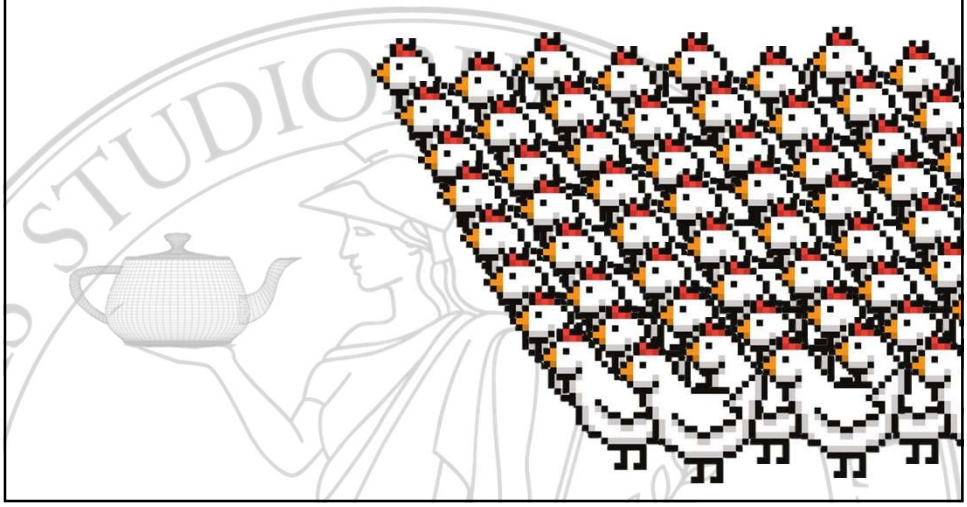
Marco Tarini - Grafica & Immagini Digitali 2025/2026
Università degli Studi di Milano

GPU, Drivers, API e librerie (in breve)



