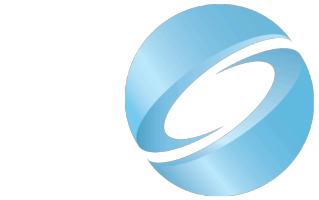
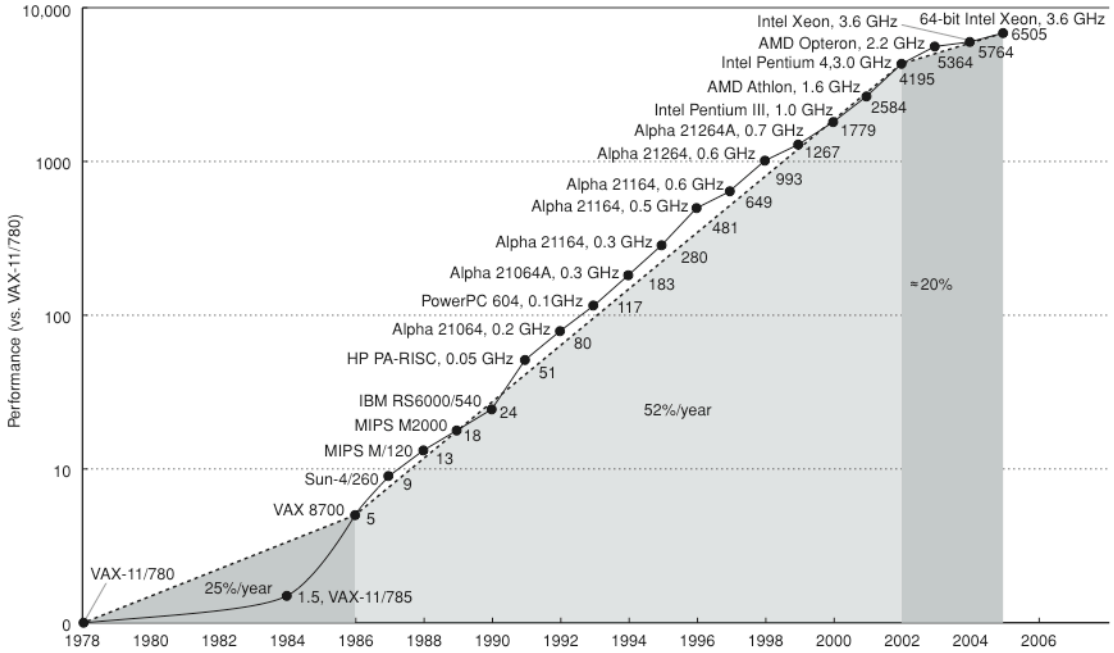


GPU Architecture Overview

John Owens
UC Davis

The Right-Hand Turn

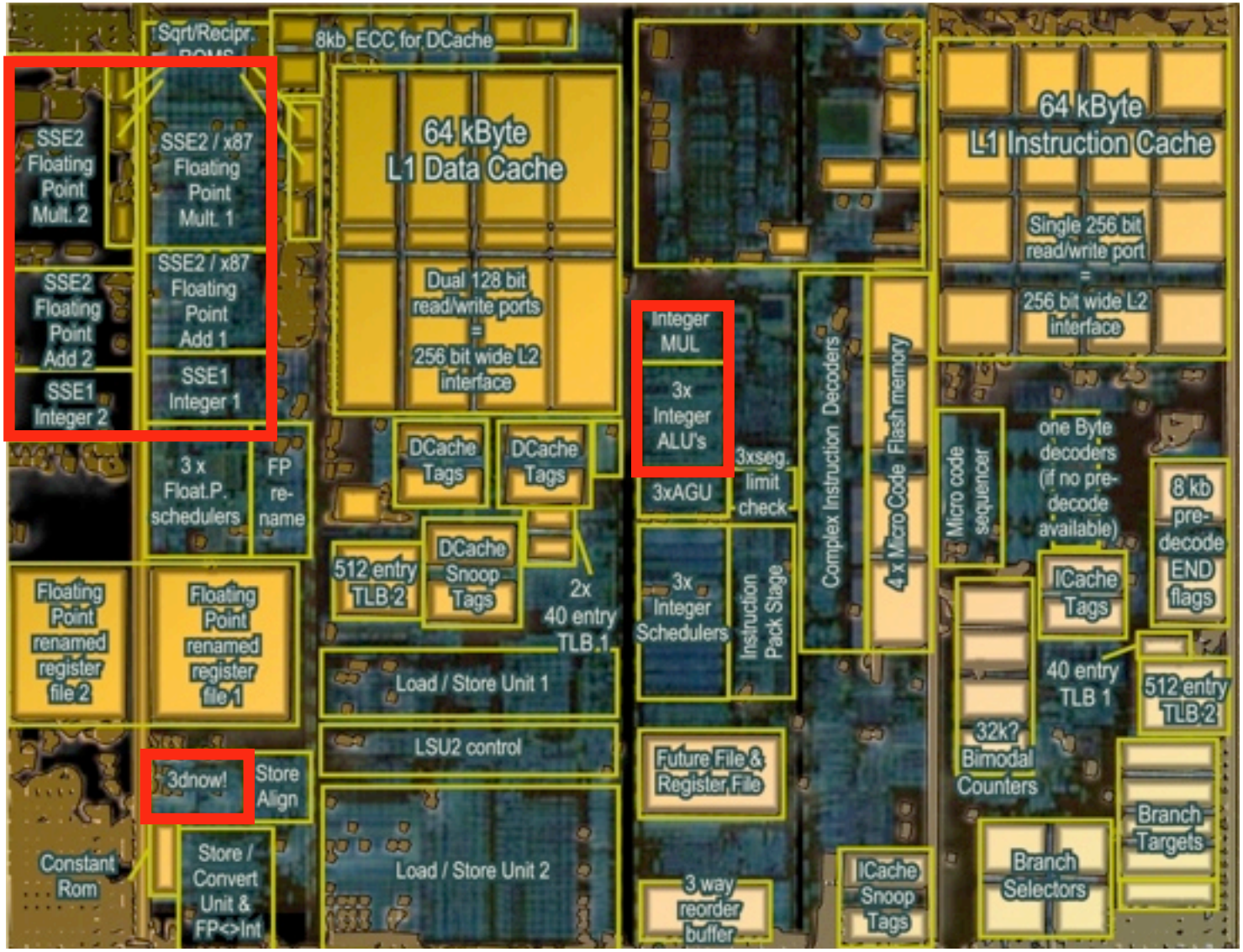


[H&P Figure 1.1]

