

CODE3

Summer semester 2026

Prof. Dr.-Ing. Uwe Hahne

A trip down the 3D graphics pipeline

Learning goals: Lecture 01

Get an overview of what happens during rendering.

Get to know the basic terms, typical hardware and standard APIs.

What is a pipeline?

- Input → Command sequence → Output
- Command sequence consists of several steps
- The steps must be executed sequentially for a single data element
- For a larger amount of data, each step can be executed in **parallel**



In [computing](#), a **pipeline**, also known as a **data pipeline**, is a set of [data](#) processing elements connected in series, where the output of one element is the input of the next one. The elements of a pipeline are often executed in parallel or in time-sliced fashion.

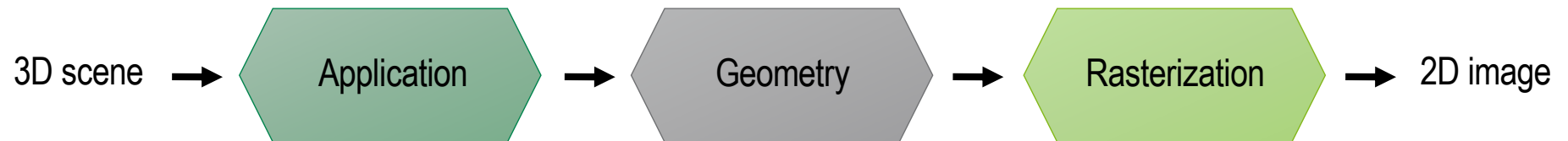
[Wikipedia](#)

What is a graphics pipeline?

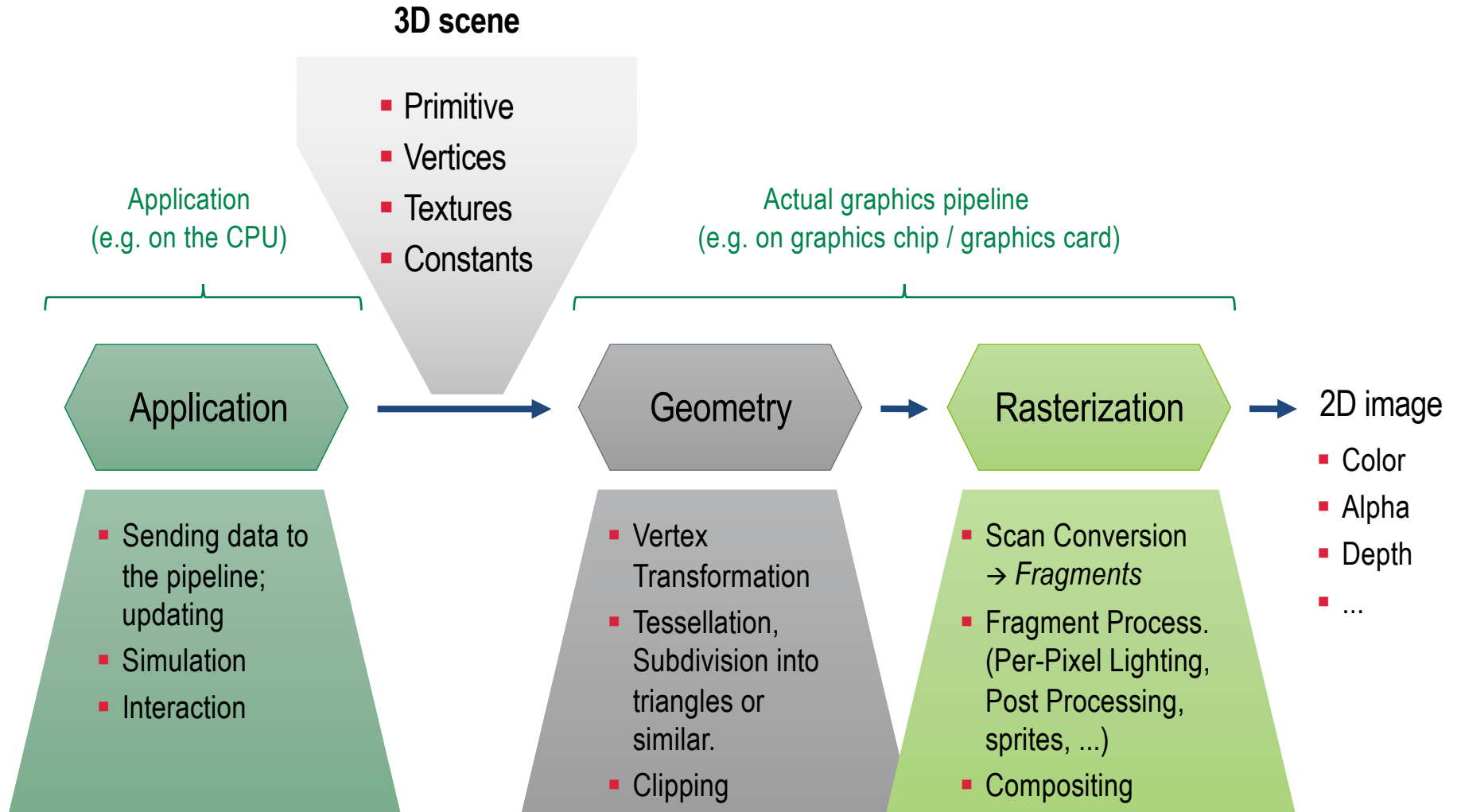
- Input: 3D scene description (polygons, transformations, ...)
- Output: 2D image
- What are the steps / what is the command sequence?



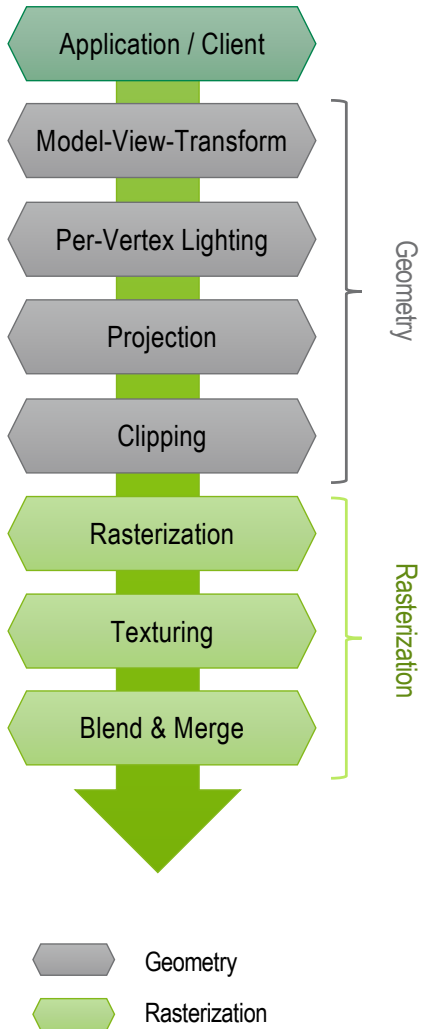
Conceptual model of the graphics pipeline



Conceptual model of the graphics pipeline



Classic graphics pipeline



- Functionality was hard-wired (fixed function pipeline)
- Configuration through many model parameters
 - Modelview matrix
 - Projection matrix
 - Light and material properties
 - Texturing modes
 - Alpha blending modes
 - Fog modes
 - ...

