Goals

• Learn GPU architecture and programming; both for graphics and for computing (GPGPU)
• Shading languages (GLSL, Cg, HLSL), compute APIs (CUDA, OpenCL, DirectCompute)

Time and location

• Monday + Thursday, 16:00 – 17:30, Building 9, Room 4220

Webpage:

http://faculty.kaust.edu.sa/sites/markushadwiger/Pages/CS380.aspx

Contact

• Markus Hadwiger: markus.hadwiger@kaust.edu.sa
• Peter Rautek (assignments): peter.rautek@kaust.edu.sa
• Ronell Sicat (assignments): ronell.sicat@kaust.edu.sa

Prerequisites

• C/C++ programming (!), basic computer graphics, basic linear algebra
Lecture Structure

Lectures

• Part 1: GPU Basics and Architecture (graphics, compute)
• Part 2: GPUs for Graphics
• Part 3: GPUs for Compute

Some lectures will be on research papers (both seminal and current)

Assignments

• 4 programming assignments
• Weekly reading assignments (required; also some optional)

Quizzes

• 6 quizzes, 30 min each, ~every second Sunday
  (tentative dates: Feb 17, Mar 3, Mar 17, Apr 7, Apr 21, May 5)
• From lectures and (required) reading assignments

Semester project + final presentations, but no mid-term/final exam!

Grading: 40% programming assignments; 30% semester project; 30% quizzes
Resources (1)

Textbooks

- GPUs for Graphics: OpenGL 4.0 Shading Language Cookbook
Resources (1)

Textbooks

- GPUs for Graphics: OpenGL 4.0 Shading Language Cookbook
Long list of links on course webpage:

http://faculty.kaust.edu.sa/sites/markushadwiger/Pages/CS380.aspx

- www.opengl.org
- www.gpgpu.org
- www.nvidia.com/cuda/
- www.khronos.org/registry/cl/
- ...

Very nice resources for examples: GPU Gems books 1-3 (available online)
GPU Computing Gems, Vol. 1 + 2 (Emerald/Jade edition)
Resources (3)

**OpenGL Programming Guide** *(red book)*


Computer graphics and OpenGL

Current edition: 8th
OpenGL 4.3
contains extended chapters on GLSL

7th edition (OpenGL 3.0/3.1)
available in the KAUST library
Resources (4)

**OpenGL Shading Language** (orange book)

Current edition: 3rd
OpenGL 3.1, GLSL 1.4
no geometry shaders

Available in the KAUST library
also electronically
Resources (5)

CUDA by Example: An Introduction to General-Purpose GPU Programming, Jason Sanders, Edward Kandrot

See reference section of KAUST library
Syllabus (1)

GPU Basics and Architecture (~February)

- Introduction
- GPU architecture
- How shader cores work
- GPU shading and GPU compute APIs
  - General concepts and overview
  - Learn syntax details on your own!
    - GLSL book
    - CUDA book
    - Online resources, ...
Syllabus (2)

GPUs for Graphics (~March)
- GPU texturing, filtering
- GPU (texture) memory management
- GPU frame buffers
- Virtual texturing
Syllabus (3)

GPU Computing (~April, May)
  • GPGPU, important parallel programming concepts
  • CUDA memory access
  • Reduction, scan
  • Linear algebra on GPUs
  • Combining graphics and compute
    – Display the results of computations
    – Interactive systems (fluid flow, ...)

Semester project presentations
4 assignments

• Based on C/C++, OpenGL, and CUDA

Organization

1. Explanation during lecture (and Q&A sessions if required)
2. Get framework online (bitbucket+git)
3. Submit solution and report online (bitbucket+git) by submission deadline
4. Personal presentation after submission
Programming Assignments: People

Teaching Assistants:

• Peter Rautek (peter.rautek@kaust.edu.sa) – programming assignments; assignment presentations
  Office: Bldg 1, Room 2220

• Ronell Sicat (ronell.sicat@kaust.edu.sa) – programming assignments; programming-related questions
  Office: Bldg 1, Room 2101 (lab area)

Help in programming assignments (in this order!):

1. Think about it, read about it, google it!
2. Ask other students/post in forum!
3. Ask TAs (Peter and Ronell)
Programming Assignments: Places

GMSV Building 1

- Ronell Sicat office #2101
- Markus Hadwiger office #2219
- Peter Rautek office #2220

lab-space 2106-WS17
2106-WS18

entrance
Programming Assignments: Requirements

• Submit via bitbucket+git at the latest on day the assignment is due (code, libs, everything that is needed to run your program)