Why? [Architecture Reasons]

- **ILP increasingly difficult to extract from instruction stream**
- **Control hardware dominates µprocessors**
  - Complex, difficult to build and verify
  - Takes substantial fraction of die
  - Scales poorly
    - Pay for max throughput, sustain average throughput
    - Quadratic dependency checking
  - Control hardware doesn't do any math!
    - Intel Core Duo: 48 GFLOPS, ~10 GB/s
    - NVIDIA G80: 330 GFLOPS, 80+ GB/s

AMD "Deerhound" (K8L)



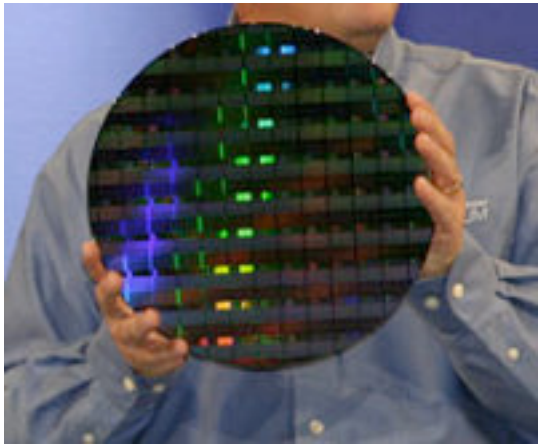
GPGPU

Why? [Technology Reasons]

- **Industry moving from "instructions per second" to "instructions per watt"**
  - "Power wall" now all-important
  - Traditional µproc techniques are not power-efficient
- **We can continue to put more transistors on a chip …**
  - … but we can't scale their voltage like we used to …
  - … and we can't clock them as fast …

Go Parallel

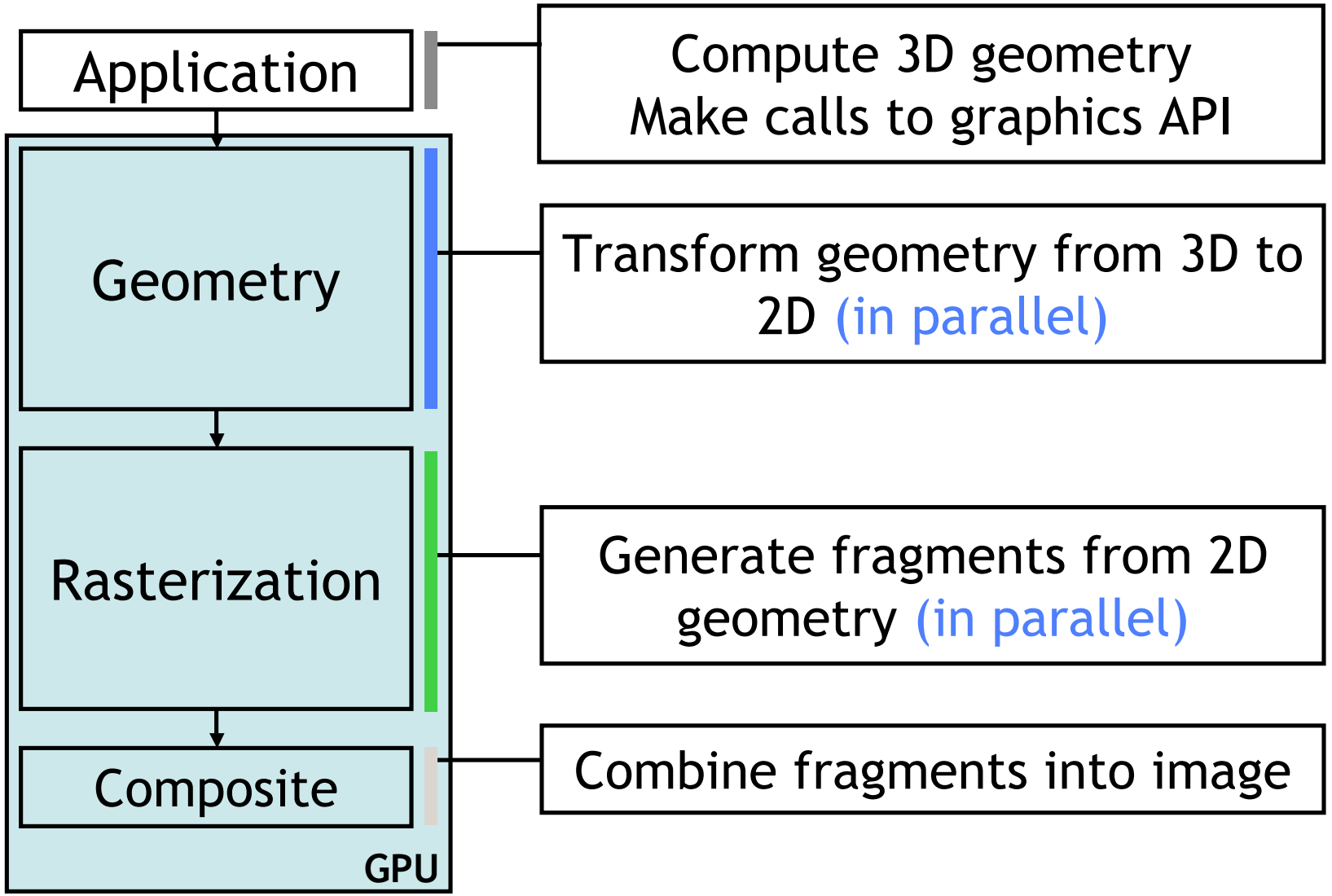
- **Time of architectural innovation**
  - GPUs let us explore using hundreds of processors now, not 10 years from now
- **Major CPU vendors supporting multicore**
- **Interest in general-purpose programmability on GPUs**
- **Universities must teach thinking in parallel**