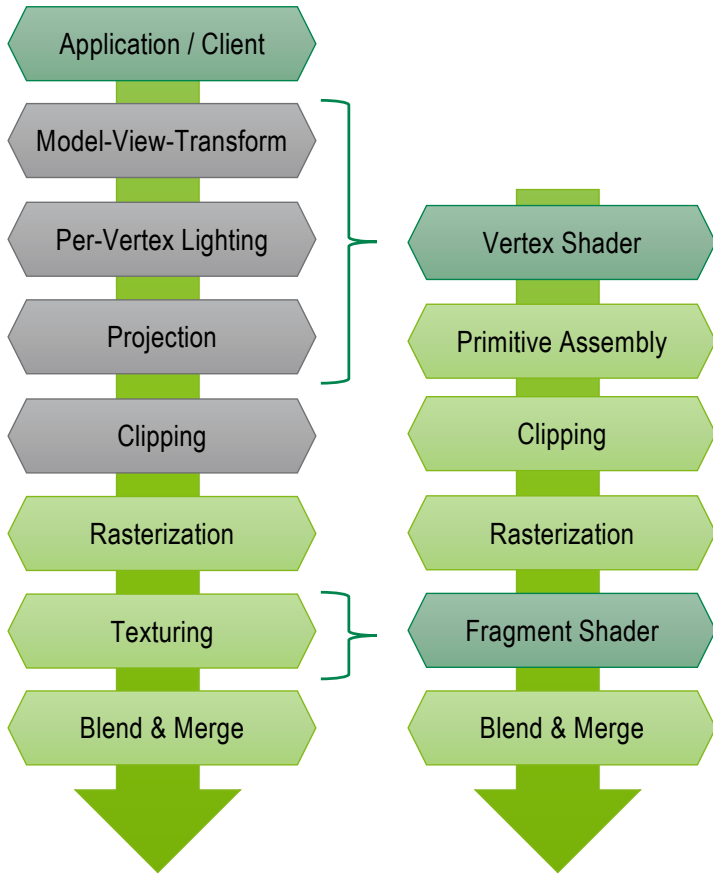
```
glMatrixMode(GL_MODELVIEW);  
glMultMatrix(...);
```

```
glLight(GL_LIGHT0, GL_POSITION, {5,4,-1,1} );  
glLight(GL_LIGHT0, GL_DIFFUSE, {1,1,1} );
```

```
glAlphaFunc(GL_GREATER, 0.2);
```

```
glFog(GL_FOG_MODE, GL_EXP);  
glFog(GL_FOG_COLOR, {1,1,1} );
```

From the classic to the programmable pipeline



```
// this is a shader executed in the pipeline
void main() {
    outPos = modelViewMatrix * inPos;
    normal = normalMatrix * inNormal;
    outColor = phong(outPos, normal);
    outPos = projectionMatrix * outPos;
}
```

Programmable shaders

- Since OpenGL 2.0 / DirectX 8.0 (2004/2000)
- Replace parts of the fixed function pipeline
- Pipeline becomes more flexible
 - Other lighting models
 - Other uses of textures and alpha blending
 - Realize effects directly *in the pipeline* using shaders

Geometry	Fixed Function
Rasterization	Programmable

From graphics computers to programmable GPUs

1981 - 200x: Silicon Graphics (SGI) graphics workstations

- 1987 SGI "4D" series → most popular computer for animation
- 1992 MIPS R4000, use of one of the first 64-bit processors

1996: 3DFX Voodoo, first PC graphics card

1999: Transform & Lighting (T&L) Unit

- NVIDIA Geforce 256 "the world's first GPU"; Pipeline on a single chip

2001: GeForce 3: First fixed-length Vertex Programs

2006-2011: Hardware tessellation

- Distribution through DirectX 11 and OpenGL 4.1

2011-2013: Compute Shaders

- For general, massively parallel calculations

2014: New proprietary low-level APIs

- AMD *Mantle*, NVIDIA *Cuda*
- Apple *Metal*
- Design goal: less "ballast" than OpenGL / DirectX
- DirectX 12 and Vulkan: similar design goals

1999
15 million triangles/sec.
480 million pixels/sec.



2010
100,000 million vertices/sec. *
40,000 million pixels/sec.



* See this [side node](#) about GPU performance measures

Art, Science and GPU's
Adam Savage & Jamie Hyneman
Explain Parallel Processing



From <https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/>
Video at: https://www.youtube.com/watch?v=8_ZTvG1WQxM

Inside view of a computer



Inside view of a computer

Find the CPU!
(central processing unit)



[This image](#) is licensed under [CC-BY 2.0](#)

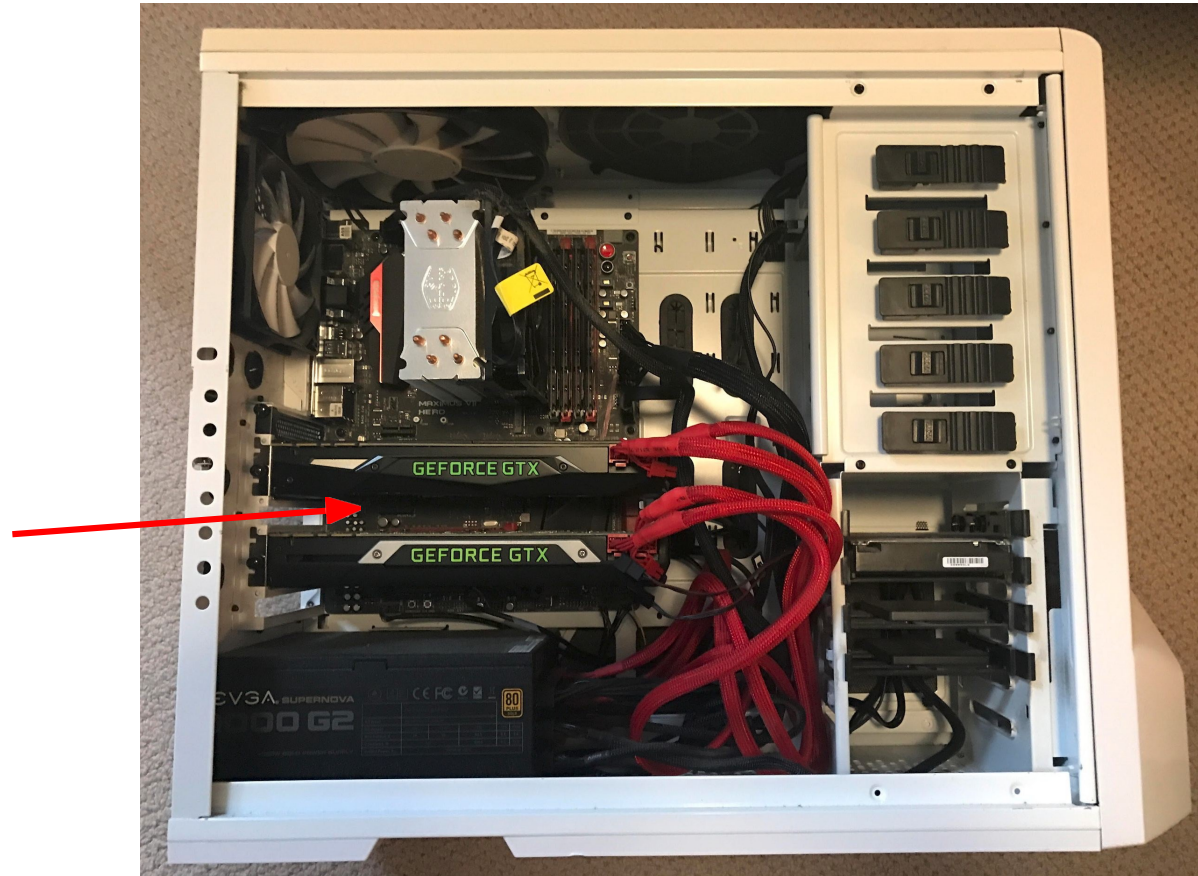


Inside view of a computer

Find the GPUs!
(graphics processing unit)



