GPU Computing with CUDA

Part 1: GPU Computing Evolution

Dortmund, June 4, 2009
SFB 708, AK "Modellierung und Simulation"

Dominik Göddeke
Angewandte Mathematik und Numerik
TU Dortmund

dominik.goeddeke@math.tu-dortmund.de // http://www.mathematik.tu-dortmund.de/~goeddeke

Acknowledgements

- Slides based on previous courses by
  - Mark Harris, Simon Green, Gregory Ruetsch (NVIDIA)
  - Robert Strzodka (MPI Informatik)
  - Dominik Göddeke (TU Dortmund)

  - ARCS 2008 GPGPU and CUDA Tutorials
    http://www.mathematik.tu-dortmund.de/~goeddeke/arcs2008/
  - University of New South Wales Workshop on GPU Computing with CUDA
    http://www.cse.unsw.edu.au/~pls/cuda-workshop09/

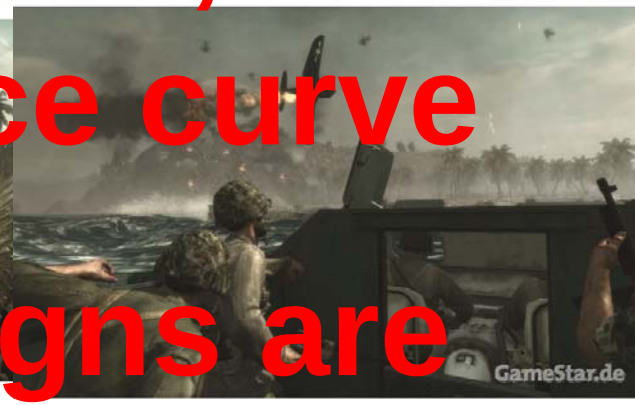
Outline

- Why GPUs need to be fast
- The graphics pipeline
- Evolution towards programmability
- The first wave of GPGPU: 2003-2006
- Example architecture: GeForce 6800 Ultra (2004)
- Consolidation with DirectX 10?

Why GPUs need to be fast

- Huge FLOP requirements
  - Full HD 1080p screen resolution: 2 million pixels
  - 50-100 Hz refresh rate
  - 10-100 operations per vertex
  - 100-1000 operations per fragment (proto-pixel)
- Huge bandwidth requirements
  - Texturing from multiple texture maps
  - Bilinear and anisotropic filtering
  - Full-screen antialiasing
- Performance demands insatiable

Gaming market volume is billions of $$$



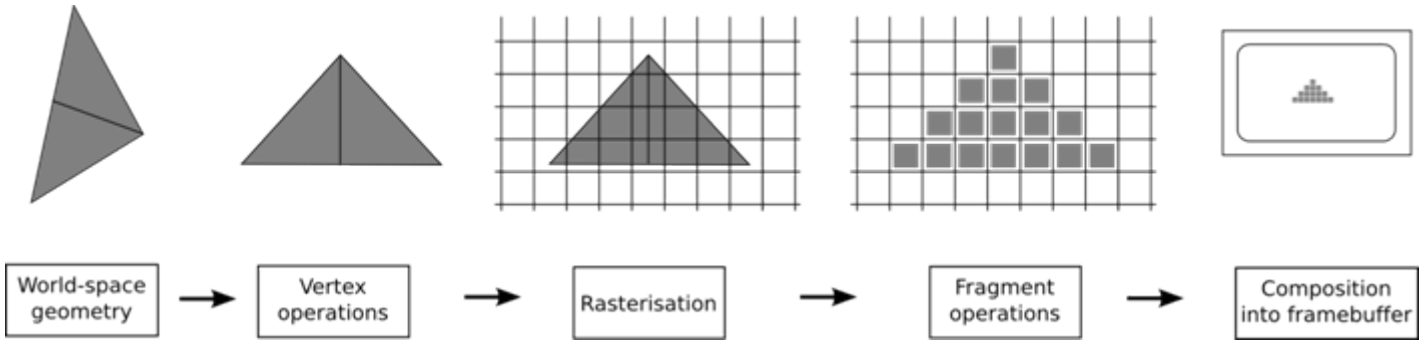
We, the compute folks, can ride the performance curve

CPU and GPU designs are converging (parallelism)

Graphics workloads

- Inherently massively parallel
  - Task- and data-parallelism
  - Vertices in parallel
  - Pixels (fragments) in parallel
  - All simultaneously
- 6 orders of magnitude gap
  - Human visual system: millisecond scale
  - GHz processor: nanosecond scale
- Latency of individual operations not (so) important
  - Throughput is maximised
- GPU hardware is designed around these observations

The graphics pipeline



- Common abstraction of graphics workloads and hardware since 1992
- Exploits parallelism on all scales
- Maximizes throughput over latency

Graphics APIs

- Expose pipeline, not hardware
  - Configure pipeline stages
- OpenGL
  - Extension model for rapid adoption of new features
  - Vendor- and platform independent
- DirectX
  - Windows only
  - Rigorous mandatory feature set for 3-year+ product cycles
  - DirectX version used to identify GPU generations

Evolution towards programmability

- 1999: NVIDIA GeForce 256
  - Term GPU was coined during the launch
  - First to implement the entire pipeline in hardware
  - Limited „programmability“ (hardware T&L)
  - Register combiners for fragment stage

- GPGPU:
  - Very preliminary work by very few enthusiasts
  - „Hacking the GPU“

Evolution towards programmability

- Late 2000: DirectX 8
  - GeForce 3 and Radeon 8500
  - Assembly language for vertex and fragment stage
  - Programmable shading
  - Very limited functionality
    - Instruction count
    - Precision
    - ...
- GPGPU:
  - Slow momentum build-up
  - First PDE solvers mostly for physically-based effects, visually-accurate
  - Cumbersome programming through graphics APIs
  - Hacky, cumbersome and error-prone