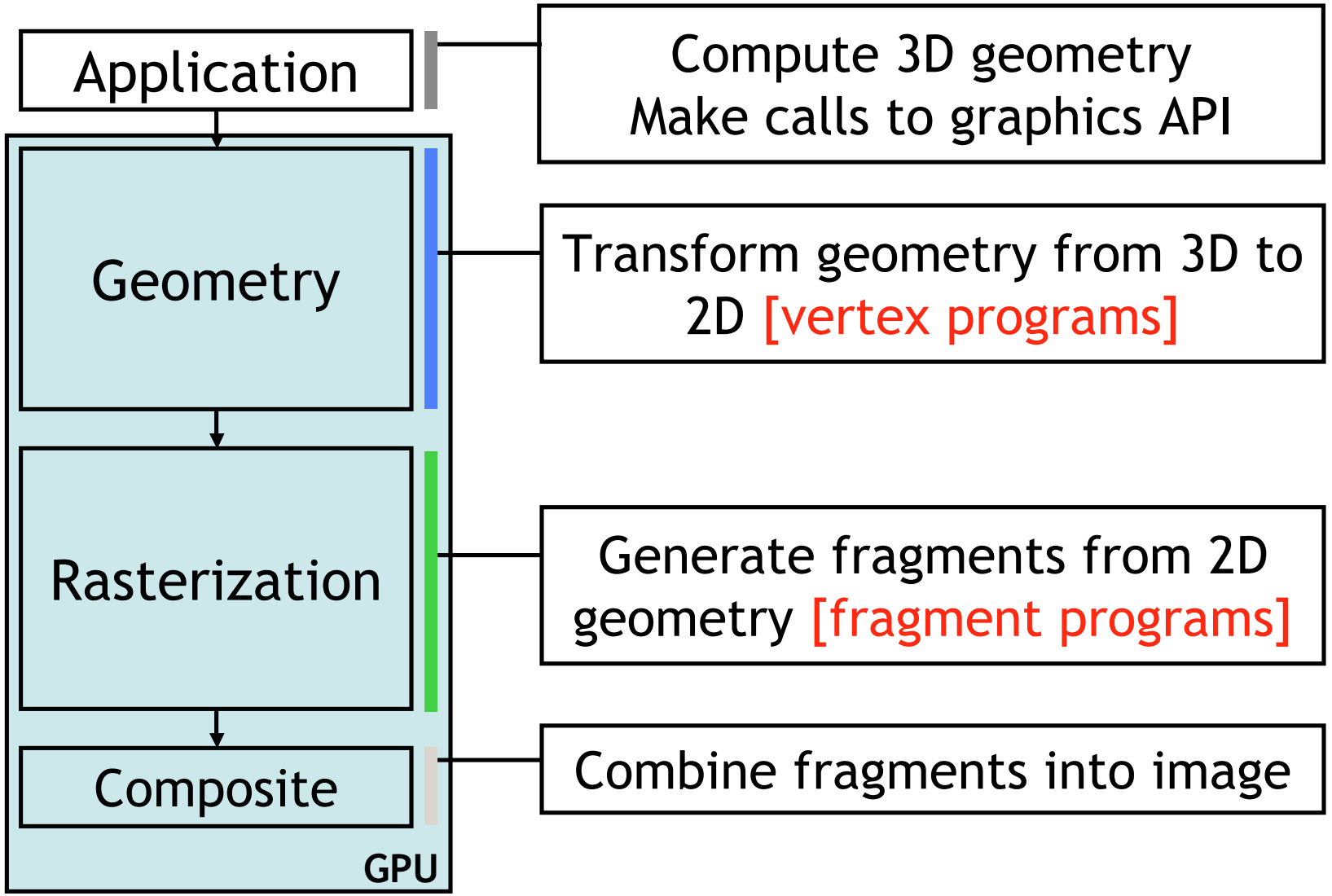
What's Different about the GPU?

- **The future of the desktop is parallel**
  - We just don't know what kind of parallel
- **GPUs and multicore are different**
  - Multicore: Coarse, heavyweight threads, better performance per thread
  - GPUs: Fine, lightweight threads, single-thread performance is poor
- **A case for the GPU**
  - Interaction with the world is visual
  - GPUs have a well-established programming model
  - Market for GPUs is 500M+ total/year