This image is licensed under [CC-BY 2.0](https://creativecommons.org/licenses/by/2.0/)



CPU vs GPU

	Cores	Clock Speed	Memory	Price	Speed (throughput)
CPU (Intel Core i9-7900k)	10	4.3 GHz	System RAM	\$385	~640 GFLOPS FP32
GPU (NVIDIA RTX 3090)	10496	1.6 GHz	24 GB GDDR6X	\$1499	~35.6 TFLOPS FP32

CPU: Fewer cores, but each core is much faster and more powerful; ideal for sequential tasks

GPU: More cores, but each core is much slower and "dumber"; ideal for parallel tasks

A graphics pipeline (anno 2004) in hardware

NVIDIA GeForce 6 graphics card (2004)

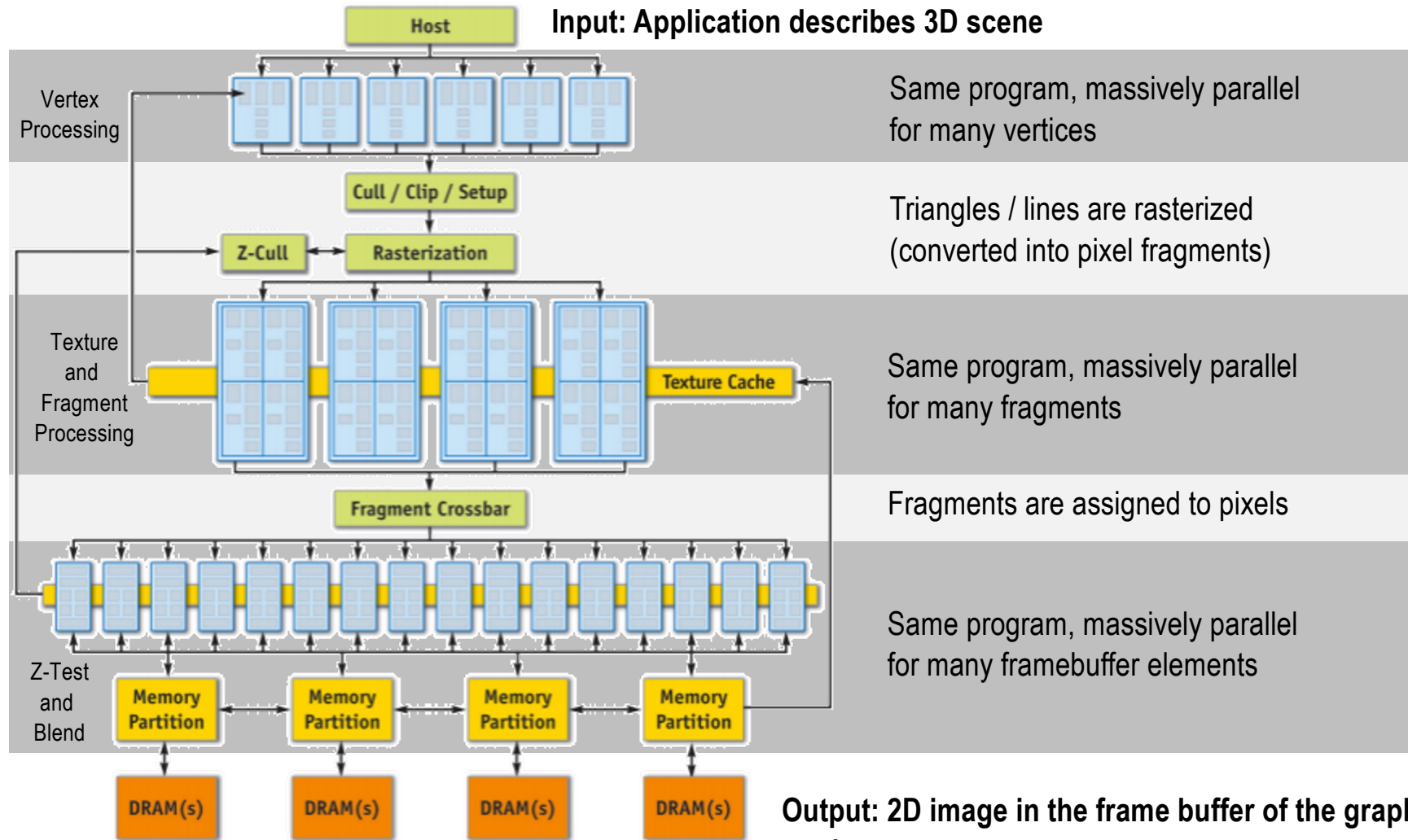


Image source: GPU Gems 2 (2005), ftp://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch30.pdf

Graphics pipeline: SIMD = Single Instruction, Multiple Data

NVIDIA GeForce 6 graphics card (2004)

