CS 5984
Parallel Computing and Visualization on GPU

Introduction

Yong Cao

Course Goals

- Understand the parallel architecture programming framework of Graphics Processing Units (GPUs)
  - Parallelize existing sequential algorithms
  - Design GPU-friendly parallel algorithms
- Learn 3D visualization algorithms
  - Ray tracing algorithm
  - Acceleration techniques, global illumination, and more …
- Bring them together: Real-time visualization on GPU
  - Parallelize visualization/graphics algorithms
  - General purpose computing + visualization On GPUs

About me

- Prof. Yong Cao
  - Office hour: By appointment at KWII 1127 or MCB 638
  - Email: yongcao@vt.edu (Please use CS5984 in your subject line)
  - Phone: 540-231-0415
  - Website: www.cs.vt.edu/~yongcao

Course Website

- http://people.cs.vt.edu/~yongcao/teaching/csx984/fall2010/index.html
- Or go to my website, and click on the course link.

- Five sections:
  - Home page
  - Syllabus/Schedule
  - Notes
  - Projects
  - Resources

Course Materials

- Textbooks: (Not required)
  - *Programming Massively Parallel Processors: A Hands-on Approach*, David Kirk and Wen-mei Hwu.
    (A downloadable draft is located at
    http://courses.engr.illinois.edu/ece498/al/syllabus.html)
  - *Physically Based Rendering: From Theory to Implementation (Second Edition)*, Matt Pharr and Greg Humphreys.

Course Materials

- Other Web Resources:
  - NVIDIA CUDA Programming Guide. NVIDA CUDA website, http://www.nvidia.com/object/cuda_home.html.
  - UIUC Parallel Programming Course Website: http://courses.ece.uiuc.edu/ece498/al/.
  - Astro GPU Workshop Videos: http://www.astrogpu.org/videos.php
  - SC07 GPU Tutorials on GPGPU.org Website: http://www.gpgpu.org/sc2007/
  - Other Courses on GPGPU at NIVIDA Website: http://www.nvidia.com/object/cuda_university_courses.html
  - GPU Computing Course at SIGGRAPH Asia: http://sa08.idav.ucdavis.edu/

Course Work

- Participation, Quizzes 15%
  - You are expected to participate in class discussion.
  - 3-4 pop quizzes before class start.
- Programming Assignments 25%
  - Assignment 1: Min, Max, Median
  - Assignment 2: Basic Ray Tracer on GPU
  - Assignment 3: Ray Tracer with acceleration structure
- Paper Presentation 25%
  - Each student will present two papers in class. (One paper per class)
- Project Presentation & Report 35%
  - Prefer visualization applications, but **can be any topic**.

Academic Honesty

- You are allowed and encouraged to discuss assignments with other students in the class. Getting verbal advice/help from people who've already taken the course is also fine.
- Any reference to assignments from previous terms or web postings is unacceptable
- Any copying of non-trivial code is unacceptable
  - Non-trivial = more than a line or so
  - Includes reading someone else's code and then going off to write your own.

Late Assignment Policy

- Assignments will be downgraded 25% for each day late. No exception permitted.

Final Project

- Can be any topic.
- One person team only!
- Demo and presentation are required.
- Final report is required. (4-6 pages)
  - Please see class website for the detail.

What Are Computer Graphics?

- Pictures generated by a computer

What Are Computer Graphics?

- Pictures generated by a computer

What Are Computer Graphics?

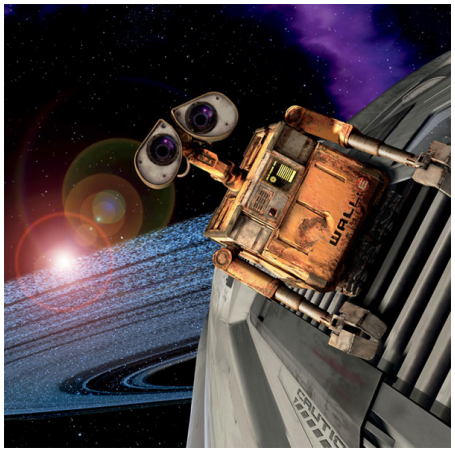
- Pictures generated by a computer

What Are Computer Graphics?

- Pictures generated by a computer

Applications

- Entertainment
- Computer-aided design
- Scientific visualization
- Training
- Education
- E-commerce
- Computer art



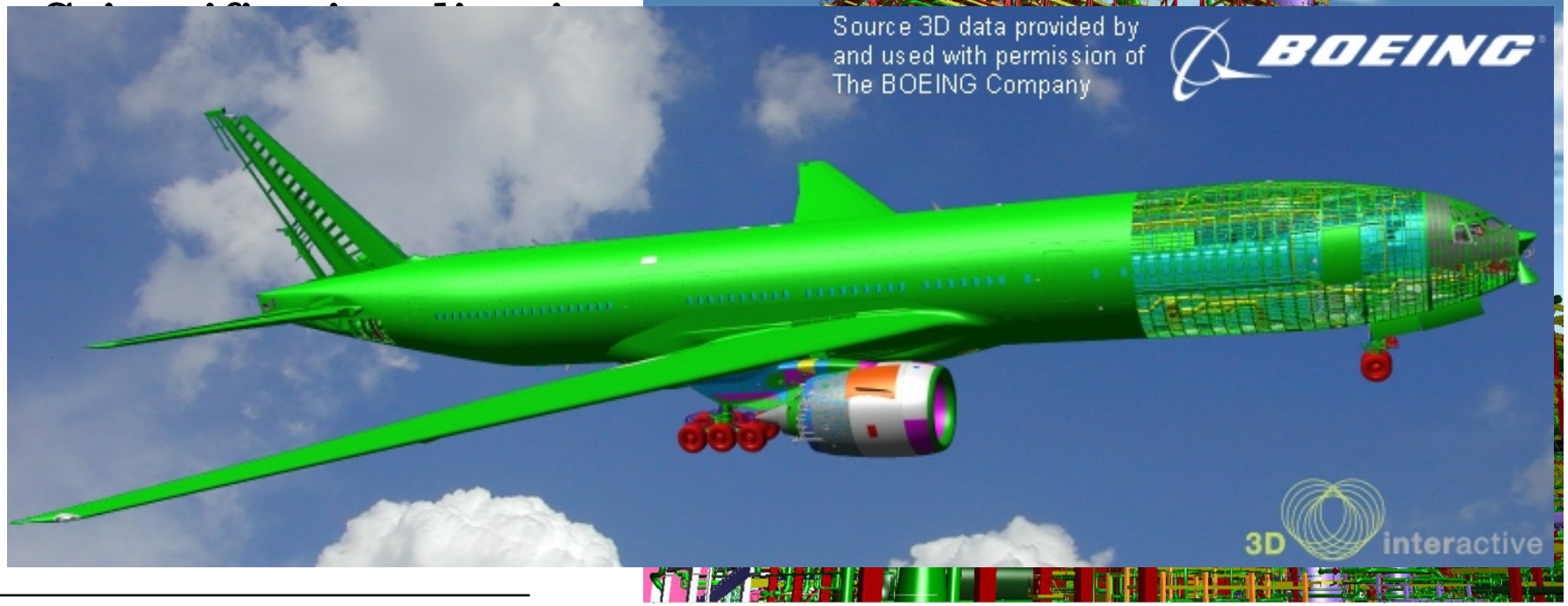
Pixar Inc.

Fox.

