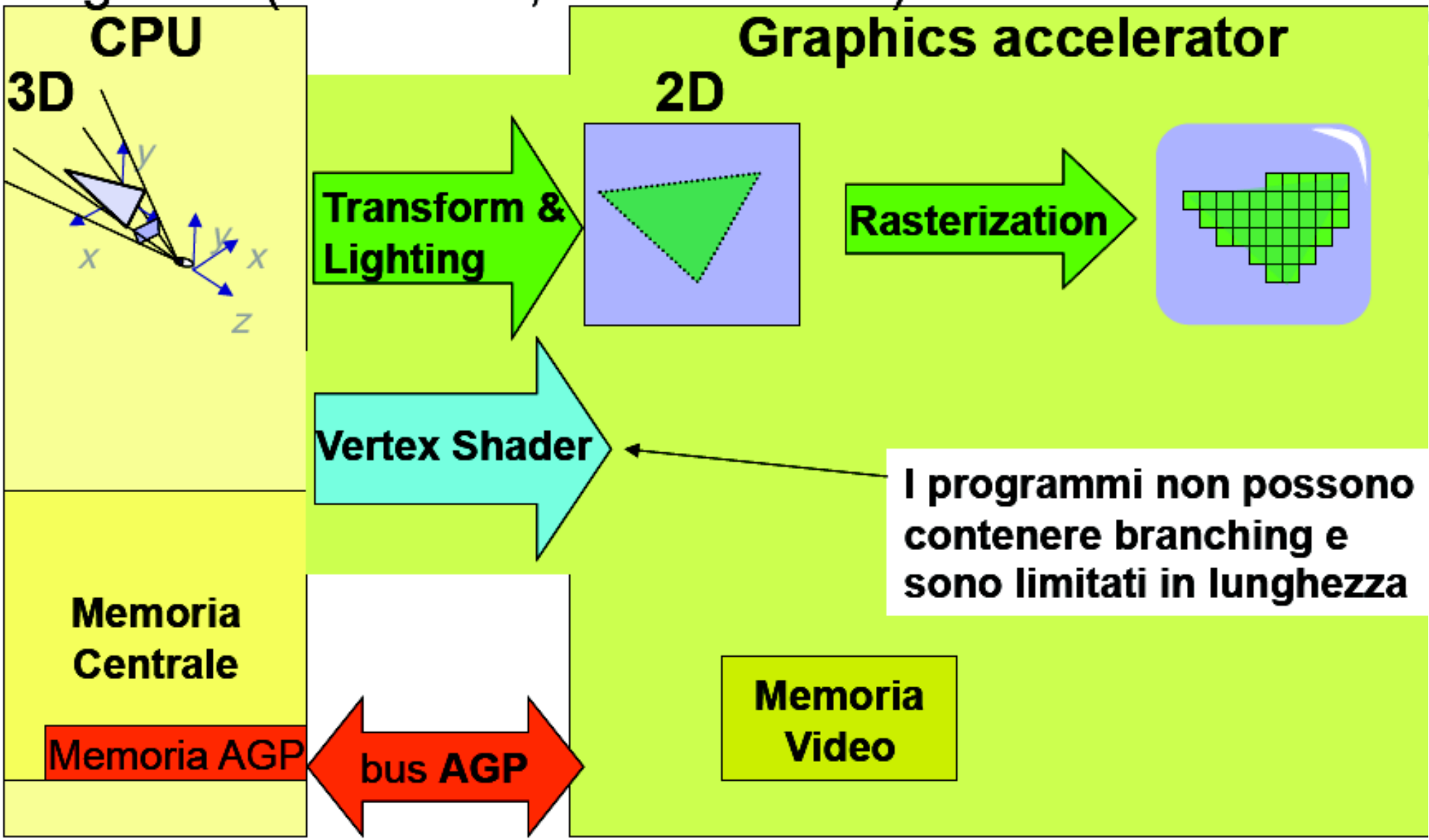
- 1998: NVidia TNT, ATI Rage



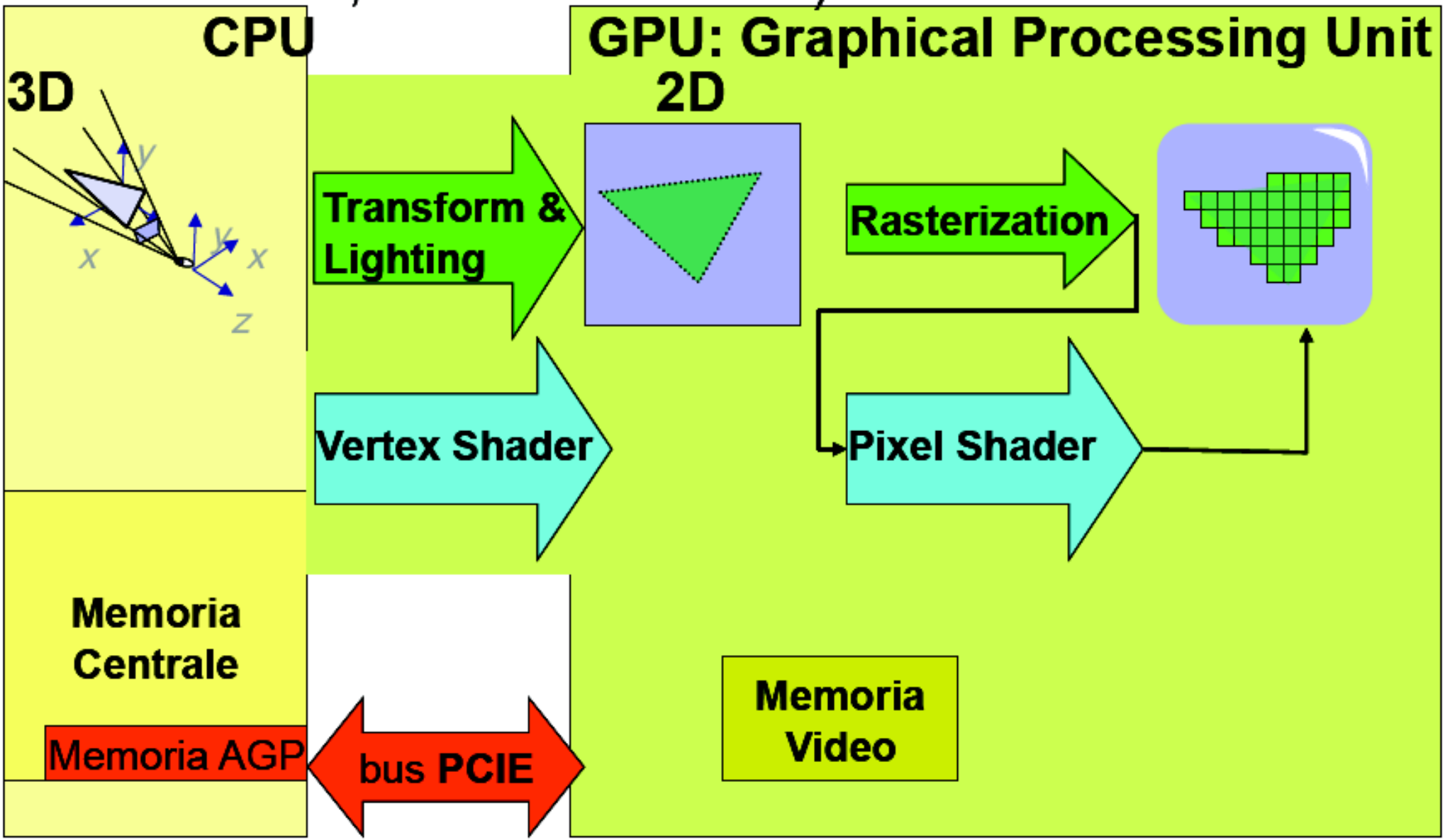
- 1998: NVidia TNT, ATI Rage