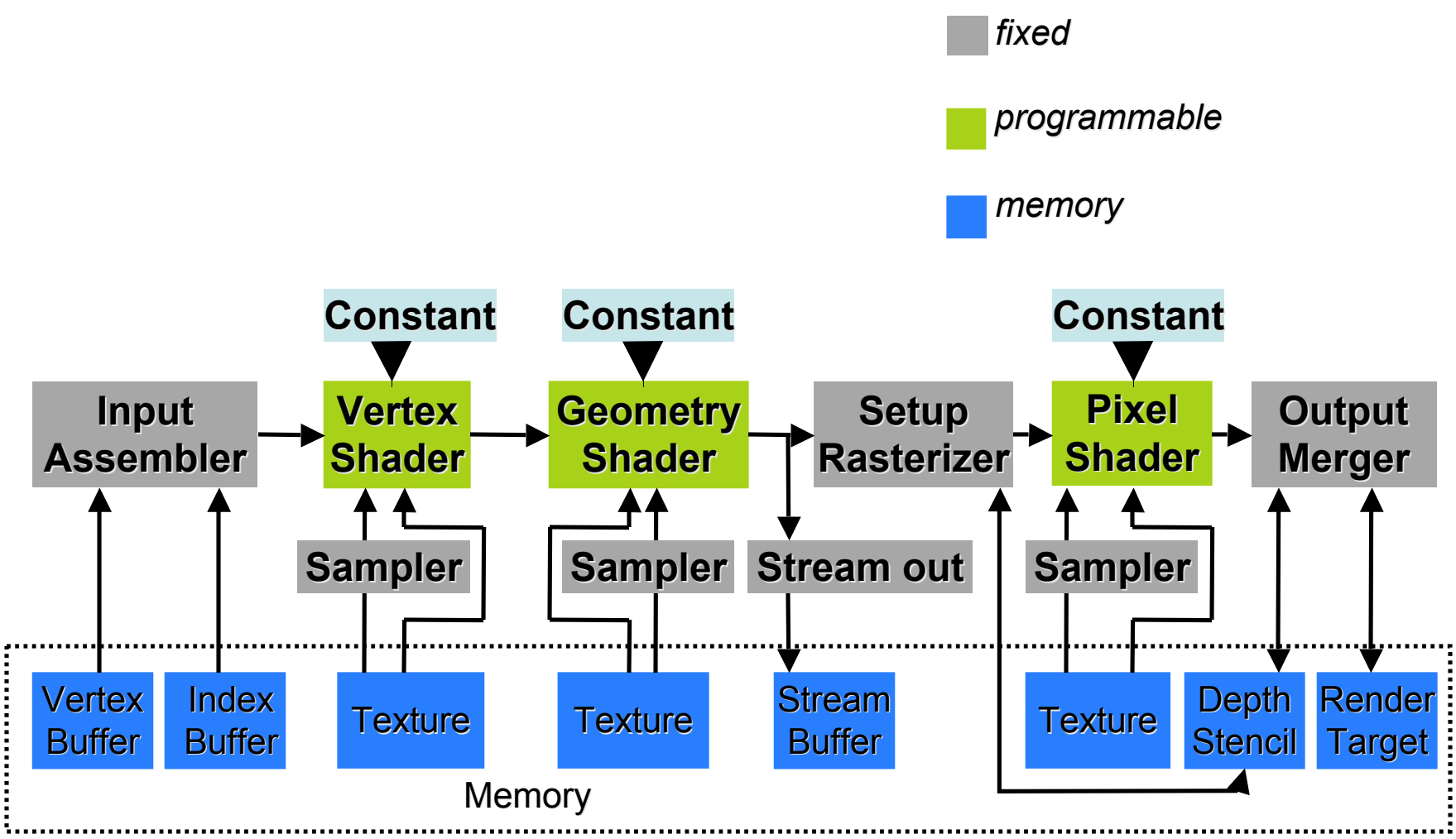
The Rendering Pipeline



The Programmable Pipeline



DirectX 10 Pipeline

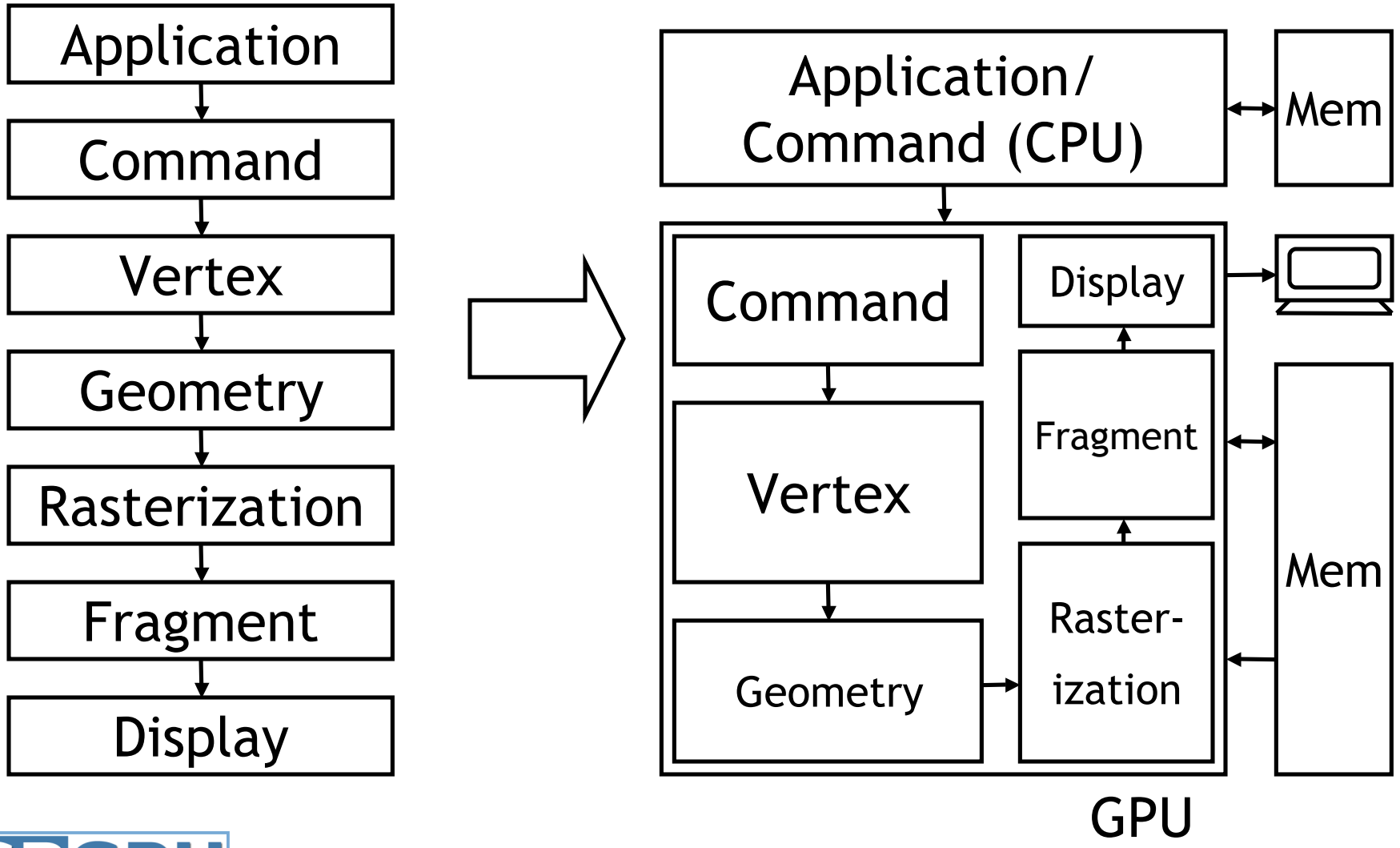


Courtesy David Blythe, Microsoft

Characteristics of Graphics

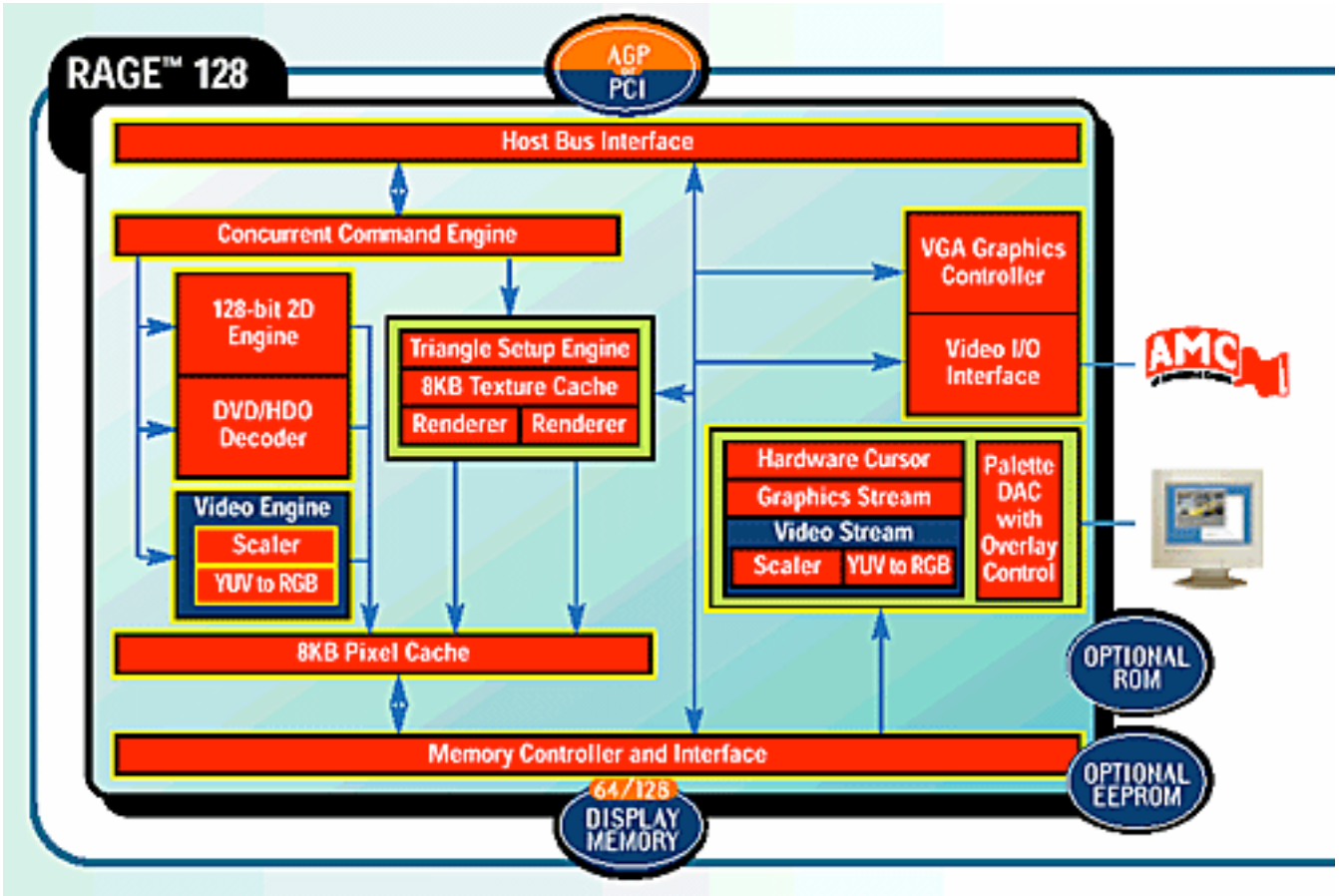
- Large computational requirements
- Massive parallelism
  - Graphics pipeline designed for independent operations
- Long latencies tolerable
- Deep, feed-forward pipelines
- Hacks are OK—can tolerate lack of accuracy
- GPUs are good at parallel, arithmetically intense, streaming-memory problems