EA

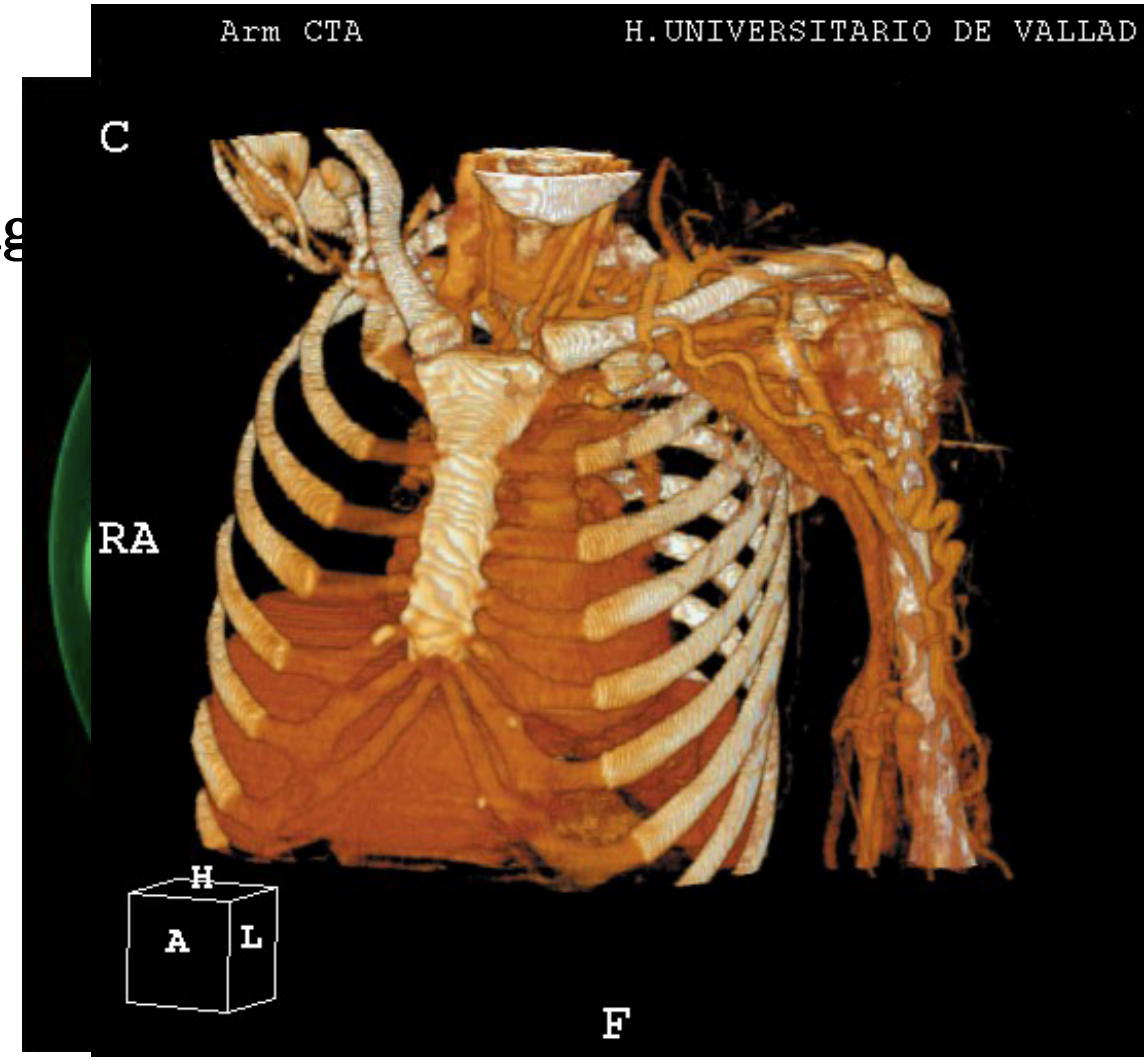
Applications

- Entertainment
- Computer-aided design



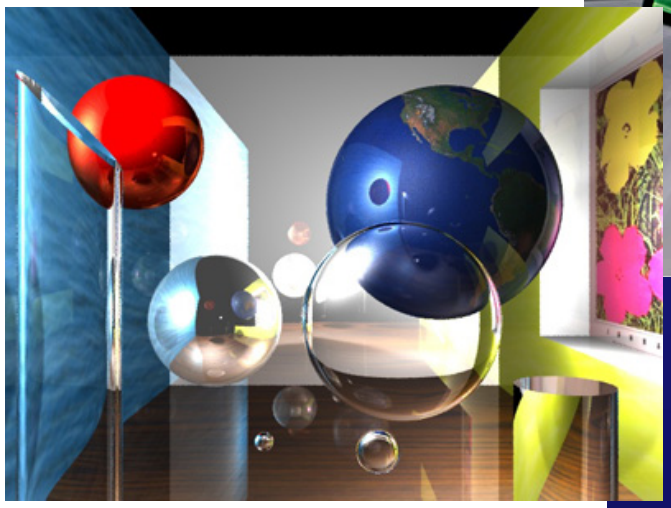
Applications

- Entertainment
- Computer-aided desig
- Scientific visualization
- Training
- Education
- E-commerce
- Computer art



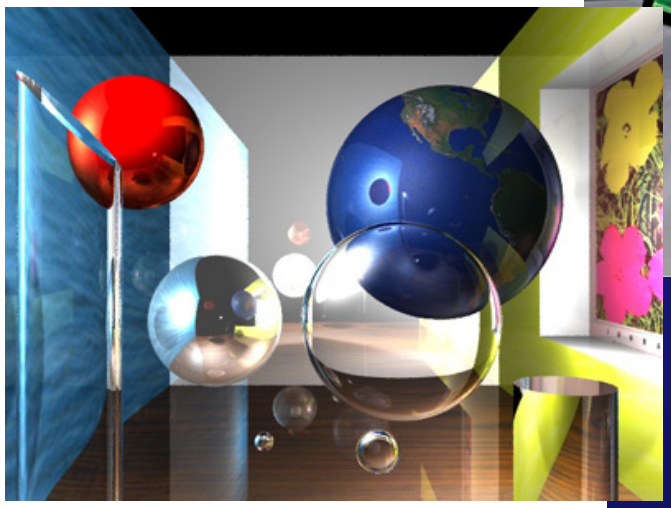
This Course: Rendering

- Simulating light propagation
  - Reflection
  - Absorption
  - Scattering
  - Emission
  - Interference



This Course: Rendering in Real-time

- Simulating light propagation
  - Reflection
  - Absorption
  - Scattering
  - Emission
  - Interference