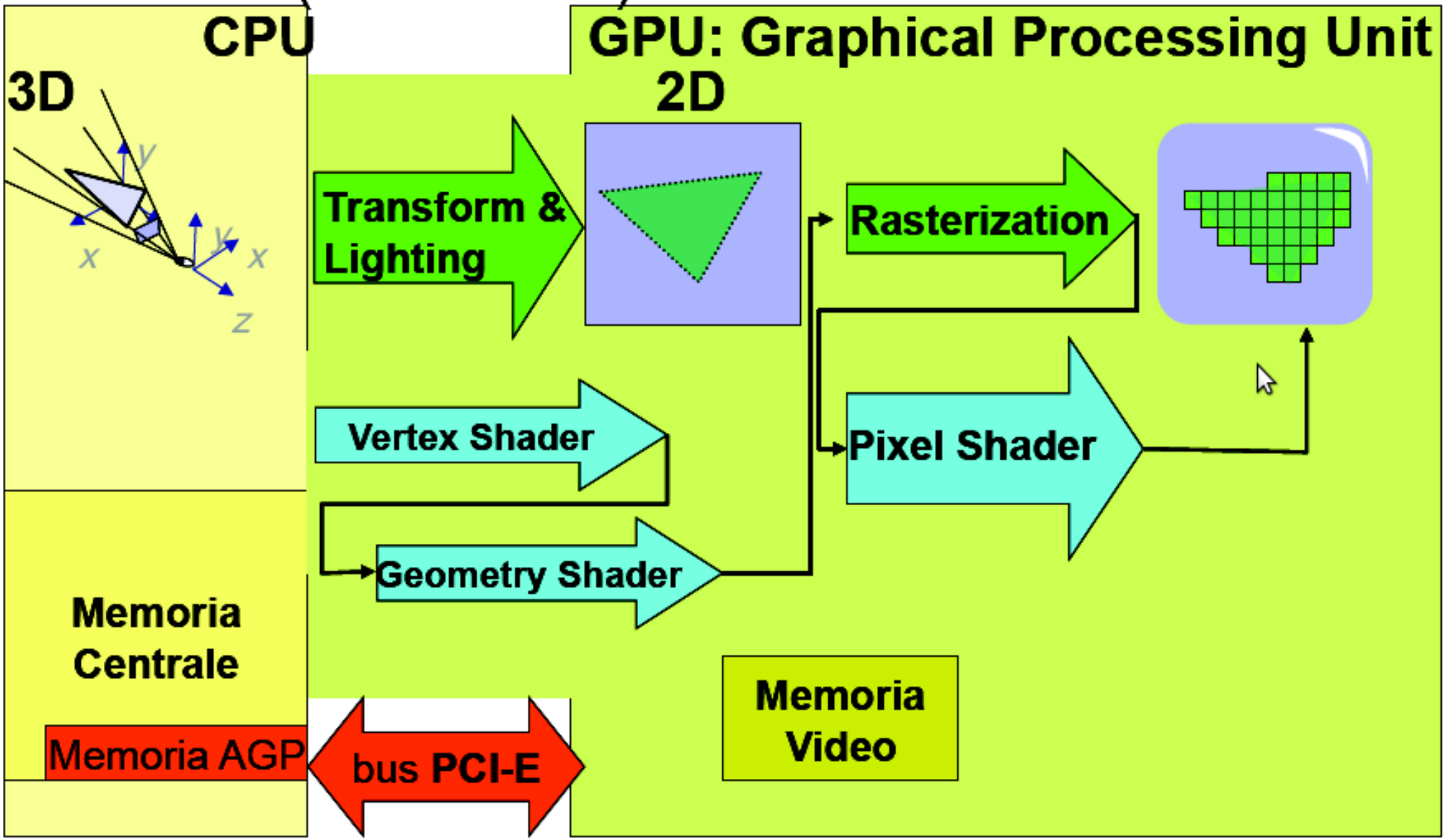
- 2001: Vertex Shader. Programmare il chip della scheda grafica (GeForce3, Radeon 8500)

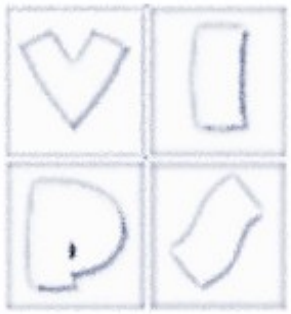


- 2004-5: PCI-Express, branching negli shader (GeForce 6800-7800, ATI Radeon 9200)



