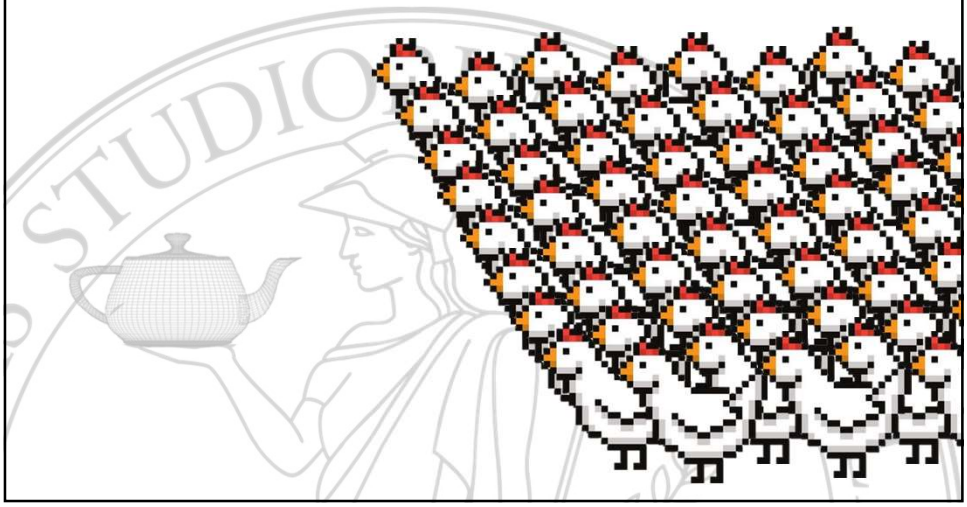
Marco Tarini - Grafica & Immagini Digitali 2022/2023
Università degli Studi di Milano