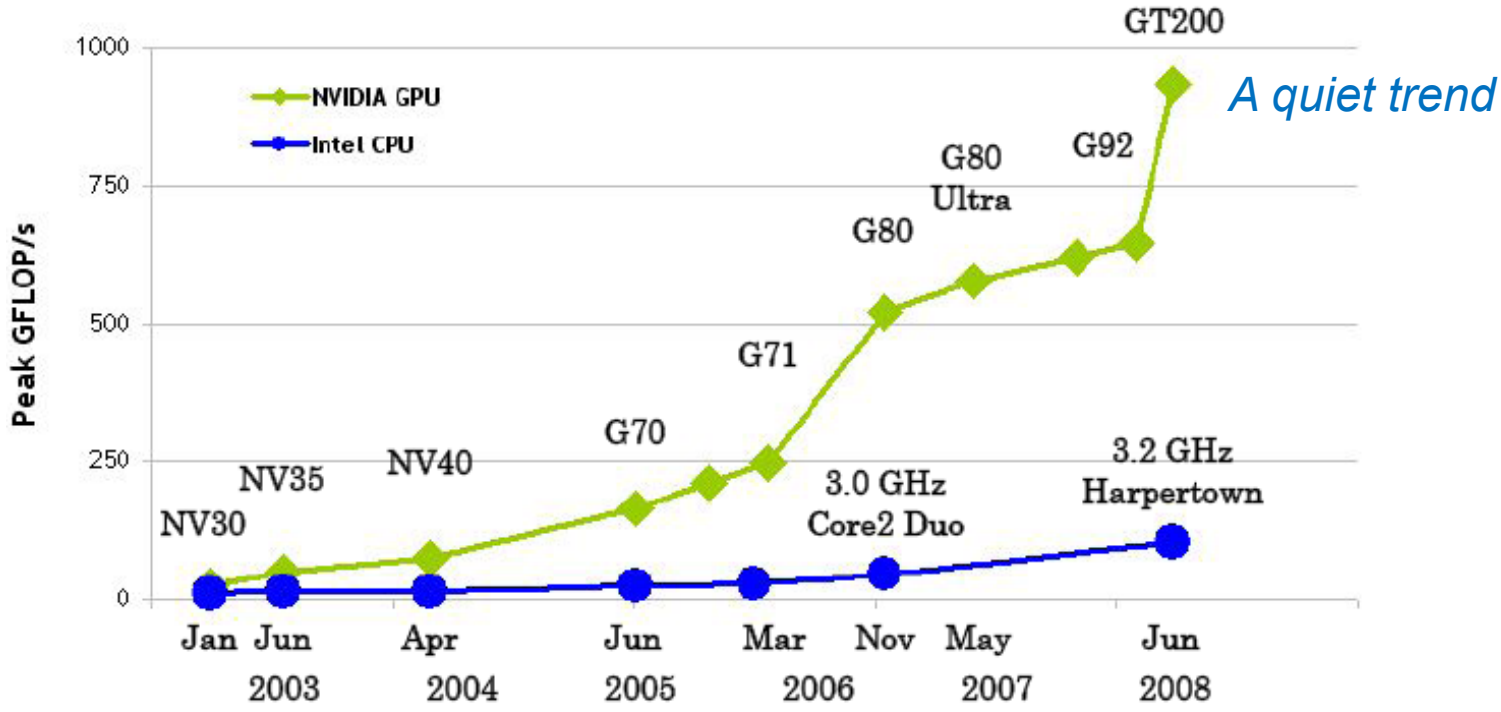


This Course: Rendering in Real-time

- Challenge: Computation!

Why Graphics Card?

- It's powerful!



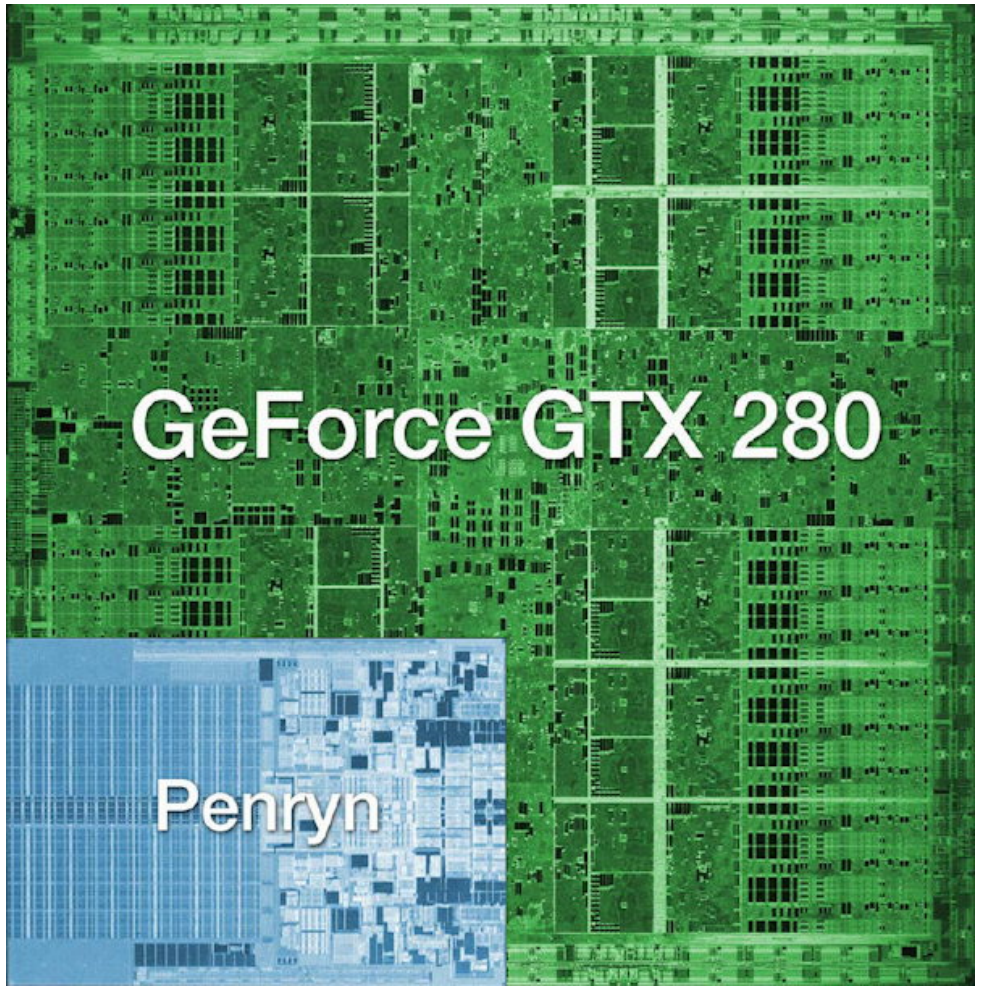
GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

Why Graphics Card?

- It's powerful!
  - NVIDIA GTX 480
    - 480 Cores
    - Bandwidth 177GB/sec
    - 1345 GFLOPs (Close to top 1 super computing 10 years ago)

Why Graphics Card?

- It's powerful!
  - Die size 576 mm^2
  - 1.4 Billion Transistors
  - Memory width: 512bit
  - Bandwidth 142GB/s



Max board Power:236 W
GTX285: 183 W (55nm)

Why Graphics Card?

- It's cheap and everywhere.
  - E.g. NIVIDA sold 200 Million high-end GPGPU devices.
  - A 128-core Geforce 9800GTX (648 GFLOPs) is $129 on Newegg.com.

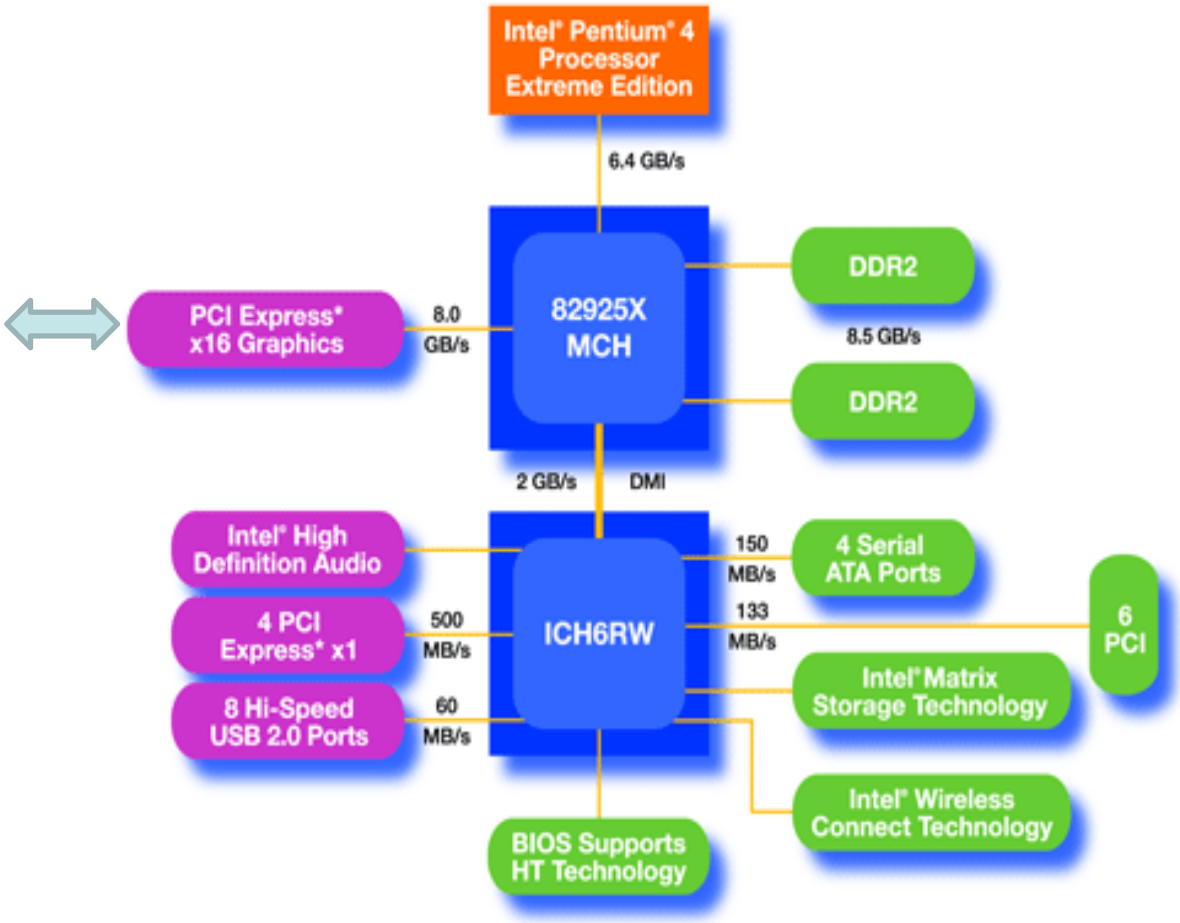
Why Graphics Card NOW?