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



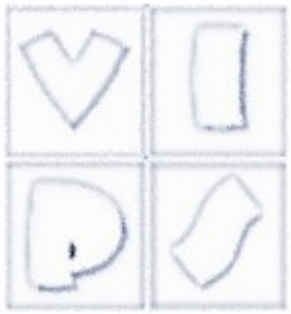


Livello applicazione

- 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

Livelli gerarchici/grafi

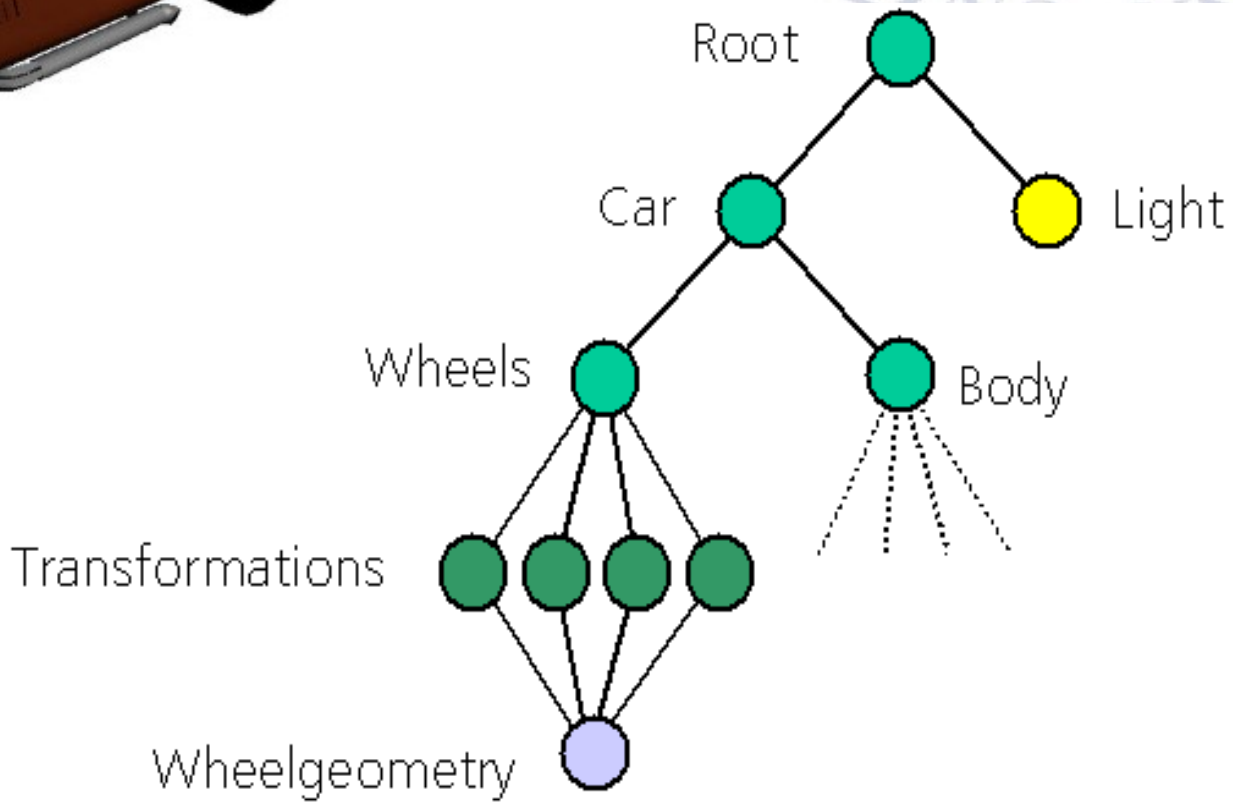
- Gli oggetti modellati possono essere inseriti in strutture gerarchiche per semplificare la gestione
- Questo è spesso implementato nelle librerie di più alto livello sopra openGL (es. OSG, java3D, ecc.).
- Distinguiamo
  - Gerarchie di oggetti: in tal caso gli oggetti vengono distribuiti su un albero 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

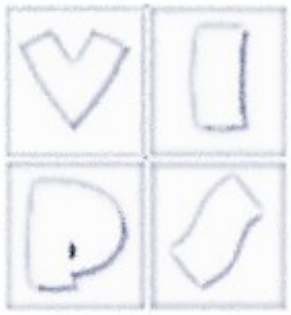


Nota

- In OpenGL non ci sono nemmeno i “modelli” degli oggetti, al più gruppi di triangoli, ecc.
- Anche il generare modelli da riutilizzare rientre nei compiti del layer applicativo superiore

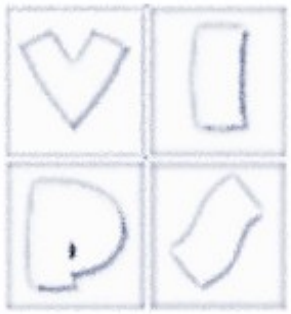
Esempio


Root
Car
Light
Wheels
Body
Transformations
Wheelgeometry



Nota

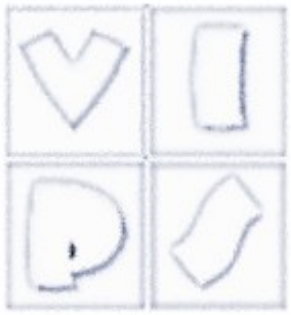
- Usando questa astrazione con oggetti e grafi possiamo fare anche la distinzione tra oggetto logico e istanza fisica
  - Se ho una scena con N automobili uguali posso istanziare N versioni trasformate dello stesso oggetto logico
  - Le istanze potranno avere variazioni nelle proprietà (es. colore, posizione), esprimibili volendo anche con nodi o attributi del grafo
  - Si risparmia memoria



Nota

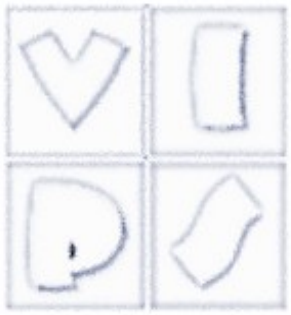
Sul livello applicazione, sugli oggetti potremo poi applicare algoritmi opportuni per ottimizzare il lavoro della pipeline grafica

- Es. selezionare le primitive da visualizzare
- Evitare di usare livello di dettaglio inutile
- Ne parleremo quando avremo chiari i dettagli della pipeline



Ma ora

- Torniamo ai fondamenti e vediamo
  - La modellazione
  - Il problema della generazione delle immagini e gli algoritmi di rendering
  - La pipeline di rasterizzazione



Riferimenti

- Scateni et al. Cap. 2,3
-