Graphics Hardware—Task Parallel

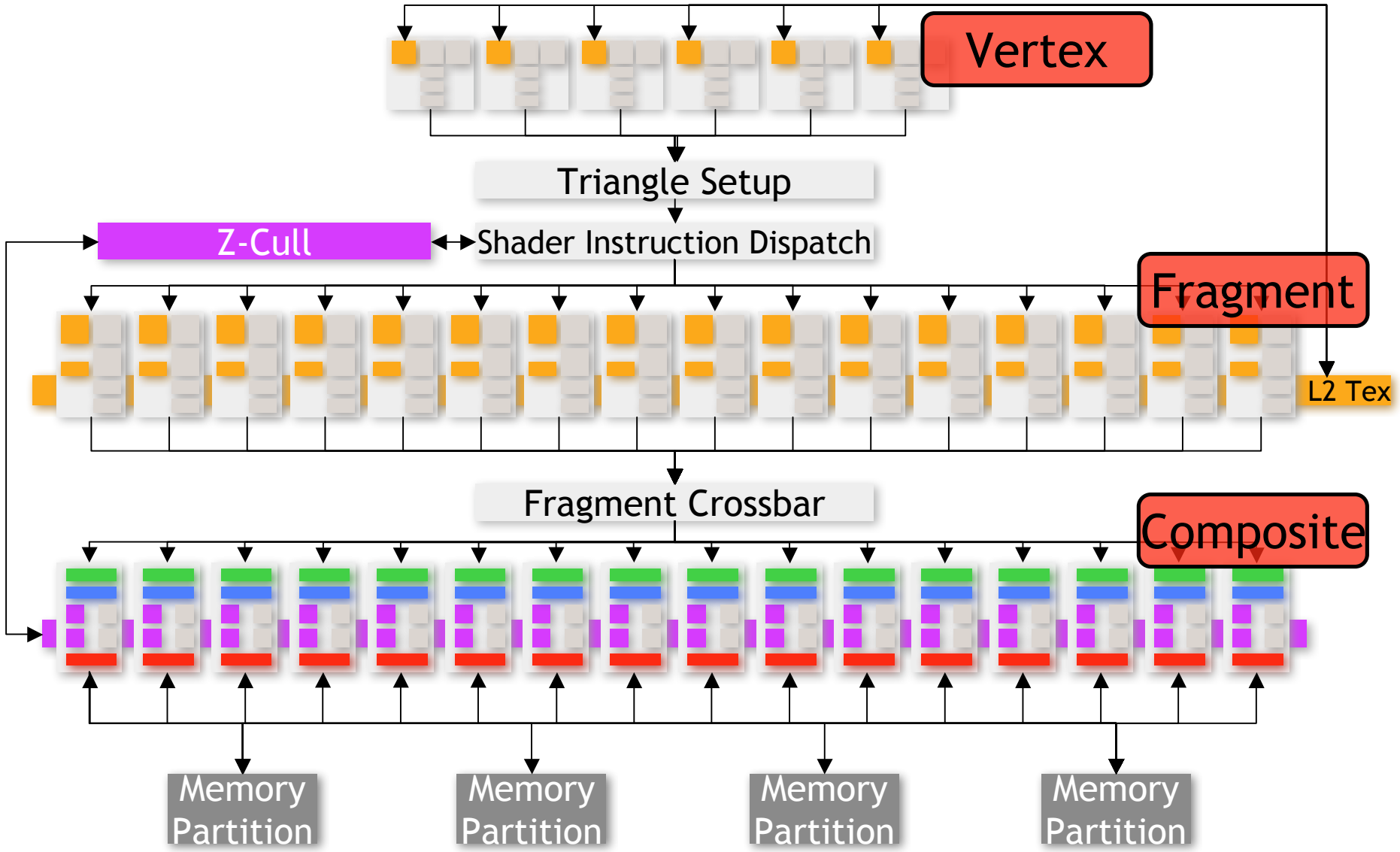


GPGPU

Rage 128


RAGE™ 128
AGP
PCI
Host Bus Interface
Concurrent Command Engine
128-bit 2D Engine
DVD/HDO Decoder
Video Engine
Scaler
YUV to RGB
Triangle Setup Engine
8KB Texture Cache
Renderer
Renderer
8KB Pixel Cache
VGA Graphics Controller
Video I/O Interface
Hardware Cursor
Graphics Stream
Video Stream
Scaler
YUV to RGB
Palette DAC with Overlay Control
AMC
OPTIONAL ROM
OPTIONAL EEPROM
Memory Controller and Interface
64/128
DISPLAY MEMORY

