Introduction and Overview

Aaron Lefohn
University of California, Davis
GPGPU
(General-Purpose computation on Graphics Processing Units)
is about doing this...

Molecular Dynamics (Buck)
...and this...

Cloud Simulation (Harris)
...and this...

Flow Visualization (Krueger)
...and this..

Motion estimation computer vision (Strzodka)
...more than 10x faster than a on a CPU.
GPGPU is about...

Parallel desktop computing
Revolution
Old software model
1) You write serial software
2) It runs 2x faster every 18 months
(thanks Intel/AMD/IBM/Motorola!)
But...this model ended in 2003;
But...this model ended in 2003;

CPU Clock speeds stopped increasing.
“The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”
Herb Sutter, Dr. Dob’s Journal, 30(3), March 2005
“The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”
Herb Sutter, Dr. Dob’s Journal, 30(3), March 2005
Parallel desktop computing Revolution

New model
1) You rewrite your software to be highly parallel
2) It runs 2x faster when Intel/AMD double the number of cores
The parallel desktop computing revolution
The parallel desktop computing revolution

Faster software requires increased parallelism
But...

Revolutions give opportunity for new ideas
“Instigating a platform tug of war: Graphics vendors hunger for CPU suppliers' turf”

Brian Dipert
Senior Technical Editor
EDN, 10/13/2005
GPGPU

(General-Purpose programming on Graphics Processing Units)
What is GPGPU?
What is GPGPU?

Use GPU as alternate parallel desktop computing platform
Why GPGPU?
Why GPGPU?

1) GPUs are highly parallel
Why GPGPU?

1) GPUs are highly parallel

Parallelism is future

(remember parallel revolution)
Why GPGPU?
1) GPUs are highly parallel

CPUs have 2 processors (dual-core)
Why GPGPU?
1) GPUs are highly parallel
IBM Cell has 9 processors
Why GPGPU?

1) GPUs are highly parallel

GPUs have > 24 processors

(“24 fragment shading pipes”)
Why GPGPU?

2) GPUs are highly threaded
Why GPGPU?

2) GPUs are highly threaded

Threads keep processors busy
(even though memory is slow)
Memory is slow...even on GPUs

Based on data from http://graphics.stanford.edu/projects/gpubench/results/
Why GPGPU?

2) GPUs are highly threaded

CPUs support

1-2 hardware threads
Why GPGPU?

2) GPUs are highly threaded

GPUs support

100s of hardware threads
Why GPGPU?

2) GPUs are highly threaded

GPU threading good at hiding memory access cost
Why GPGPU?

3) GPUs are ubiquitous and cheap
Why GPGPU?

3) GPUs are ubiquitous and cheap

GPU in every laptop, desktop (and PDA, cellphone)
Why GPGPU?

3) GPUs are ubiquitous and cheap

Driven by $$$ from games
Why GPGPU now?
Why GPGPU now?

Feature set gaining maturity
Why GPGPU now?
Feature set gaining maturity
Programmability
2001
Why GPGPU now?
Feature set gaining maturity
Floating point
2002
Why GPGPU now?
Feature set gaining maturity
Conditional execution
2004, 2005, ...
GPGPU History

GPGPU started before
desktop parallel revolution
GPGPU History

GPGPU gaining momentum quickly due to parallel desktop revolution
GPGPU History
(see www.gpgpu.org)

1990
Lengyel
Motion planning
GPGPU History

1999

Hoff

Voronoi diagrams

GPGPU History

2000

Peercy

Renderman with OpenGL

Image from “Interactive Multi-pass Programmable Shading,” Peercy et al., ACM SIGGRAPH 2001
GPGPU History

2001

Strzodka

2D PDE image processing

Image from “Level Set Segmentation in Graphics Hardware,” Rumpf et al., ICIP 2001
GPGPU History

2002

Purcell / Carr

Ray tracing

Image from “Ray Tracing on Programmable Graphics Hardware,” Purcell et al., ACM SIGGRAPH 2002
GPGPU History

2002

Harris

Cellular automata

Image from “Physically-Based Visual Simulation on Graphics Hardware,” Harris et al., Graphics Hardware 2002
GPGPU History

2003

Krueger / Boltz / Goodnight

Linear algebra

Image from “Linear Algebra Operators for GPU Implementation of Numerical Algorithms,”
Kruger et al, ACM SIGGRAPH 2003
GPGPU History

2003

Lefohn

3D level-set solver

Image from “Interactive Deformation and Visualization of level-Set Surfaces using Graphics Hardware,” Lefohn et al., IEEE Visualization 2003
GPGPU History