## Graphic Hardware, Drivers, API e librerie (cenni)



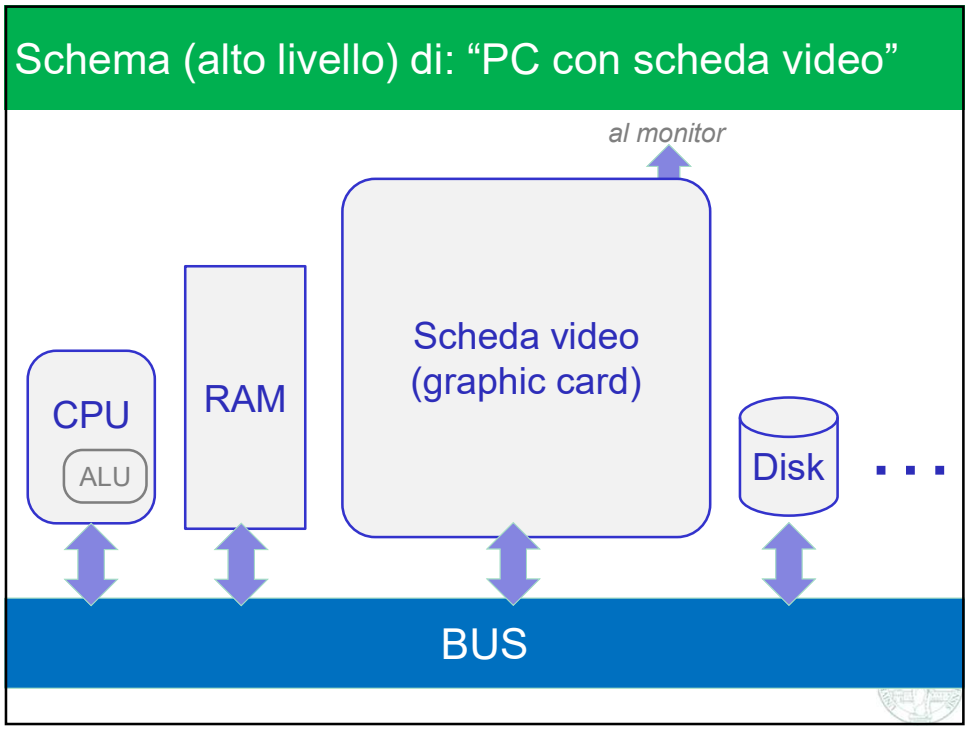
1

## Hardware specializzato per il rendering

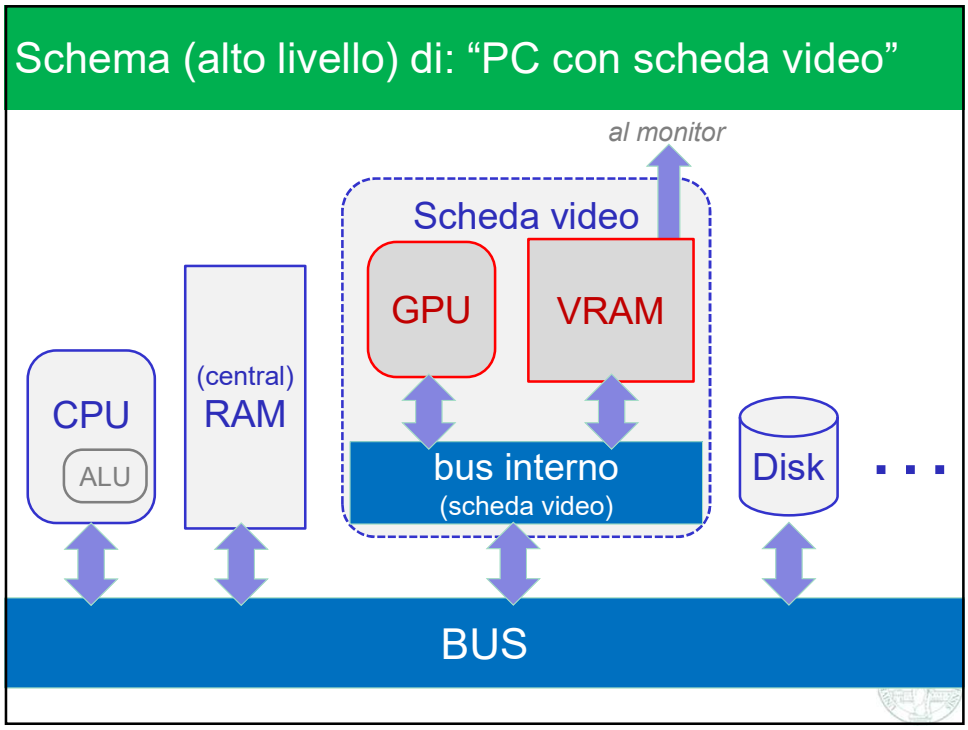
### Visione di insieme:


- ✓ "**GPU**":
  - ⇒ Graphics Processing Unit
  - ⇒ La CPU della scheda video
  - ⇒ *Instruction Set* specializzato!
- ✓ Architettura a ***pipeline***
  - ⇒ a "catena di montaggio"
- ✓ Modello di computazione **SIMD**
  - ⇒ sfrutta l'alto grado di parallelismo insito nel problema
- ✓ Possiede la **propria** memoria RAM a bordo
  - ⇒ "RAM CPU" vs "RAM GPU"
  - ⇒ grandi copie di memoria da una all'altra dispendiose

3

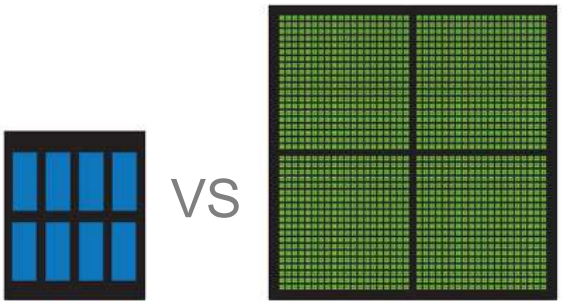

Schema (alto livello) di: “PC con scheda video”
al monitor
CPU
ALU
RAM
Scheda video
(graphic card)
Disk
BUS