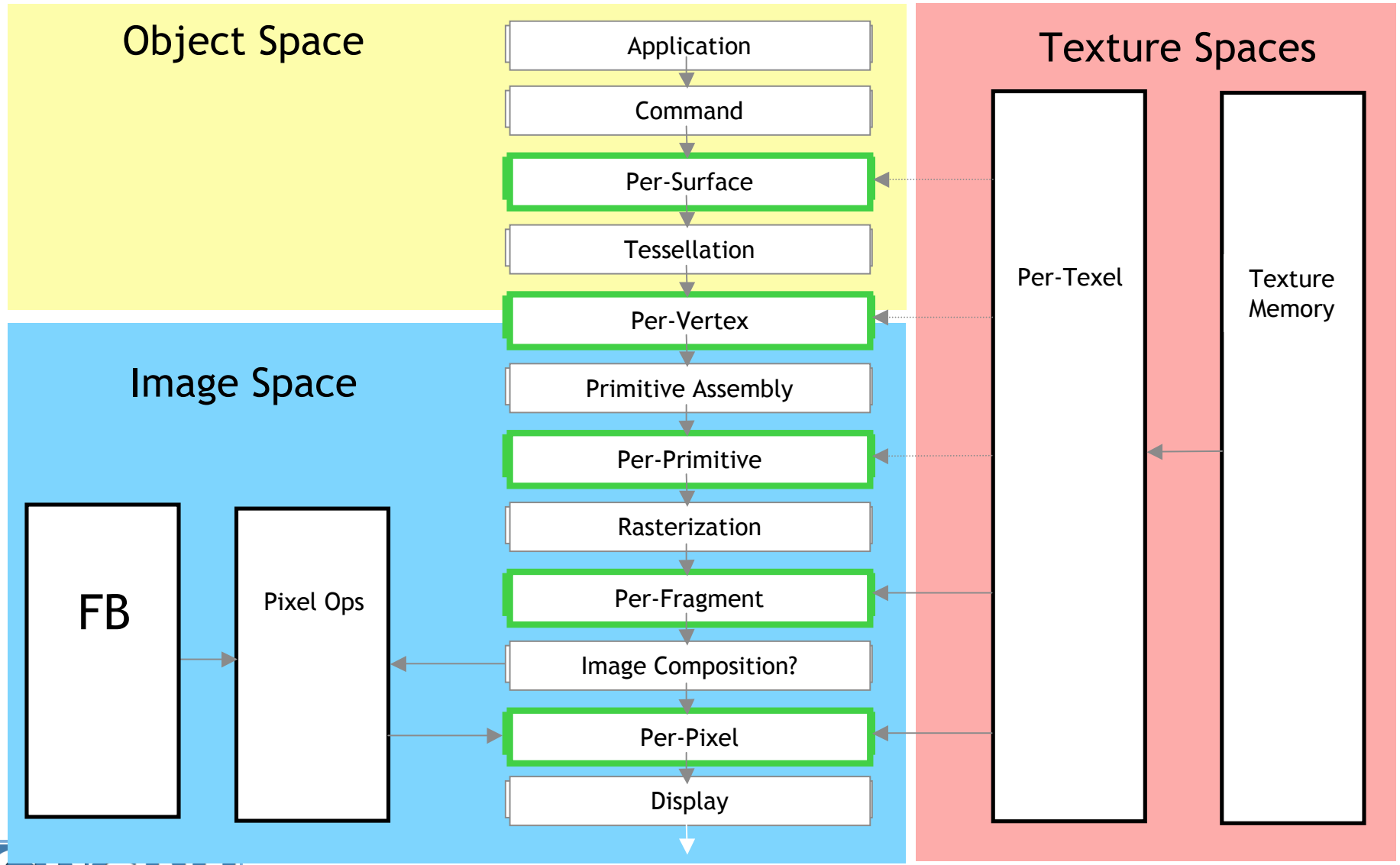
NVIDIA GeForce 6800 3D Pipeline



GPGPU

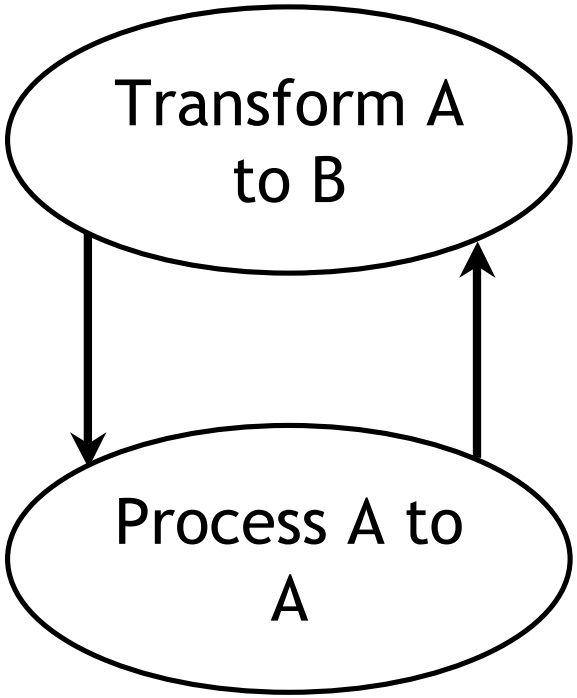
Courtesy Nick Triantos, NVIDIA

Programmable Pipeline



[From Akeley and Hanrahan, Real-Time Graphics Architectures]