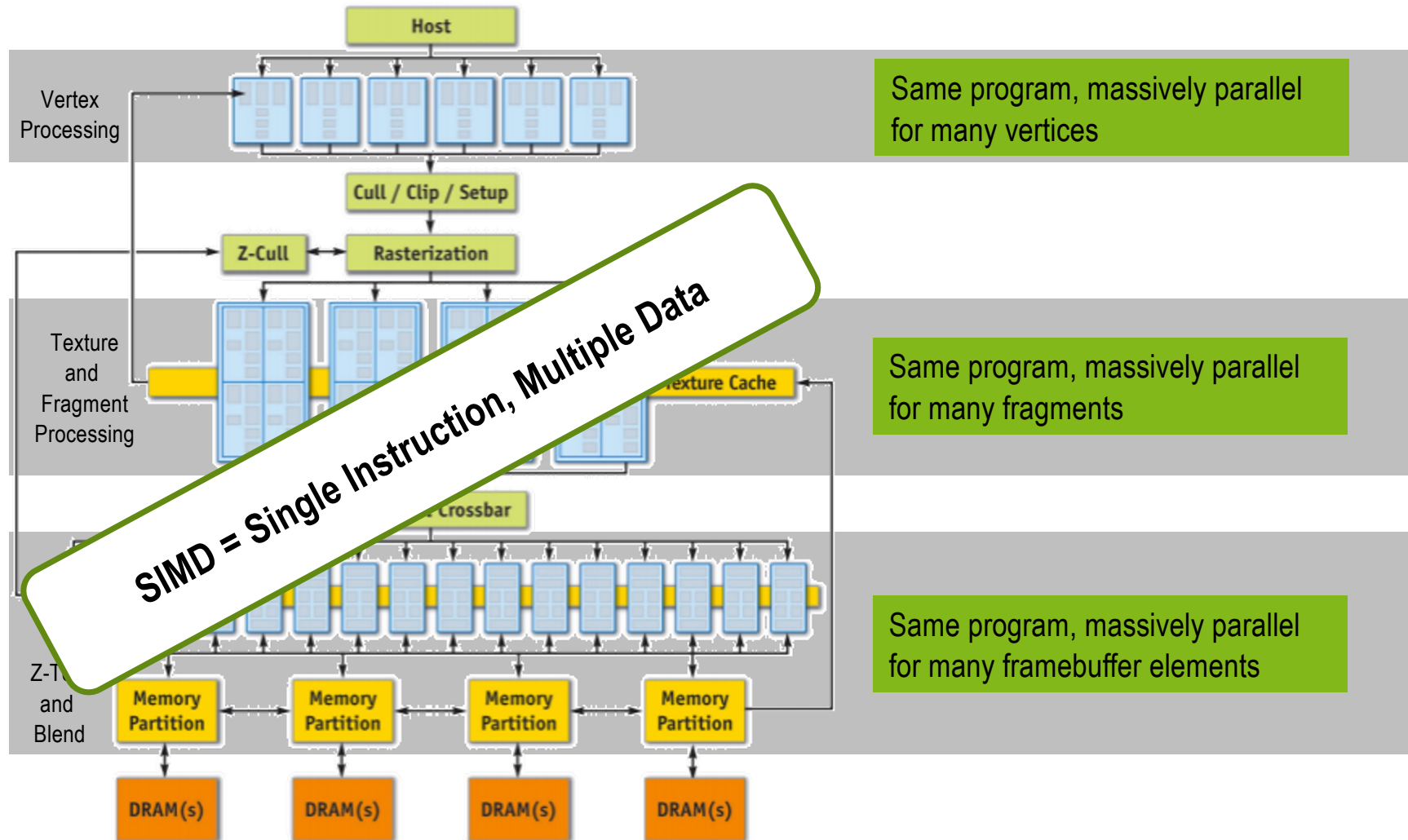


Image source: GPU Gems 2 (2005), ftp://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch30.pdf

Example: NVIDIA Jetson TX2 mobile chipset (2017)

	TX2 NX
Technical data of the graphics processor	
NVIDIA Pascal™ (number of computing units)	256
Technical data of the memory *:	
Standard memory configuration	4 x 4 GB LPDDR
Speed	51.2 GB/s
Power consumption	7.5 / 15 Watt

* Shared CPU+GPU memory

Example high-end GPU: NVIDIA RTX 3080 Ti (2021)

GeForce RTX 3080 Ti	
Technical data of the graphics processor	
NVIDIA CUDA computing units®	10.240
Boost clock (GHz)	1,67
Base clock rate (GHz)	1,37
Technical data of the memory:	
Standard memory configuration	12 GB GDDR6X
Width of the memory interface	384 bit
Speed	76 - 84 GB/s *
Power consumption	350 Watt

10,000+ shader cores running in parallel!

*** Approximation from [Wikipedia](#)**

Example high-end GPU: NVIDIA Titan RTX (2021)

	Titan RTX
Technical data of the graphics processor	
NVIDIA CUDA computing units®	4608 + 576 <u>Tensor Cores</u> + 72 RT Cores
Boost clock (GHz)	1,77
Base clock rate (GHz)	1,35
Technical data of the memory:	
Standard memory configuration	24 GB GDDR6X
Width of the memory interface	384 bit
Speed	672 GB/s
Power consumption	280 watts *

Different specialized cores!

* Thermal design power

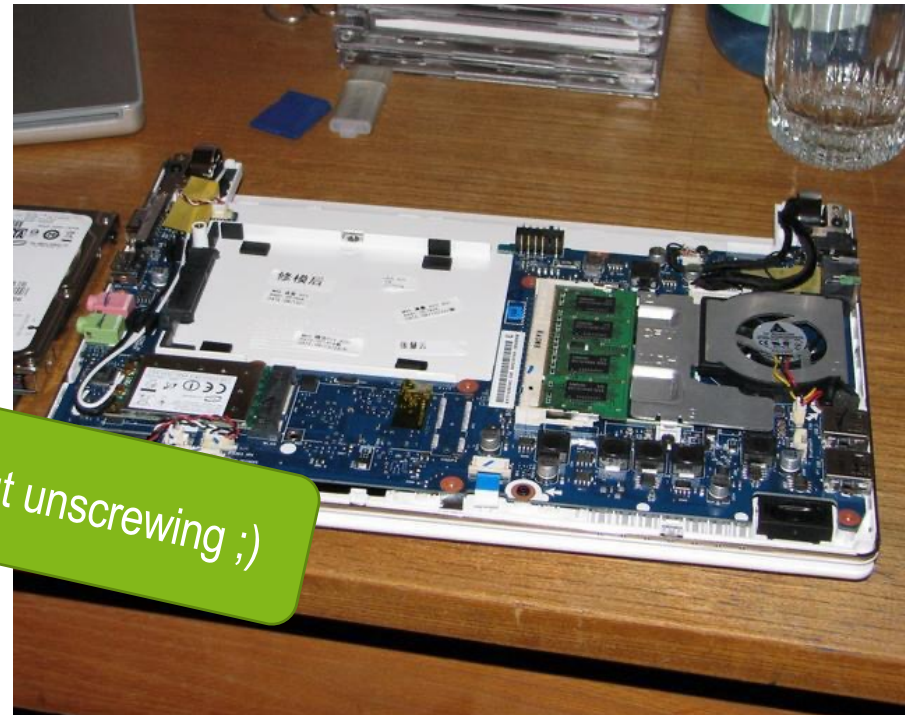
Example high-end GPU: NVIDIA 5090 RTX (2025)

	5090 RTX
Technical data of the graphics processor	
NVIDIA CUDA computing units®	21760 + 680 <u>Tensor Cores</u> + 170 RT Cores
Boost clock (GHz)	2,41
Base clock rate (GHz)	2,01
Technical data of the memory:	
Standard memory configuration	32 GB GDDR7
Width of the memory interface	512 bit
Speed	1792 GB/s
Power consumption	575 Watt