- Before:
  - Two years ago, everyone is using Graphics API (Cg, GLSL, HLSL) for GPGPU programming.
  - Restrict random-read (using Texture), NOT be able to random-write. (No pointer!)

Why Graphics Card NOW?

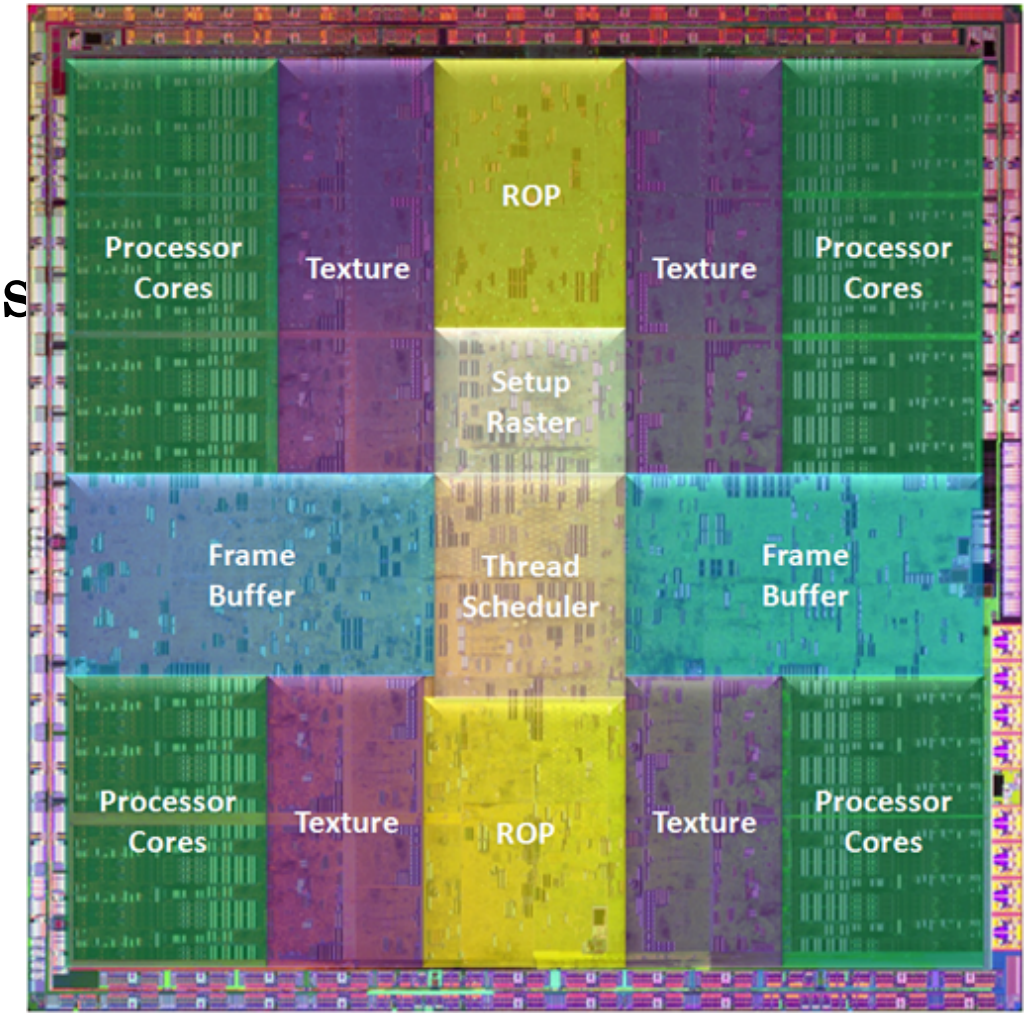
- Now:
  - NIVIDA released CUDA 4 years ago, since then
    - Tens of Thousands of CUDA software engineers
    - New job title "CUDA programmer"
    - 904 publications with "CUDA" in their title! (Google Scholar)
    - 2770 publications with "GPU" in their title since 2006.
  - Why?
    - Standard C language
    - Support Pointer! Random read and write on GPU memory.
    - Work with C++, Fortran

Where's GPU in the system



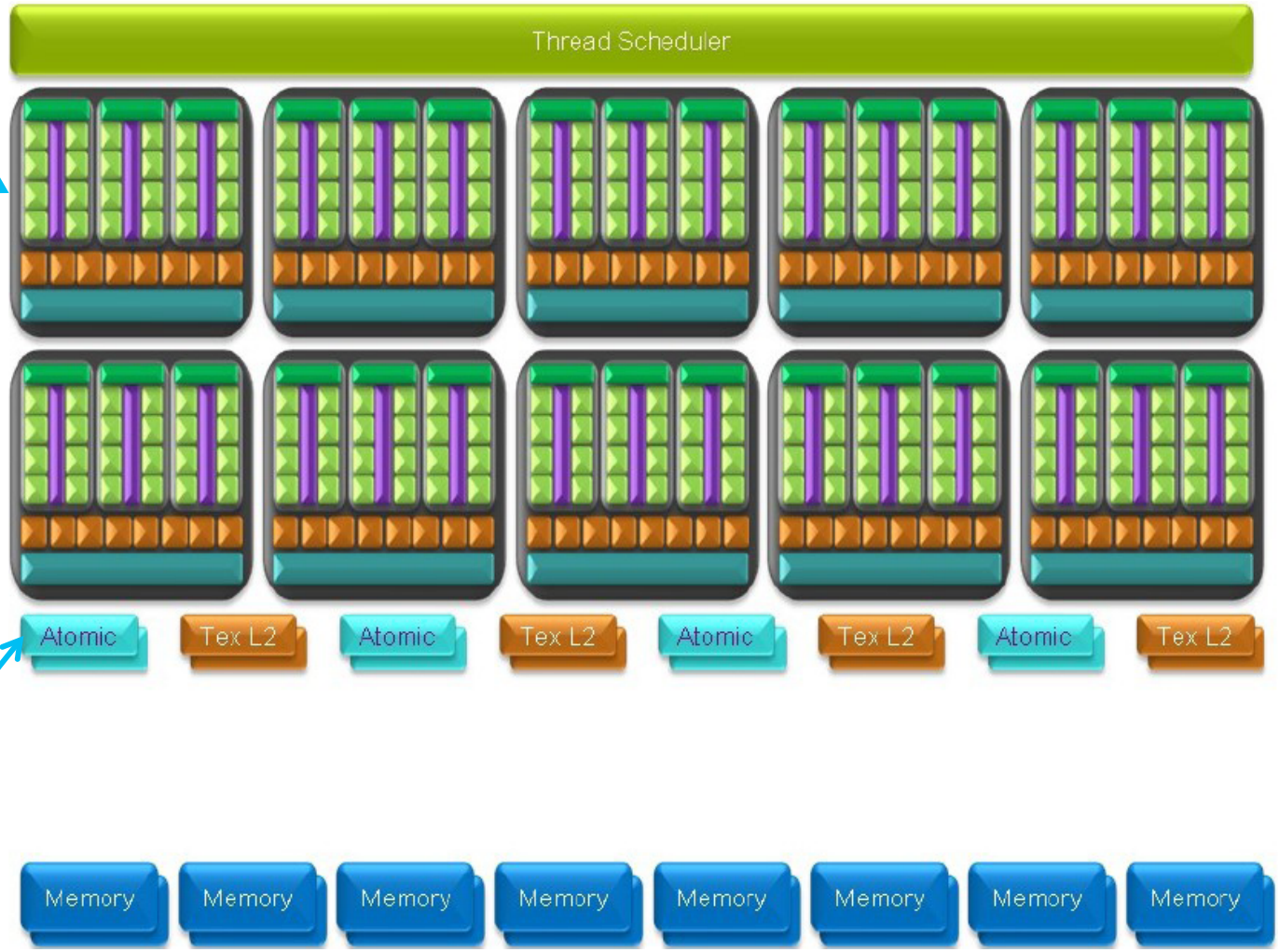
NVIDIA GPU Architecture

- Lots of ALUs
- Lots of Control
- Focus on Graphics Applicants (Data parallel)