4


Schema (alto livello) di: “PC con scheda video”
al monitor
Scheda video
GPU
VRAM
CPU
ALU
(central)
RAM
bus interno
(scheda video)
Disk
BUS

5

## CPU vs GPU

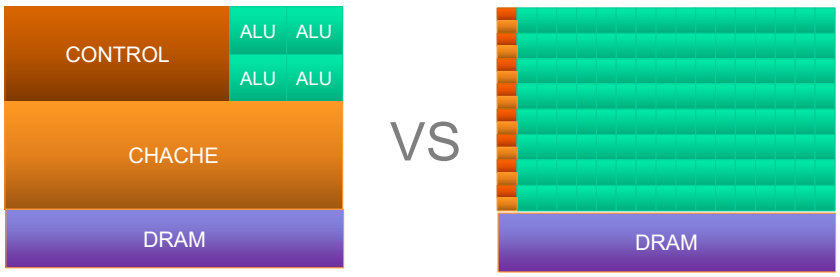



CPU
multiple cores
(per es, 6)

GPU
multiple cores
(per es, 2000)

6

## CPU vs GPU




CPU GPU

7

## CPU vs GPU

| CPU | | GPU |
|---|---|---|
| ✓ Transistors: >80% control+cache | VS | ✓ Transistors: ~80% ALU |
| → flessibilità | | → potenza (FLOPS) |

Ad esempio, attualmente (circa 2013):

Una moderna GPU può avere una potenza di calcolo misurabile in diverse decine di TERAFLOPS (cioè circa 10^13 FLOPS)

(FLOPS = FLoating-point OPeration per Second)

Una moderna CPU: «solo» centinaia di GIGAFLOPS
(cioè circa 10^11 FLOPS, due ordini di grandezza inferiore!)

Il divario si è mantenuto o allargato nelle generazioni

9

## CPU vs GPU (nota di colore)

- ✓ Intel: “GPUs are only 2.5x to 14x times faster than GPUs on many tasks” – Lee et al., INTEL, in *Debunking the 100X GPU vs. CPU myth* ACM SIGARCH 2010
- ✓ Answer by Nvidia: “It’s a rare day in the world of technology when a company you compete with stands up at an important conference and declares that your technology is *only* up to 14 times faster than theirs.”

11

## Hardware specializzato per il rendering

✓Vantaggio: **efficienza**

- instruction set specializzato
  - (computazioni più comuni sono hard-wired)
- computazioni in parallelo:
  1. fra CPU e GPU
     - » rendering demandato alla scheda grafica
     - » resto dell'applicazione libera di utilizzare la CPU e RAM base
  2. a volte: fra GPU distinte (es. più schede sullo stesso BUS)
  3. fra le fasi del pipeline (vanno tutte in parallelo)
  4. dentro *ogni fase* del pipeline (più sottoprocess. per fase)
  5. instruction level: operazioni operano su vettori di 4 operandi

✓Svantaggio: **rigidità**

- scelta quasi obbligata dell'approccio al rendering utilizzato

12

## Efficienza di un pipeline (in generale)

✓ Come tutte le architetture pipelined, la velocità della GPU grafica è determinata dallo step più lento, detto “il collo di bottiglia”, o bottleneck

⇒Le fasi successive vengono “starved”, sono spesso inattive in attesa di input

⇒Le fasi precedenti vengono “chocked”, sono spesso inattive in attesa che la fase seguente sia pronta a ricevere i propri output

13

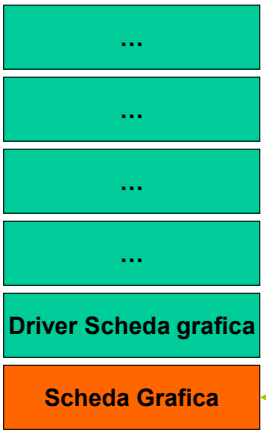
## Efficienza del pipeline di rendering rasterization based