2004

Govindaraju

Database operations

kernel void saxpy (float a, float4 x<>, float4 y<>,
                out float4 result<>)
{
    result = a*x + y;
}

void main (void)
{
    float a;
    float4 X[100], Y[100], Result[100];
    float4 x<100>, y<100>, result<100>;
    ... initialize a, X, Y ...
    streamRead(x, X);   // copy data from mem to stream
    streamRead(y, Y);
    saxpy(a, x, y, result);  // execute kernel on all elements
    streamWrite(result, Result);  // copy data from stream to mem
}
GPGPU History

2005
Govindaraju
Fast sorting
GPGPU History

2005

Lefohn

Generic data structures

GPGPU History

2005

Kondratieva

Particle Advection and Tensor Vis

Image from “The Application of GPU Particle Tracing to Diffusion Tensor Field Visualization,” Kondratieva et al., IEEE Visualization 2005
GPGPU History

2005

Horn

Hidden Markov DNA matching

Image from “ClawHMMER: A Streaming HMMer-Search Implementation,”
Horn et al., Supercomputing 2005
GPGPU
Where are we now?
Where are we now?

“*What can we do?*”

“*What should we do?*”
Where are we now?

Madly hacking

Language Development
Where are we now?

Maturity?

GPGPU Eurographics STAR report
Where are we now?

Graphics programmers
Parallel programming experts
This IEEE Visualization 2005 Course
GPGPU may be the dawn of the parallel desktop computing revolution...
...and the Visualization community has a voracious appetite for computer power...
(i.e., you are perfect early adopters)
...but using GPUs for computation is Raw Painful Non-intuitive The domain of specialists
We want to change this.
Course Goal
“Give visualization community the knowledge and tools to leverage the compute power of GPUs.”
Why GPGPU at IEEE Visualization?
Why GPGPU at IEEE Visualization?

1) You need GFLOPS
Why GPGPU at IEEE Visualization?

2) Image processing perfect for GPGPU
   Segmentation
   Registration
   ...

GPGPU
Why GPGPU at IEEE Visualization?

3) “Visulation?” / “Simulization?”

(Simultaneous simulation and visualization)
Why GPGPU at IEEE Visualization?

4) You already know GPU programming
Course speakers
Ian Buck

Senior Software Architect
NVIDIA Corporation

Ph.D., Pat Hanrahan,
Stanford University
Aaron Lefohn

Ph.D. candidate, John Owens
University of California, Davis

Graphics Software Engineer
Pixar Animation Studios
Patrick McCormick

Visualization Research Scientist
Los Alamos National Lab

Lead of Scout project
Timothy Purcell

Graphics Hardware Architect
NVIDIA Corporation

Ph.D., Pat Hanrahan
Stanford University
John Owens
Assistant Professor
Electrical and Computer Engineering
University of California, Davis

Ph.D., Bill Dally and Pat Hanrahan
Stanford University
Robert Strzodoka

Postdoctoral fellow, Ron Fedkiw
Stanford University

Ph.D., Martin Rumpf
University of Duisburg
Course Schedule
Morning

- Introduction
- GPU/data-parallel architecture overview
- GPGPU programming model and languages
Afternoon

- Computational building blocks
- “Getting your hands dirty: Making it work”
- Case studies
- The future
- Q&A
• Section 1: Introduction
  - 8:30  Introduction and Tutorial Overview  Lefohn  A
  - 9:00  A Data-Parallel Genealogy: The GPU’s family tree  Owens  B

• Section 2: GPGPU Programming
  - 9:30  The Programming Model  Owens  C
  - 10:00  Break
  - 10:30  GPGPU Toolkits and Programming Languages  Buck  D
  - 11:20  Scout GPGPU Programming Language  McCormick  E
  - 11:50  High-Level GPU Data Structures  Lefohn  F
  - 12:15  Lunch

• Section 3: GPGPU Computational Primitives
  - 1:45  Mathematical Primitives  Strzodka  G
  - 2:15  General Algorithmic Primitives  Purcell  H

• Section 4: “Getting Your Hands Dirty”
  - 2:45  GPU Memory Model Overview  Lefohn  I
  - 3:05  Computation Tips and Tricks  Buck  J
  - 3:25  Developer Tools  Purcell  K
  - 3:45  Break

• Section 5: Case Studies
  - 4:15  Ray Tracing and Ray Marching  Purcell  M
  - 4:35  GPGPU Accelerated Visualization with Scout  McCormick  L
  - 4:55  Advanced Image Processing  Strzodka  N

• Section 6: Conclusions
  - 5:30  The Future  Owens  O
  - 5:45  Open Question and Answer  All