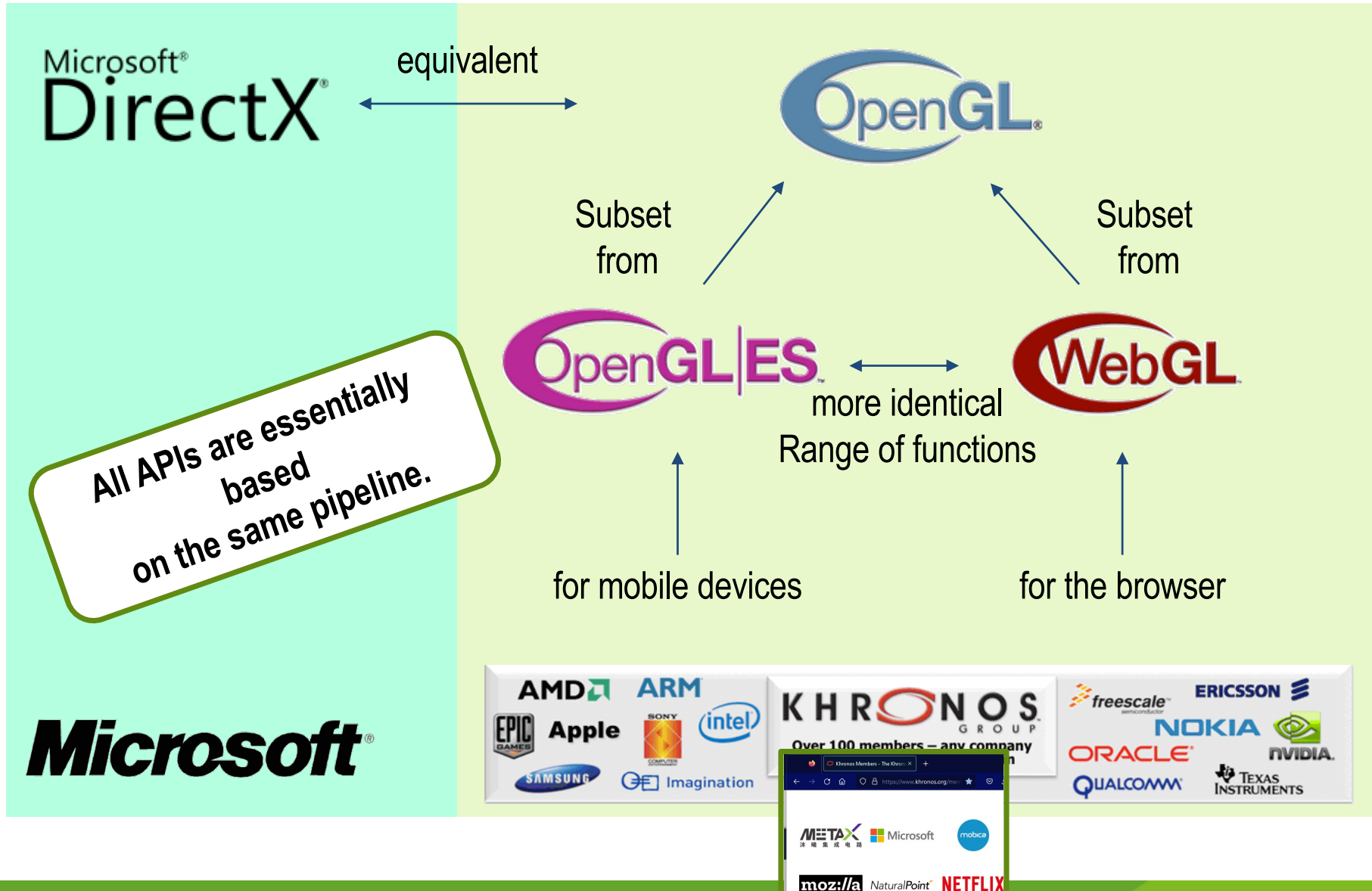
Different specialized cores!

Task

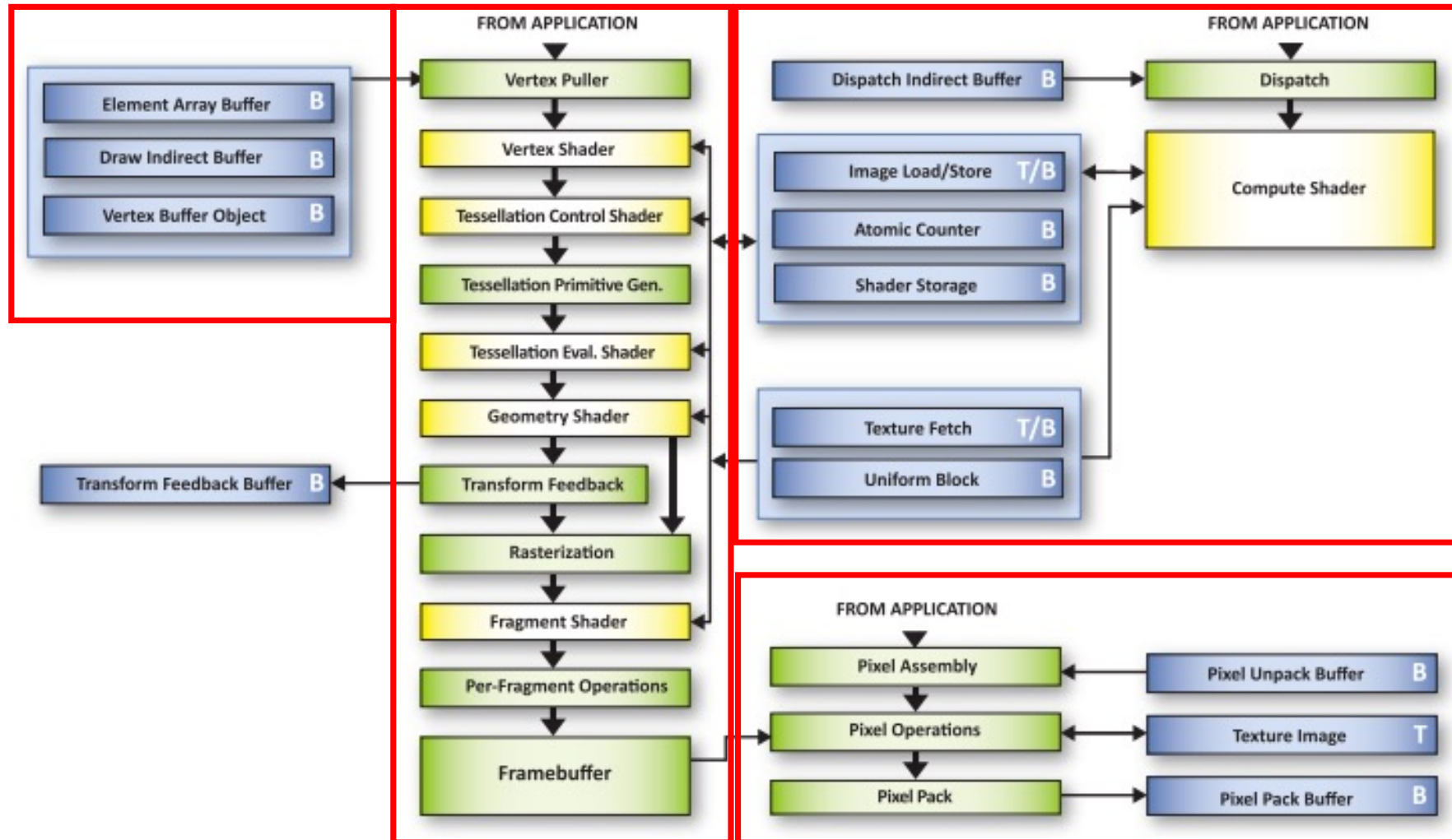
Take your laptop or smartphone and find out which CPU and GPU are installed in it.