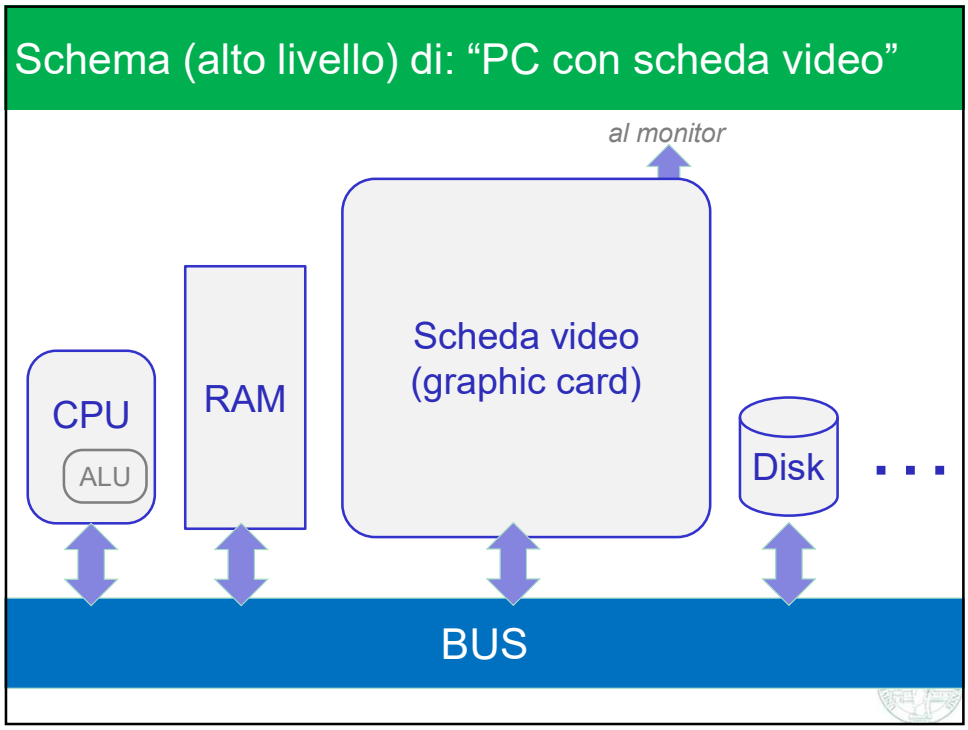
1

Hardware specializzato per il rendering

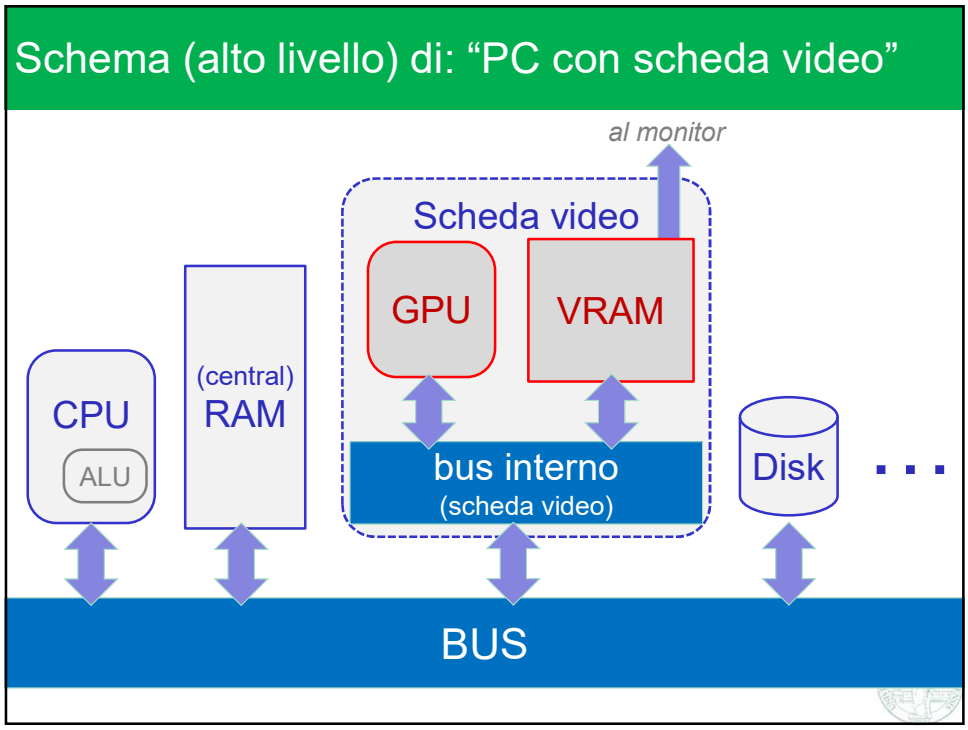
Visione di insieme:

- ✓ "**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

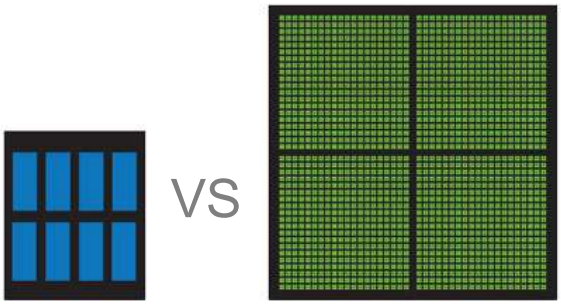

Schema (alto livello) di: “PC con scheda video”
al monitor
Scheda video
(graphic card)
CPU
ALU
RAM
Disk
BUS

4


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

5

CPU vs GPU

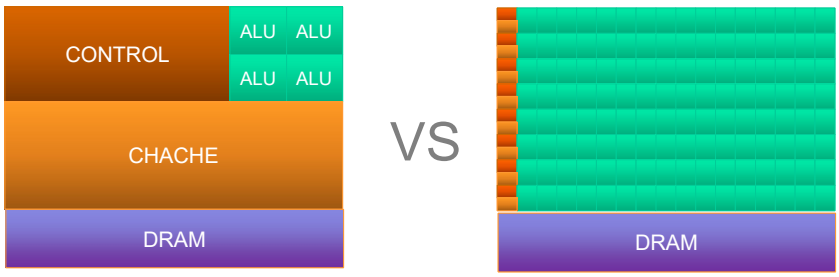


CPU
multiple cores
(per es, 6)

GPU
multiple cores
(per es, 2000)

6

CPU vs GPU



CPU

GPU

7

CPU vs GPU

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

Ad esempio, attualmente (metà 2020):

Una moderna GPU può avere una potenza di calcolo misurabile in centinaia di TERAFLOPS (cioè circa 10^14 FLOPS)

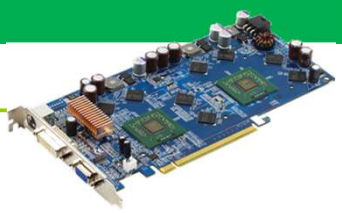
(FLOPS = FLoating-point OPeration per Second)

Una moderna CPU: «solo» qualche TERAFLOPS
(cioè circa 10^12 FLOPS, uno o due ordini di grandezza inferiore!)