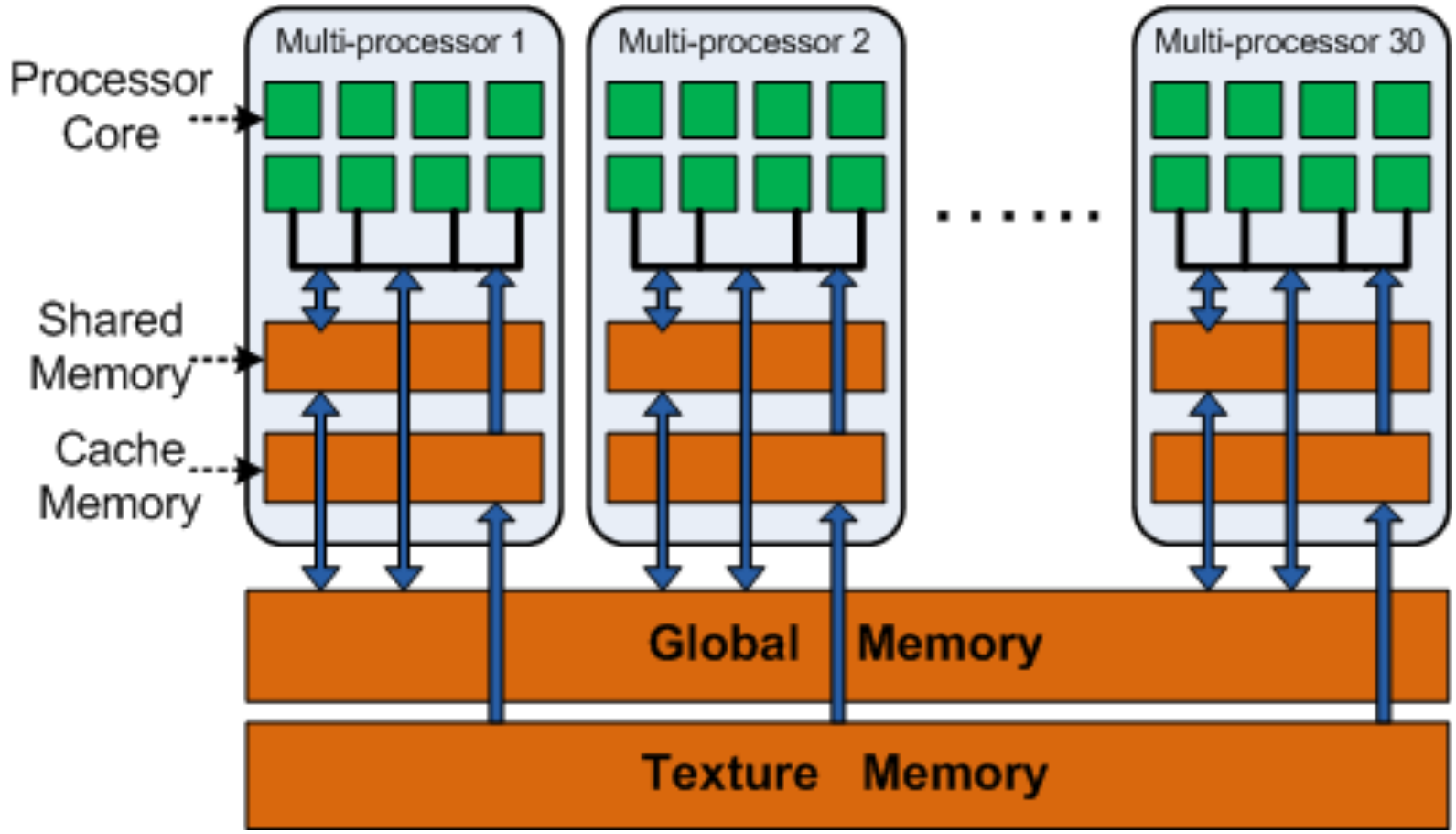
NVIDIA GT200 Architecture

Thread Processing Cluster

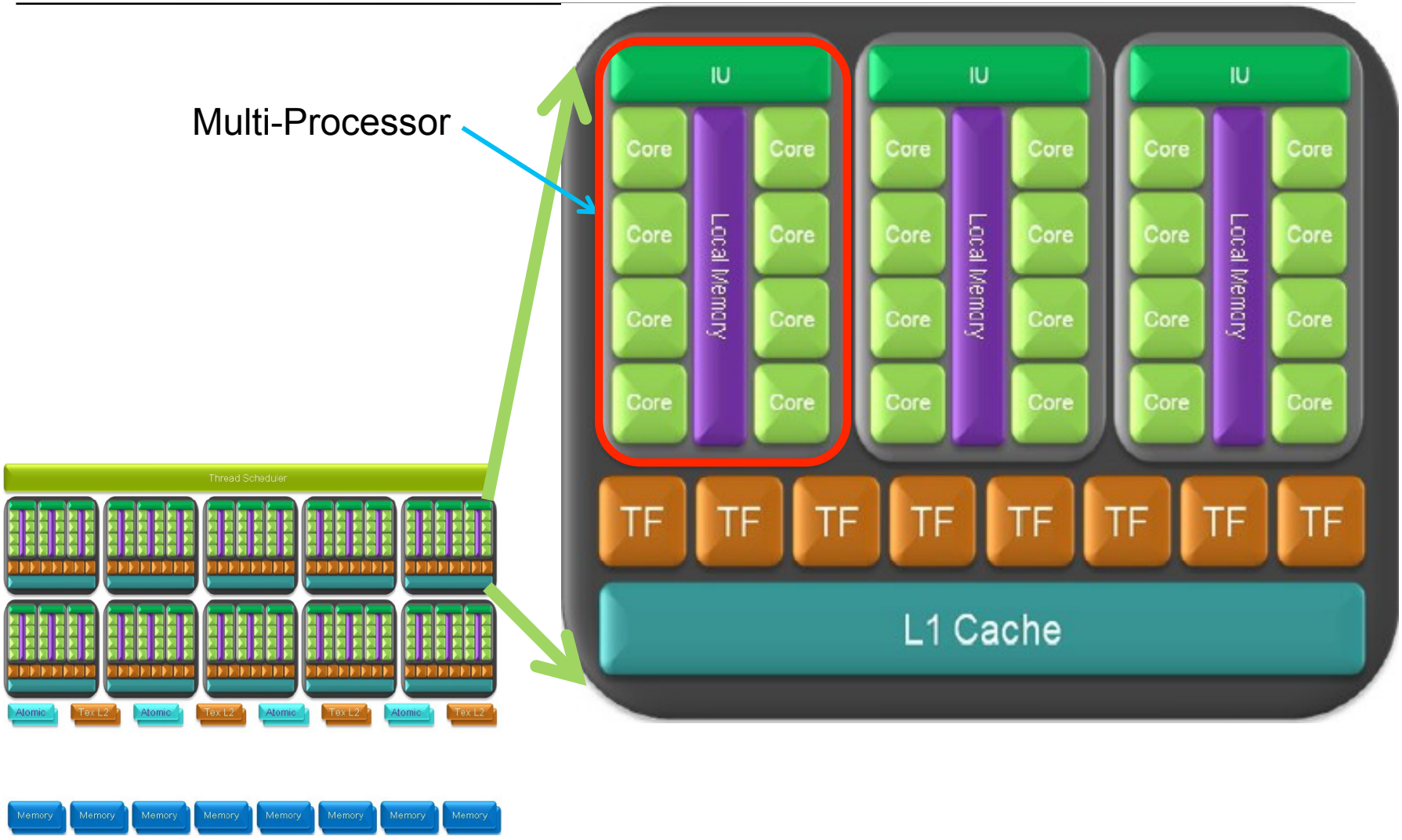


Atomic Memory Access Control

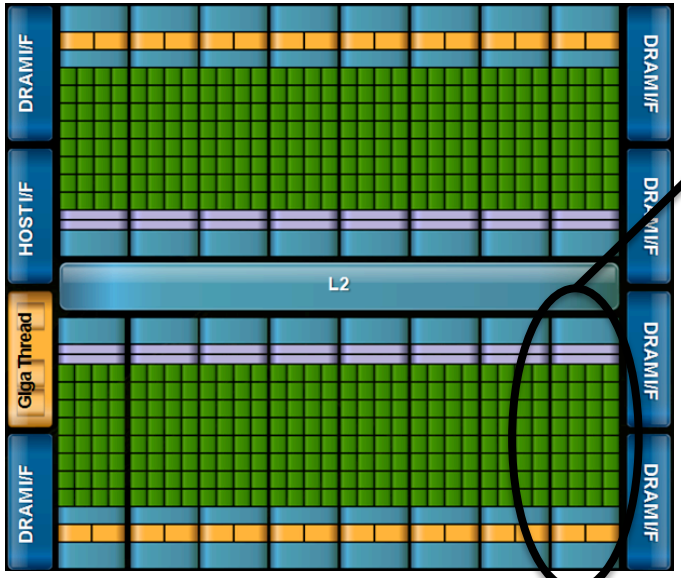