Standard low level APIs for 3D graphics

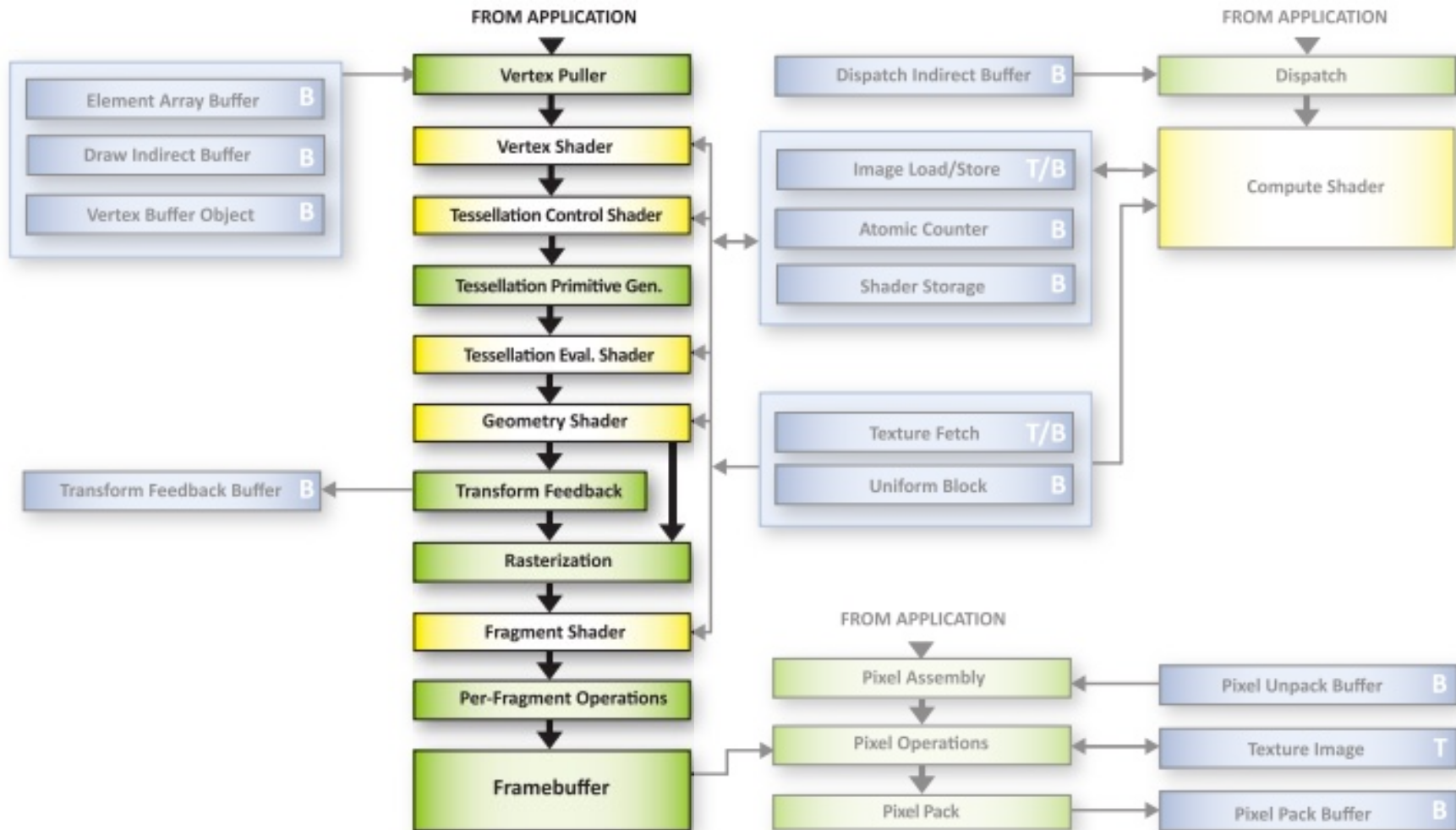


Complete OpenGL 4.6 pipeline (2017)



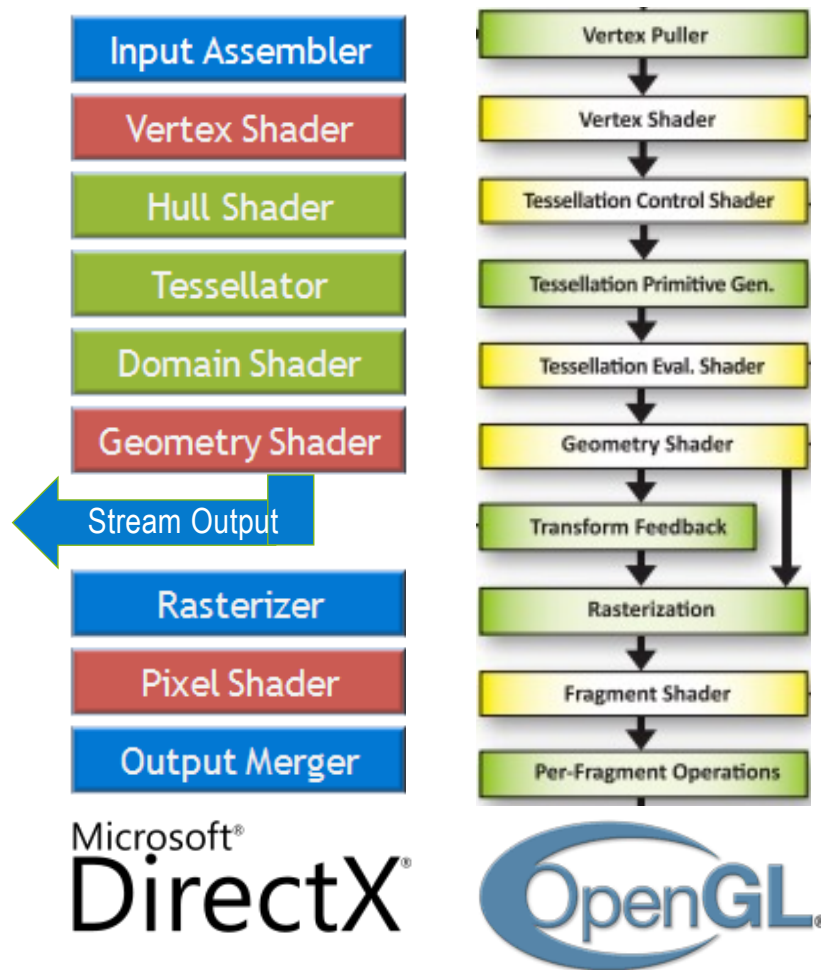
[http:](http://)

The core of the OpenGL pipeline



[http:](http://)