- ✓ Dove è il collo di bottiglia?
  - ⇒ Nella fase per vertice?
    - l'applicaizone è detta *transofrm-limited*
      sinonimo: *geometry-limited*
    - "C'è troppa geometria, la trasformazione geometrica di tutti questi vertici impiega troppo tempo"
  - ⇒ Nella rasterizzatore, o nella fase per frammento
    - applicaizone è *fill-limited*
    - "L'applicazione non riesce a riempire (fill) lo screen buffer di pixel a abbastanza in fretta"
- ✓ Conseguenza: le prestazioni (frame per secondo)…
  - ⇒ …delle applicazioni geometry limited migliorano (ad esempio) riducendo il numero e/o la risoluzione delle mesh renderizzate (o delle point cloud, etc)
  - ⇒ …delle applicazioni transform limited migliorano (ad esempio) riducendo la risoluzione dello schermo
  - ⇒ (Perché non migliorano, nell'altro caso?)

14

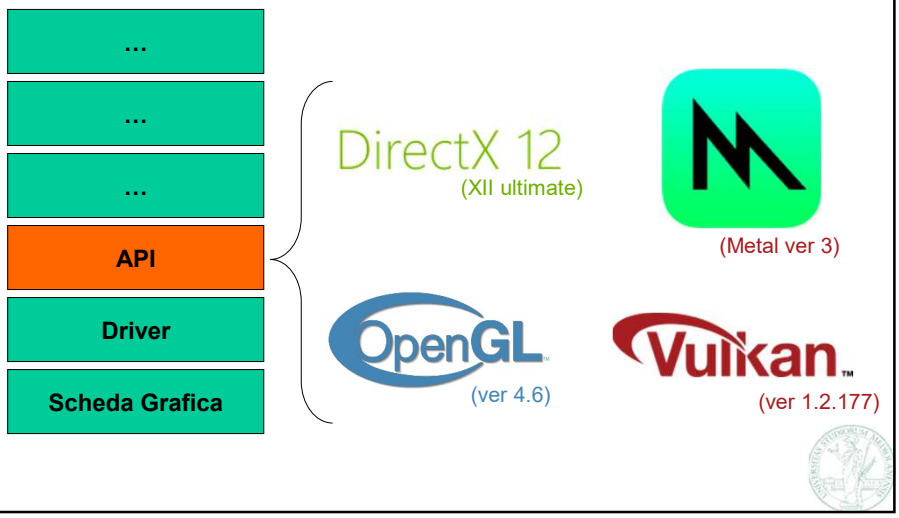
## Livelli di astrazione

Hardware specializzato per CG




21

## Livelli di astrazione




22

## OpenGL : storia

- ✓ inizialmente sviluppato da Silicon Graphics


- ✓ dal 2002 al 2006:
  OpenGL **A**rchitecture **R**eview **B**oard
  - ⇒ mantiene e aggiorna le *specifiche*
  - ⇒ industria 90%, accademia 10%
  - ⇒ ogni compagnia / gruppo, un voto
- ✓ dal 2006: ARB
  evolve nel Khronos Gruop


23

## OpenGL : variant degli API

OpenGL ES

⇒per **embedded devices**

⇒sottoinsieme di OpenGL (in pratica)

WebGL

– per **web**
– Ver 2.0: basato su ver 3.0
– HTML5
– un language binding in JavaScript
– Ver 2.0
– *soluz emergente per il 3D sul Web (senza plug-in!)*

24

## API grafiche diffuse

✓**Direct3D**

⇒Microsoft
- proprietario, e non cross platform

⇒DirectX è il nome collettivo

⇒Stessi scopi di OpenGL
- una API per usare le stesse GPU
- struttura non dissimile
  - di solito, meno elegante, più macchinoso
- C (e C++)

⇒E' l'alternativa più comune a OpenGL
- Grossomodo:
  - Direct3D = industrial standard (e XBOX)
  - OpenGL = industrial + academic standard
  - Metal = l'API usata su Mac


25

## API grafiche diffuse


⇒by Khronos (again)

⇒versione rpiù a basso livello delle API OpenGL

- “bytecode” for the shaders
- better debugging
- unified mobile / embedded / desktop

⇒Simile alla vers ≥12 di Direct3D

26

## API grafiche diffuse

✓Metal (Apple Inc.)