NVIDIA GT200 Architecture



NVIDIA GT200 Architecture

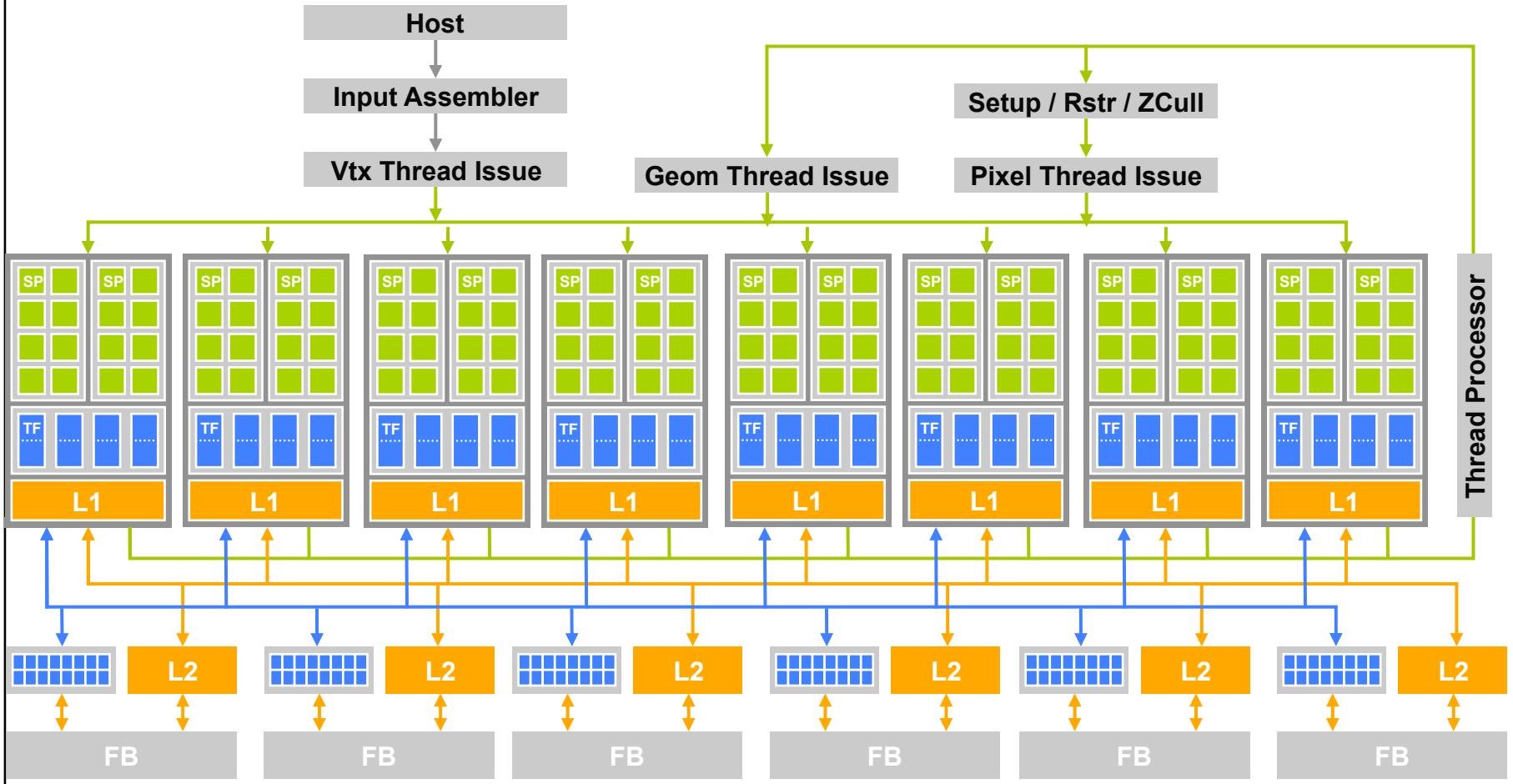


Fermi (2010)



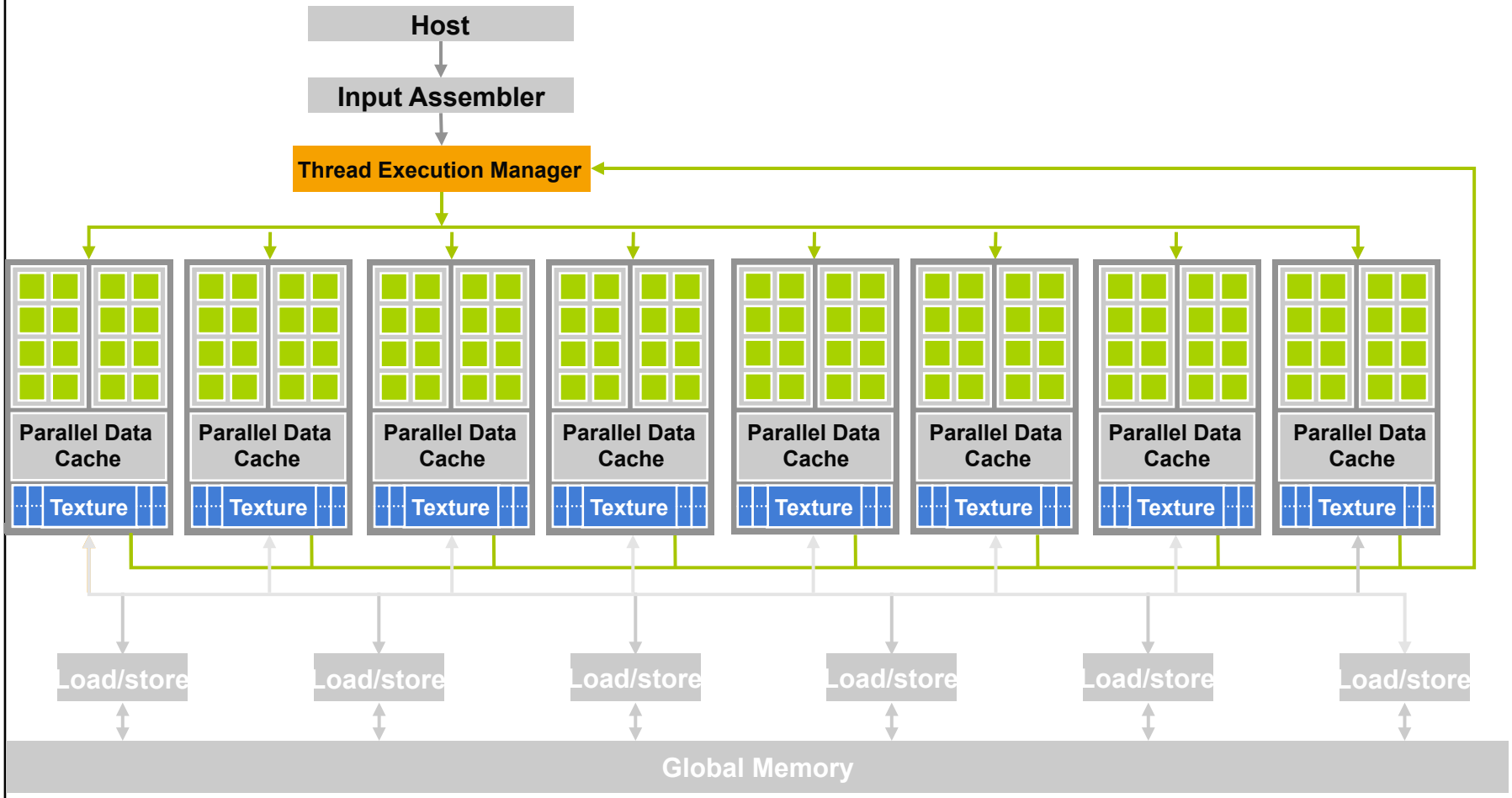
~1.5TFLOPS (SP)/~800GFLOPS (DP)
230 GB/s DRAM Bandwidth