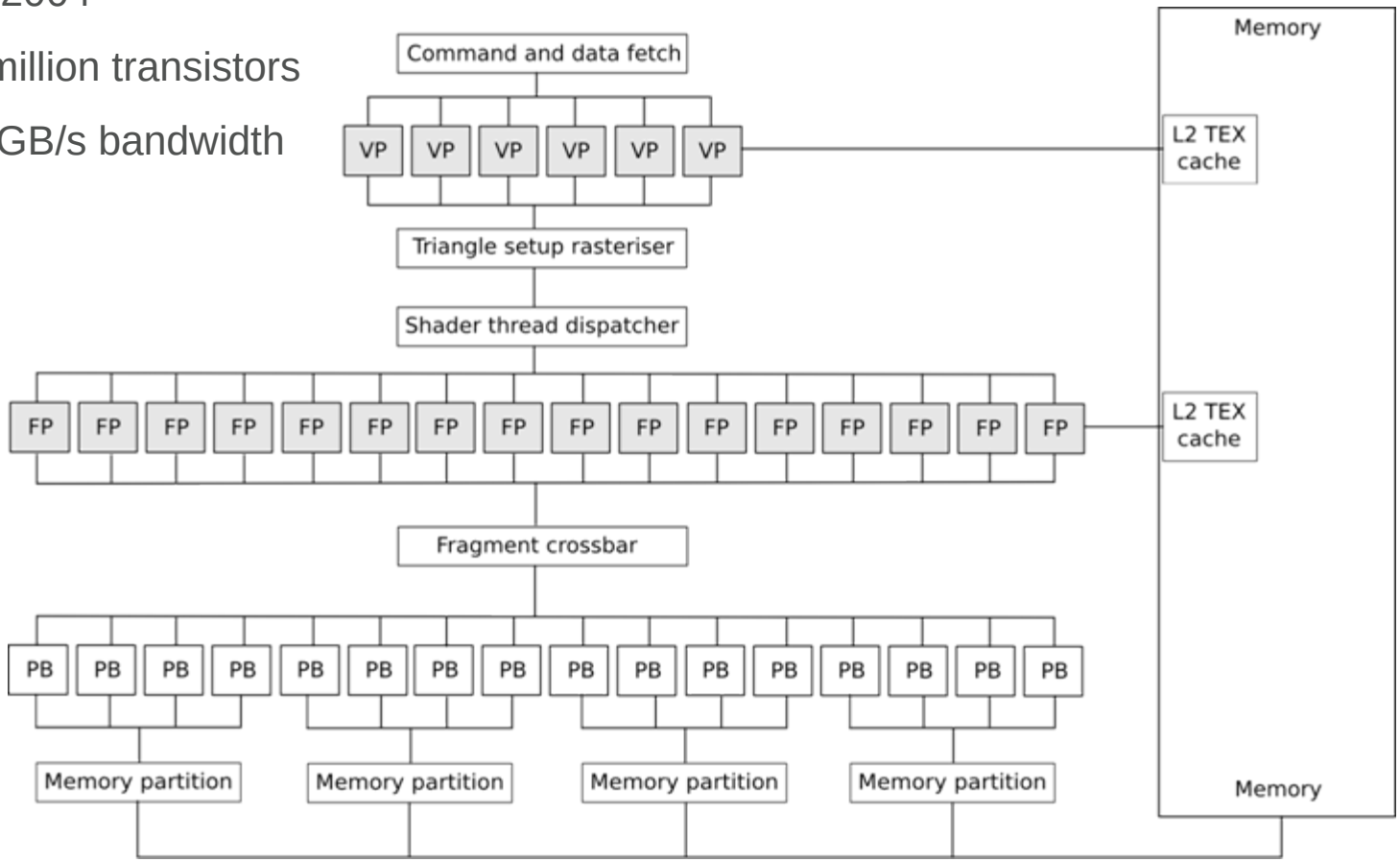
Evolution towards programmability

- DirectX 9: The first wave of GPU computing 2003-2006
  - Floating point support, proper render-to-texture
  - Performance improvements faster than Moore‘s Law

- GPGPU:
  - Growing community (http://gpgpu.org)
  - Cumbersome programming through graphics APIs
  - But: Early high level languages like BrookGPU and Sh
  - Growing # of conference tutorials and workshops
  - Many groundbreaking papers and algorithms published

GeForce 6800 Ultra

- April 2004
- 222 million transistors
- 35.6 GB/s bandwidth



Early GPGPU summary

- Main limitations
  - Programming cumbersome and error-prone (graphics APIs)
  - No scatter support (concurrent read exclusive write model)
  - Loadbalancing between stages difficult (often only fragment stage used at all)

- But:
  - Lots of success by early adoptors
  - First commercial ventures
    - RapidMind
    - Acceleware
    - PeakStream (bought by google in 2007)
  - People reluctantly started to take GPGPU seriously
  - Hardware vendors saw market opportunity

- Released 2006/2007
  - Unified architecture
    - Vertex, fragment and (new) geometry stage same virtual machine
    - Hardware loadbalancing
  - Many many hacky features now mandatory via clean interfaces
  - Big leap forward in mandatory graphics features
- Consolidation?
  - Plausible
- But something entirely different happened
  - Unified graphics and compute
  - NVIDIA CUDA and AMD Stream (CAL/HAL/CTM...)
  - GPU computing started to take off