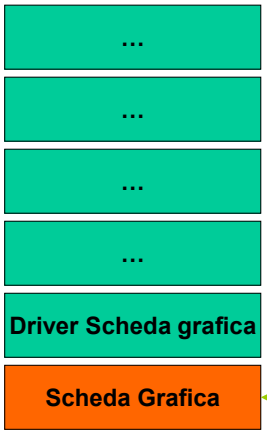
Questo divario si è mantenuto nelle generazioni

9

Livelli di astrazione

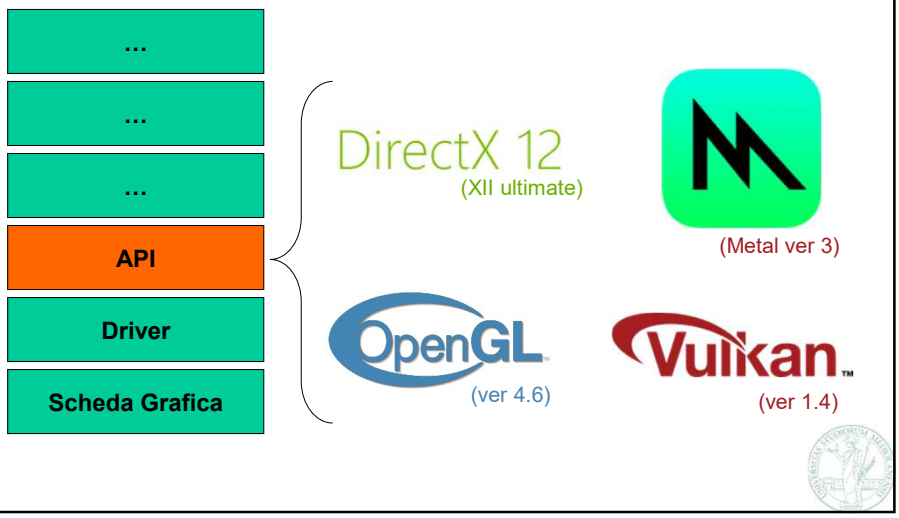


Hardware specializzato per CG



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 : varianti

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

– *soluzione stabile per il 3D sul Web*

24

API grafiche diffuse

✓**Direct3D**

⇒Microsoft

- proprietario, e non cross platform

⇒Parte di DirectX

⇒Stessi scopi di OpenGL

- una API per usare la GPU
- struttura non dissimile
- C (e C++)

⇒E' l'alternativa più comune a OpenGL

- Semplificando:
  - Direct3D = industrial standard (e XBOX)
  - OpenGL = industrial + academic standard
  - Metal = proprietaria Apple

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.)

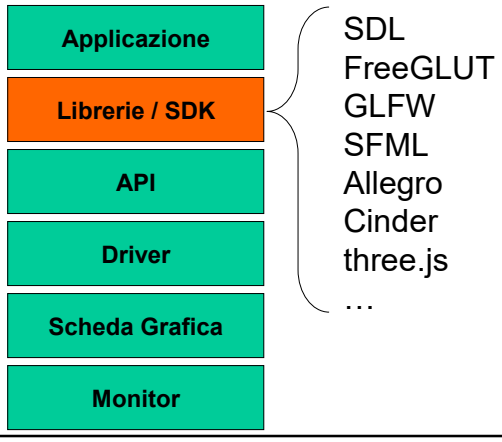
⇒Funzionalità a basso livello (come Vulkan o Direct3D)

⇒Solo per iOS, (e macOS, e tvOS…)

⇒Implementazioni in Swift, Objective-C, C++

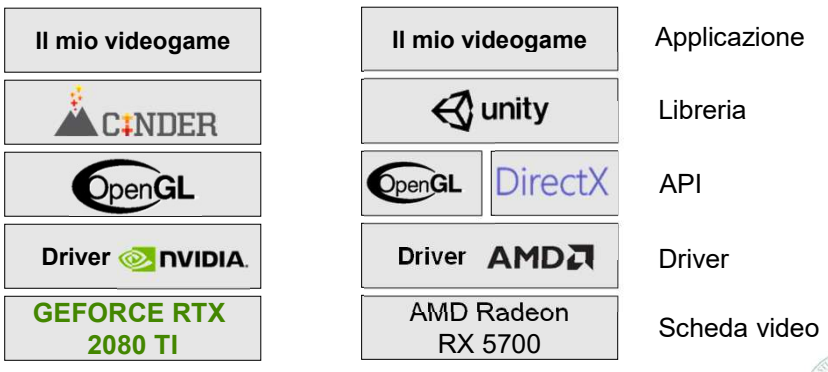
27

Livelli di astrazione



30

Esempi



33

I piccoli progetti che scriveremo in classe

Applicazione

Libreria

API

Driver

GPU

34

Three.js: CG su Web

✓ Una lib ad alto livello basato su WebGL

⇒Cross platform, cross browser, cross vendor (al 100%)

⇒Moltissime utili funzioni grafiche: support di

- Mesh poligonali (quad, tri), splines, subdivision surfaces, voxel models, etc (compreso, lettura da formati file standard)
- Tessiture
- Transformazion spaziali (modellazione, vista, proiezione…)
- Luci, materiali, e lighting
- Modelli animati ←
- Shaders (in linguaggio GLSL)
- Virtual Reality
- E molto altro

Argomento che non abbiamo trattato. Vedi cosi di Real-time Graphics Programming e/o 3D Videogames

⇒Supporto allo sviluppo:

- debuggers, molti esempi, documentazione…

37