Execution Mode– Graphics



NVIDIA G80 GPU

Execution Mode– General Computing



NVIDIA G80 GPU

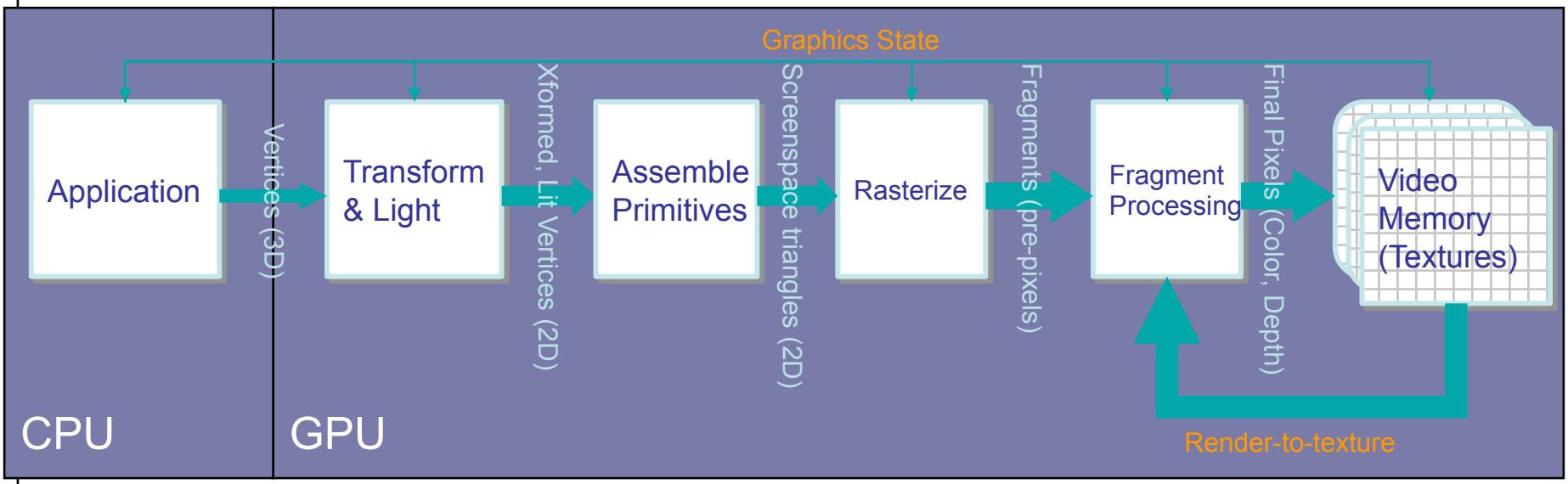
GPGPU from Graphics Point of View

- Graphics Processing Pipeline (on GPU)
  - Fixed function pipeline
  - Programmable pipeline with Shaders
- GPU Processing Model
  - Stream computing model

3D Graphics Applications

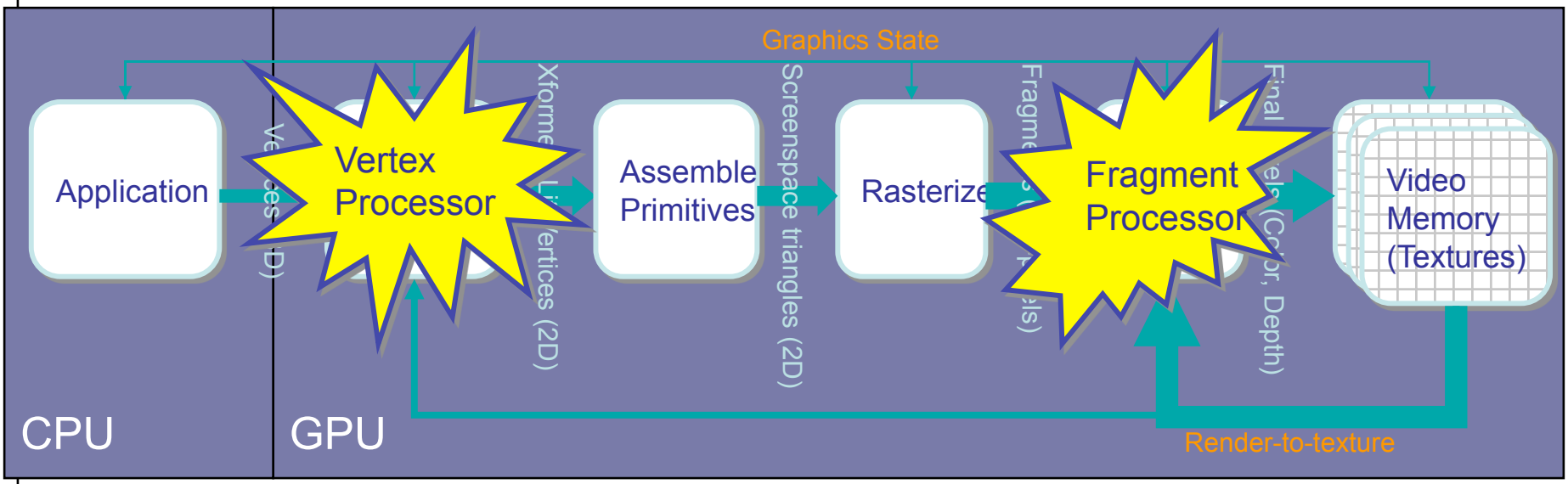
- Demos.