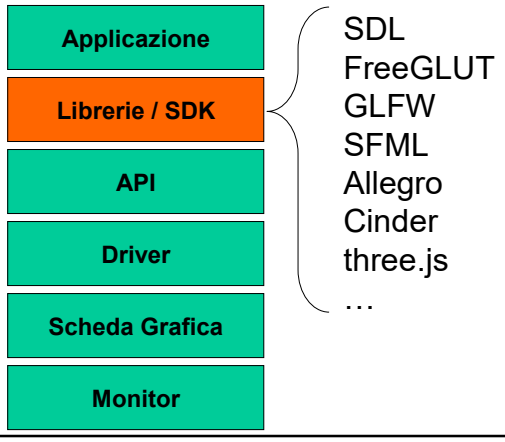
⇒Funzionalità simili
a OpenGL + OpenCL

⇒Basso livello

⇒Solo per iOS,
(e macOS, e tvOS)

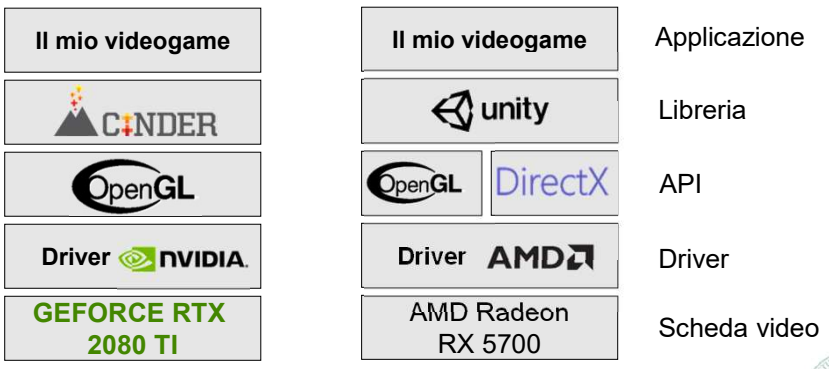
27

## Livelli di astrazione



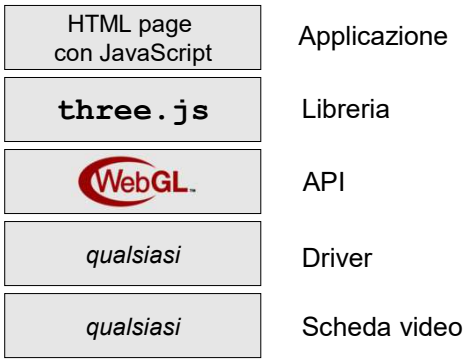

31

## Esempi




34

## I piccoli progetti che scriveremo in classe




35

## Three.js: grafica su Web made easy!

- ✓ Una lib ad alto livello basato su WebGL
  - ⇒ Cross platform, cross browser, cross vendor (al 100%)
  - ⇒ Moltissime utili funzioni grafiche, supportano
    - Mesh poligonali (quad, tri), splines, subdivision surfaces, voxel models, etc (compreso lettura di formati file standard)
    - Tessiture
    - Transformazion spaziali (modellazione, vista, proiezione…)
    - Luci, materiali, e lighting
    - Animazioni
    - Shaders (in GLSL)
    - Virtual reality
    - E molto altro
  - ⇒ Supporto allo sviluppo:
    - debuggers, molti esempi, documentazione…

36