DirectX 11 vs. OpenGL 4 Pipeline



The two pipelines are very similar, except for terminology

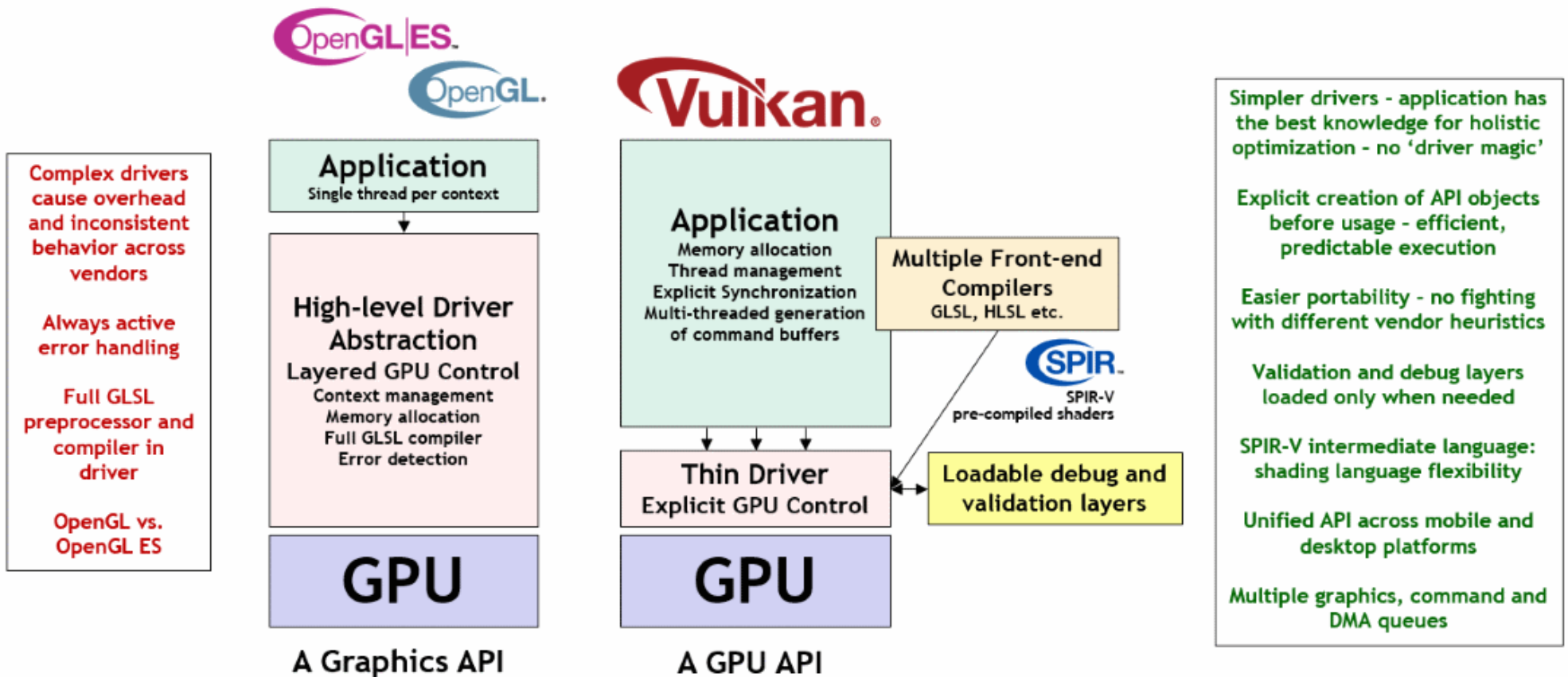
- "Input Assembler" = "Vertex Pulling"
- "Hull Shader" = "Tesselation Control Shader"
- ...

The simple reason for this:

- Both standards are developed for the latest graphics hardware
- Differences are usually only temporary until one standard has caught up with the other

Sources: Microsoft, Khronos Group

Vulkan: Performance, Predictability, Portability



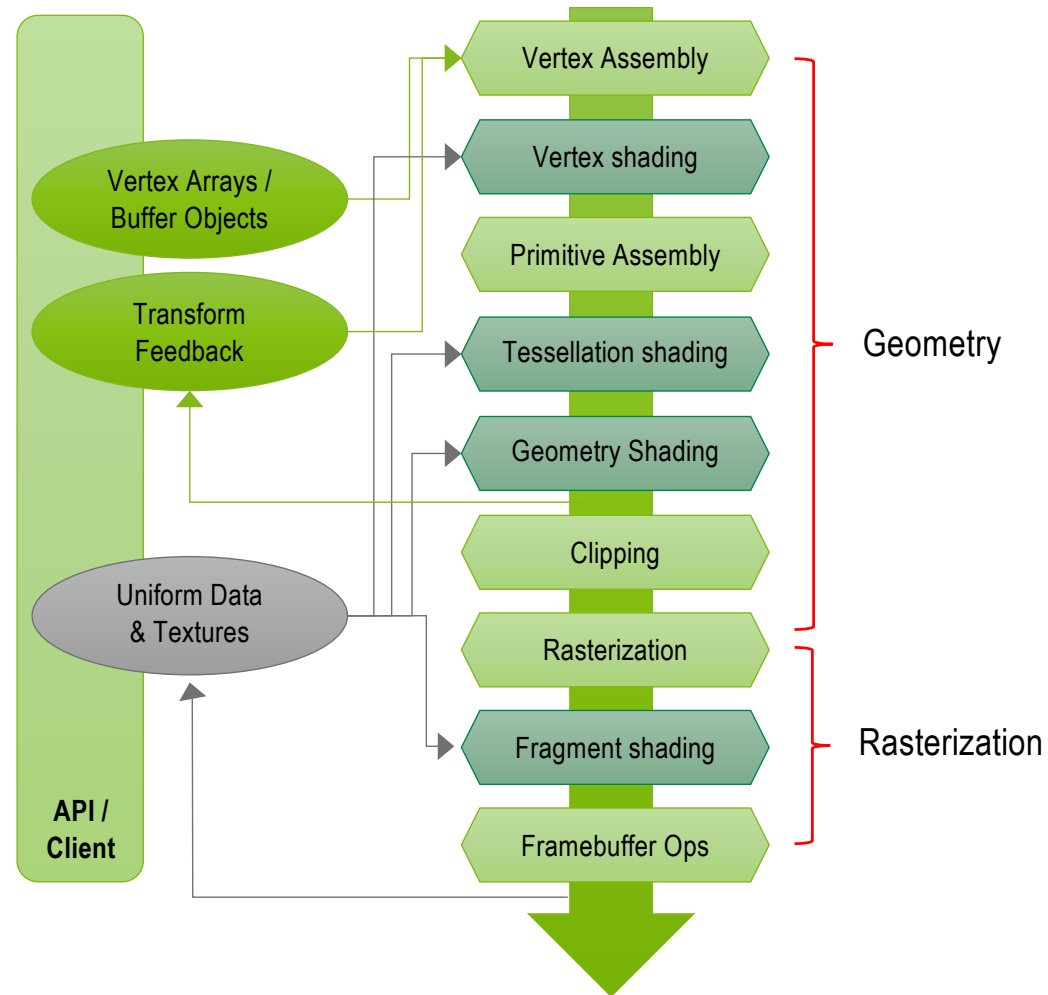
Source: Khronos Group

"Vulkan is not well-suited to simple test applications; neither is it a suitable aid for teaching graphics concepts."