GPU Fundamentals: The Graphics Pipeline



- A simplified graphics pipeline

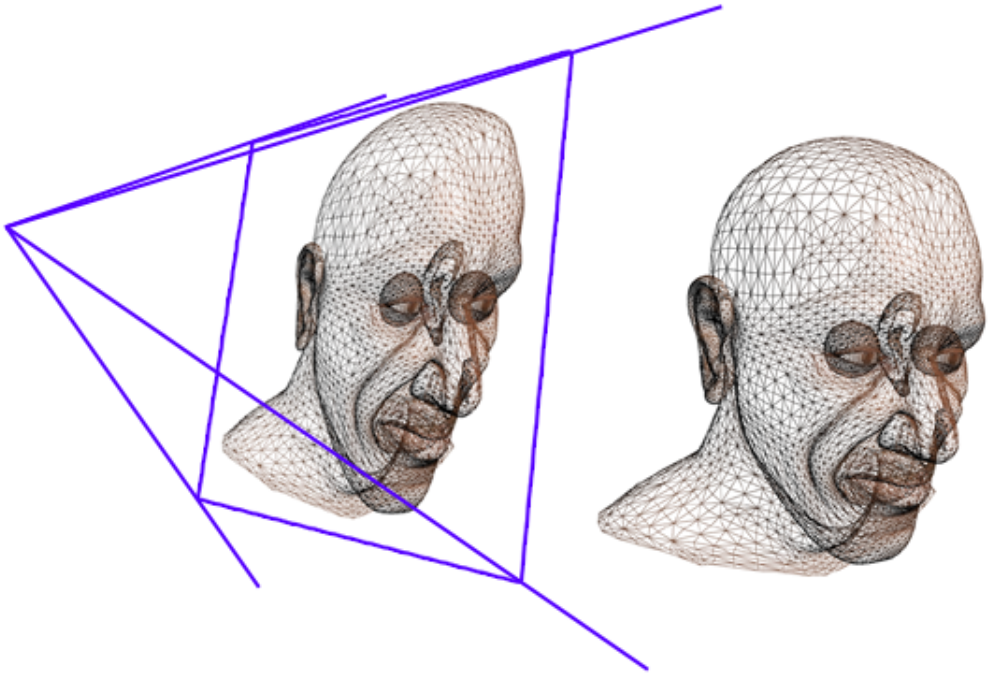
Programmable Graphics Pipeline - Shaders



- Programmable vertex processor!
- **Programmable fragment processor!**

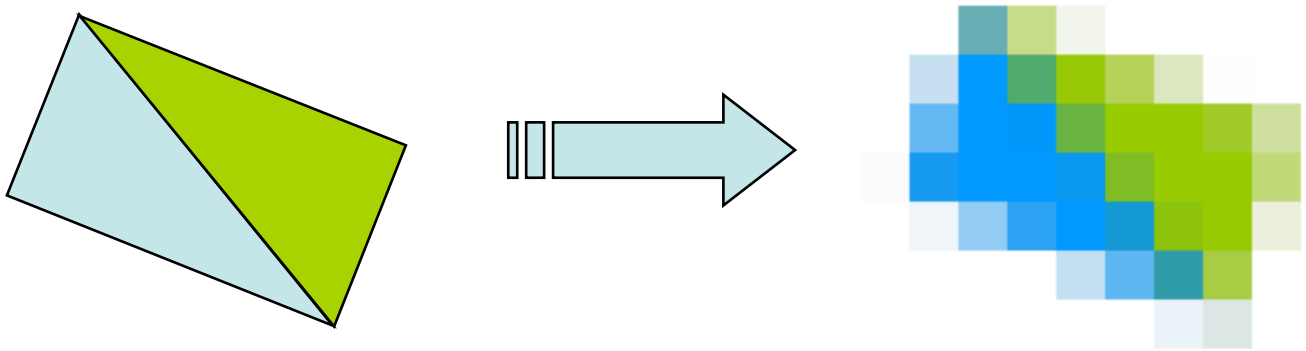
GPU Pipeline: Transform

- Vertex processor (multiple in parallel)
  - Transform from "world space" to "image space"
  - Compute per-vertex lighting



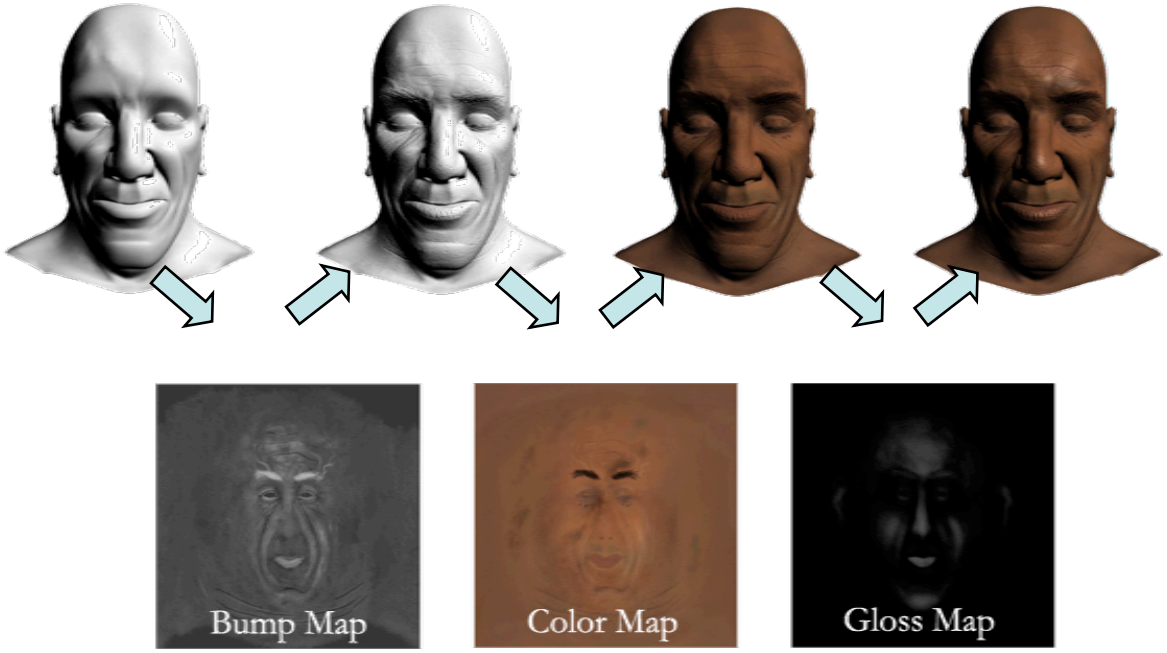
GPU Pipeline: Rasterize

- Rasterizer
  - Convert geometric rep. (vertex) to image rep. (fragment)
    - Fragment = image fragment
      - Pixel + associated data: color, depth, stencil, etc.
  - Interpolate per-vertex quantities across pixels



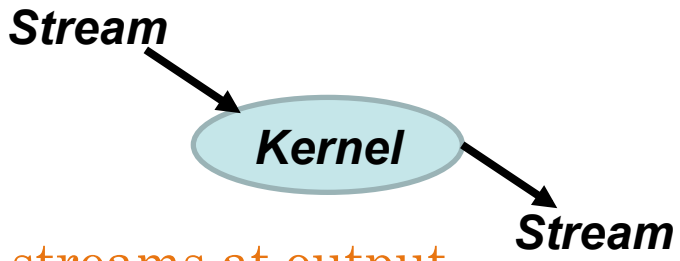
Pixel / Fragment Processor

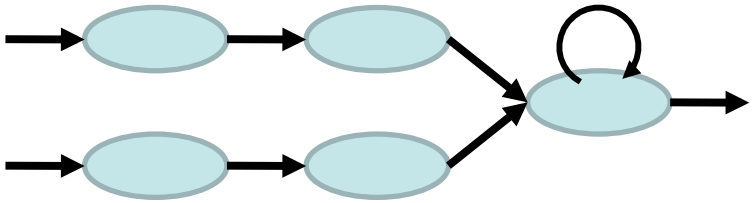
- Fragment processors (multiple in parallel)
  - Compute a color for each pixel
  - Optionally read colors from textures (images)



GPU Programming Model: Stream

- Stream Programming Model
- Streams:
  - An array of data units
- Kernels:
  - Take streams as input, produce streams at output
  - Perform computation on streams
  - Kernels can be linked together





Why Streams?

- Ample computation by exposing parallelism
  - Stream expose data parallelism
    - Multiple stream elements can be processed in parallel
  - Pipeline (task) parallelism
    - Multiple tasks can be processed in parallel
- Efficient communication
  - Producer-consumer locality
  - Predictable memory access pattern
    - Optimize for throughput of all elements, not latency of one
    - Processing many elements at once allows latency hiding

Reading Material

- NVIDIA CUDA Programming Guide, Chapter One
  http://www.nvidia.com/object/cuda_develop.html
  and looking for documentation
- NVIDIA Geforce GTX 280 Technical Brief
  http://www.nvidia.com/docs/IO/55506/GeForce_GTX_200_GPU_Technical_Brief.pdf