Generalizing the Pipeline

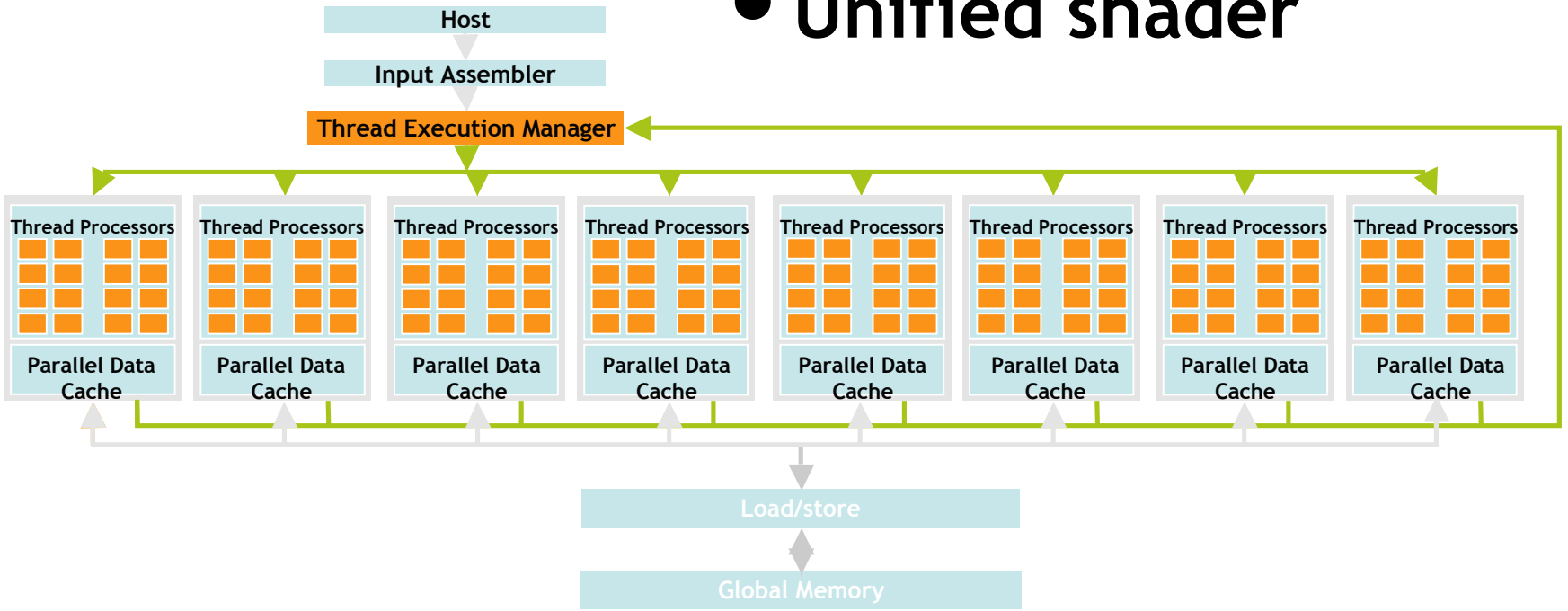


- **Transform A to B**
  - Ex: Rasterization (triangles to fragments)
  - Historically fixed function
- **Process A to A**
  - Ex: Fragment program
  - Recently programmable, and becoming more so

GeForce 8800 GPU

SIGGRAPH2007

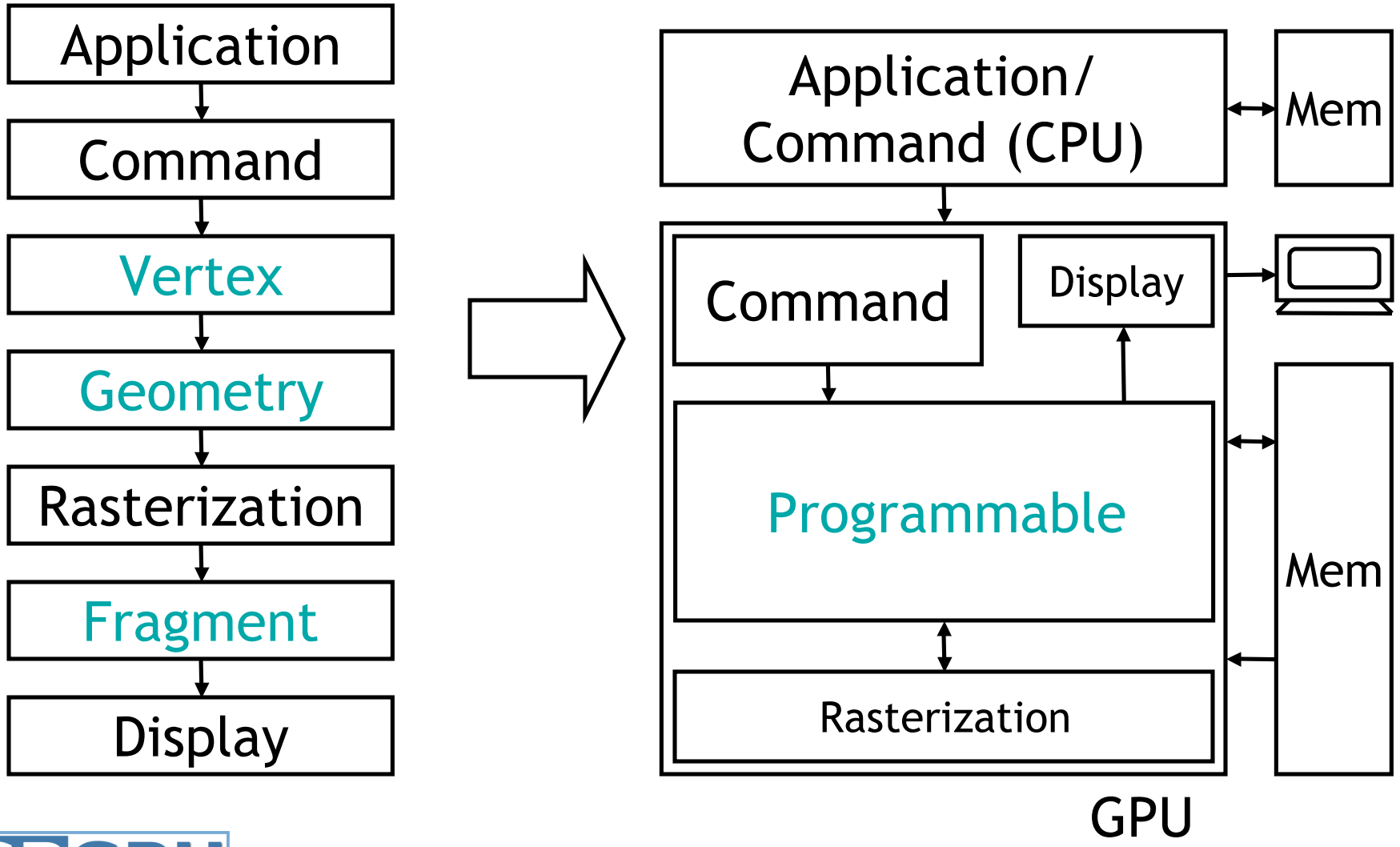
- Built around programmable units
- Unified shader