— Graham Sellers, in the book *"Vulkan Programming Guide"*

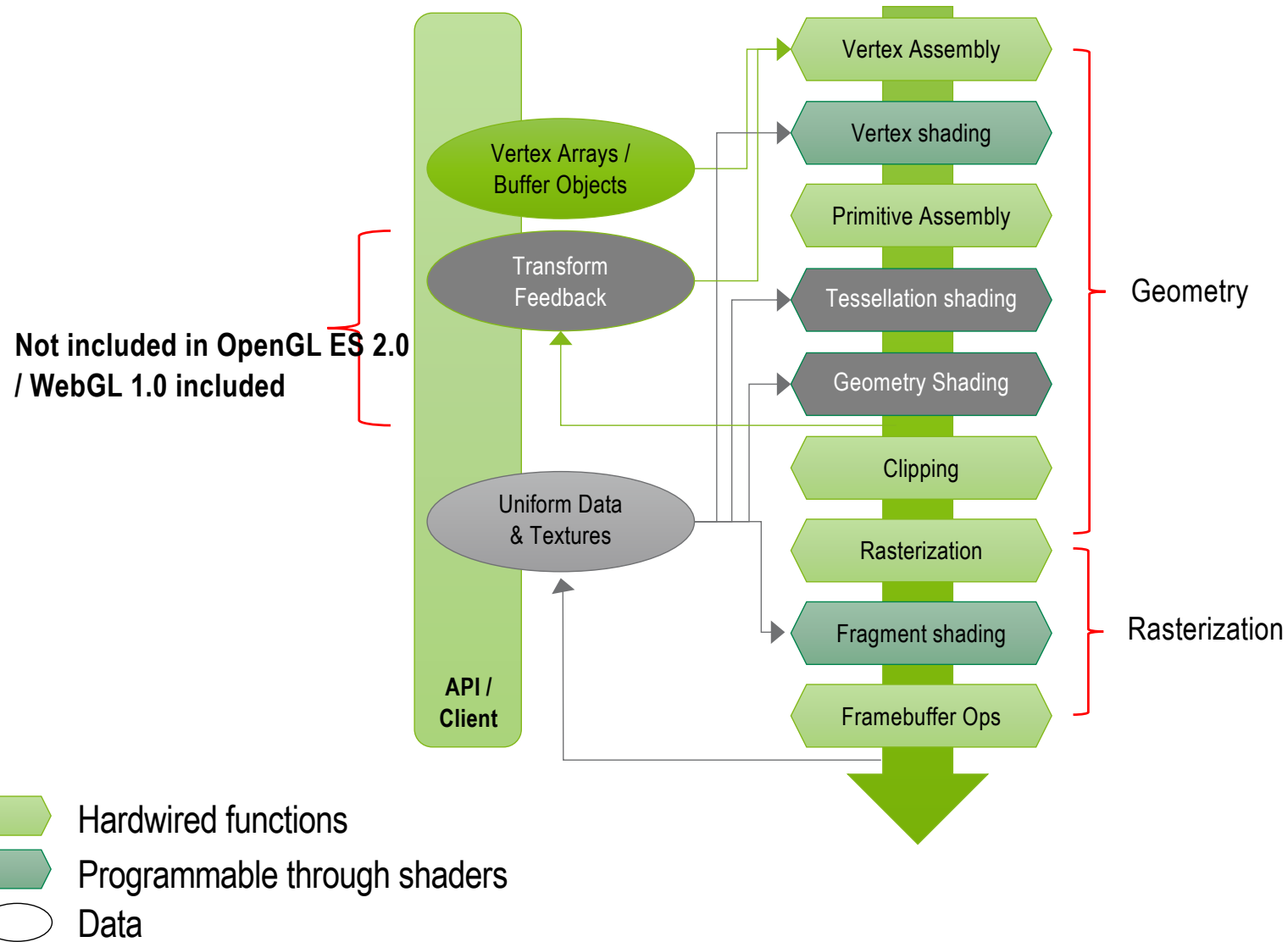


The (simplified) pipeline model of OpenGL

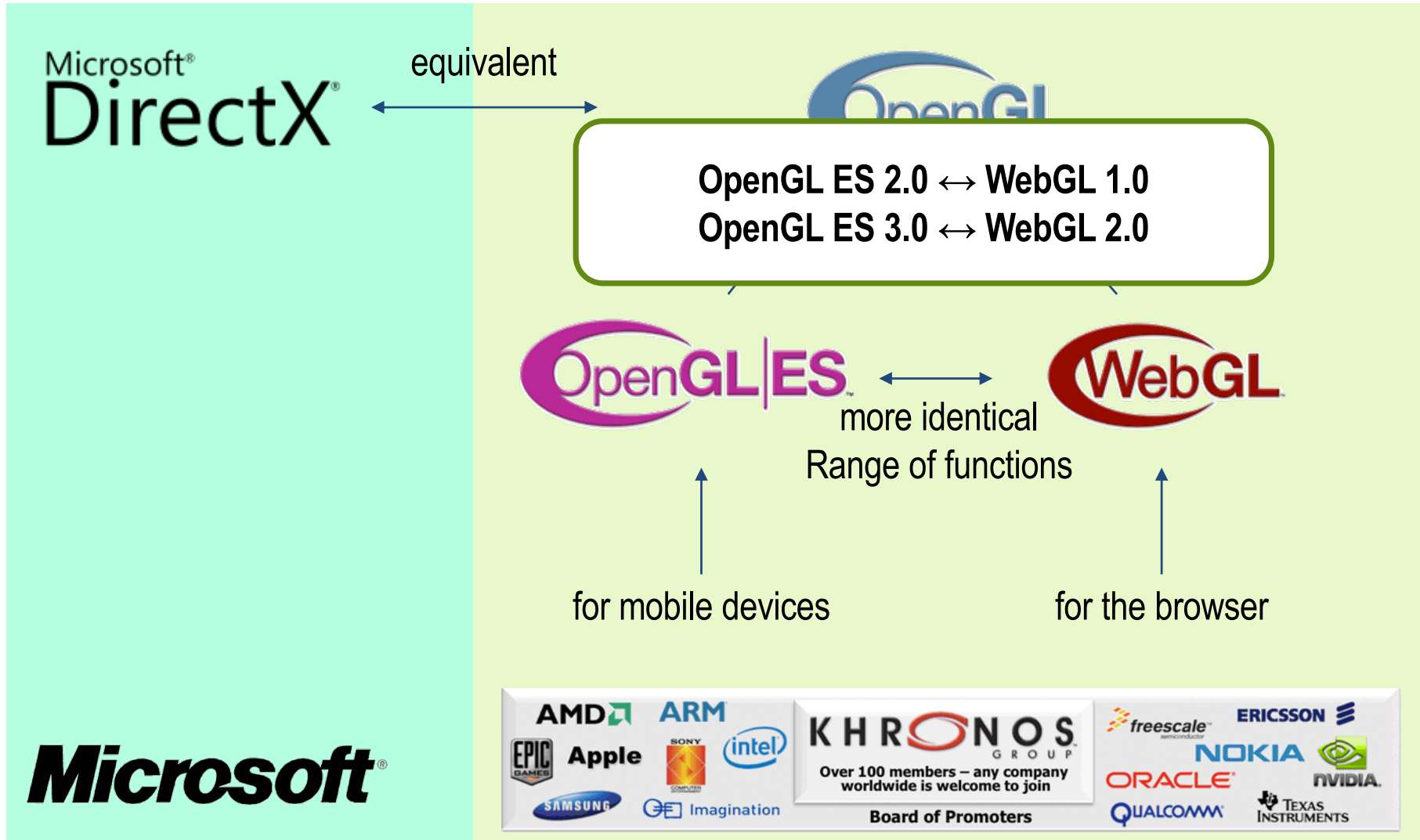


- Hardwired functions
- Programmable through shaders
- Data

The (simplified) pipeline model of OpenGL



Standard low level APIs for 3D graphics



Task: Check learning objective

Did you get an overview of what happens during rendering?

- Write all the basic terms, typical hardware and standard APIs that you can remember on a card (5 min).
- Find a partner and explain the terms to each other. Collect all the cards that you could not explain (well enough) and send the open questions to your instructor.