GPGPU

[courtesy of Ian Buck, NVIDIA]

Unified Shaders

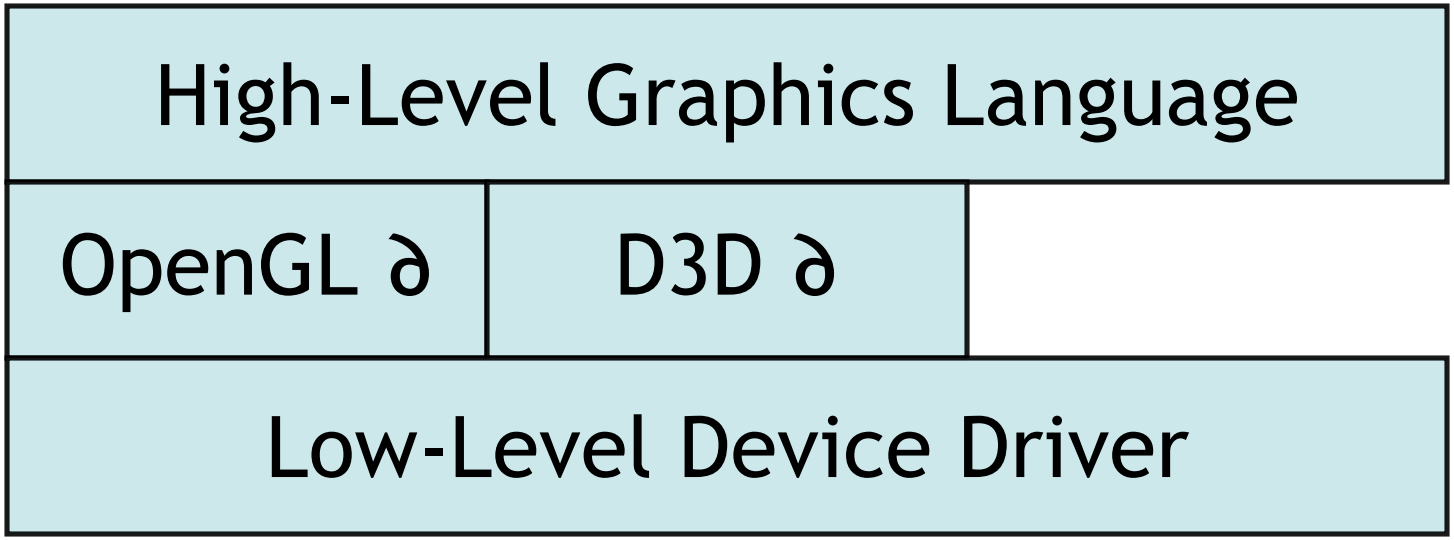


Towards Programmable Graphics

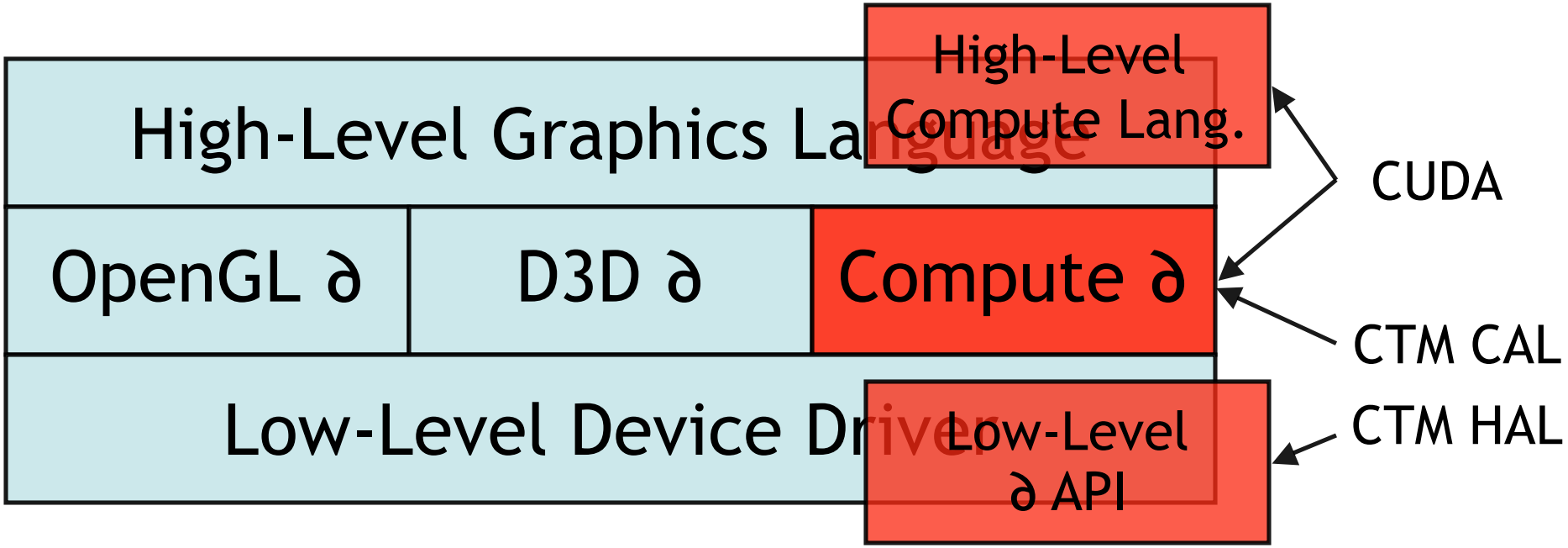
- **Fixed function**
  - Configurable, but not programmable
- **Programmable shading**
  - Shader-centric
  - Programmable shaders, but fixed pipeline
- **Programmable graphics**
  - Customize the pipeline
  - Neoptica asserts the major obstacle is programming models and tools

http://www.neoptica.com/NeopticaWhitepaper.pdf
http://www.graphicshardware.org/previous/www_2006/presentations/pharr-keynote-gh06.pdf

Yesterday's Vendor Support



Today's New Vendor Support



Architecture Summary

- **GPU is a massively parallel architecture**
  - Many problems map well to GPU-style computing
  - GPUs have large amount of arithmetic capability
  - Increasing amount of programmability in the pipeline
- **New features map well to GPGPU**
  - Unified shaders
  - Direct access to compute units in new APIs
- **Challenge:**
  - How do we make the best use of GPU hardware?
    - Techniques, programming models, languages, evaluation tools …