• Submission must include short report (1-2 pages, pdf), including short explanation of algorithms, your solution, problems, how to run it, screenshots

• Personal presentations (soon after submission), present your program live and explain source code (usually 10-15 min)
  – Sign up for presentation slot in advance; sign-up sheet on Peter’s office door, Bldg 1, Room 2220
  – Use your own laptop (preferred!) or test on lab machine to guarantee it runs!
Programming Assignments: Grading

- Submission complete, code working for all the required features
- Documentation complete (report, but also source code comments!)
- Personal presentation
- Optional features, coding style, clean solution
- Every day of late submission reduces points by 10%
- No direct copies from the Internet!
  You have to understand what you program:
  your explanations during the presentations will be part of the grade!
Assignment #1:
   • Querying the GPU (OpenGL and CUDA)  due Feb 10

Assignment #2:
   • Phong shading and procedural texturing (GLSL)  due Mar 3

----- Spring Break: Mar. 28 – Apr. 5 -----  

Assignment #3:
   • Image Processing with (a) GLSL, and (b) CUDA  due Apr 7

Assignment #4:
   • Linear Algebra (CUDA)  due Apr 28
Programming Assignments: Where to Start

- Source code is hosted on bitbucket.org
- Register with your kaust.edu.sa email address (will give you unlimited plan – nice!)
- Go to the repo https://bitbucket.org/rautek/cs380-2014 (or simply search on bitbucket for cs380) and fork it
- Get a git client http://git-scm.com/downloads and clone your own repo
- Follow the readme text-file
- Do your changes in the source code for assignment 1, commit, and push (to your own repo)
Programming Assignment 1 – Setup

- **Setup**
  - git+bitbucket
  - Visual Studio 2010
  - CUDA 5.5

- **Programming**
  - Query hardware capabilities (OpenGL and CUDA)
  - Instructions in readme.txt file
Semester Project

- Choosing your own topic encouraged! (we can also suggest some topics)
  - Pick something that you think is really cool!
  - Can be completely graphics or completely computation, or both combined
  - Can be built on CS380 frameworks, NVIDIA OpenGL SDK, or CUDA SDK

- Write short (1-2 pages) project proposal until mid-March
  - Talk to us before you start writing!
    (content and complexity should fit the lecture)

- Write semester project report (May, before project presentations)

- Present semester project (final exams week in May [11-15])
Reading Assignment #1 (until Feb. 10)

Read (required):

- Orange book, chapter 1 (*Review of OpenGL Basics*)
- Orange book, chapter 2 (*Basics*)

Download:

- NVIDIA CUDA SDK (5.5)

- Install, try out examples, browse code a bit to get a basic feel
- See what examples run on your hardware, and which don‘t
What Are GPUs?

*Graphics* Processing Units

But evolved toward

- Very flexible, massively parallel floating point co-processors
- But not entirely programmable!
- Fixed-function parts have definite advantages (e.g., texture filtering, z-buffering)

We will cover both perspectives

- GPUs for graphics
- GPU computing (GPGPU – general purpose computation on GPU)
What's Coming Up

GPU Roadmap

Volta
Stacked DRAM

Maxwell
Unified Virtual Memory

Kepler
Dynamic Parallelism

Fermi
FP64

Tesla
CUDA

2008  2010  2012  2014
Example: Fluid Simulation and Rendering

• Compute advection of fluid
  – (Incompressible) Navier-Stokes solvers
  – Lattice Boltzmann Method (LBM)

• Discretized domain; stored in 2D/3D textures
  – Velocity, pressure
  – Dye, smoke density, vorticity, …

• Updates in multi-passes

• Render current frame

Courtesy Mark Harris
Example: Volumetric Special Effects

- NVIDIA Demos
  - Smoke, water
  - Collision detection with voxelized solid (Gargoyle)

- Ray-casting
  - Smoke: direct volume rendering
  - Water: level set / isosurface

Courtesy Keenan Crane
Example: Particle Simulation and Rendering

- NVIDIA Particle Demo
Example: Level-Set Computations

- Implicit surface represented by distance field
- The level-set PDE is solved to update the distance field
- Basic framework with a variety of applications
Example: Diffusion Filtering

De-noising

- Original
- Linear isotropic
- Non-linear isotropic
- Non-linear anisotropic
Example: Linear Algebra Operators

Vector and matrix representation and operators

- Early approach based on graphics primitives
- Now CUDA makes this much easier
- Linear systems solvers

Courtesy Krüger and Westermann
Example: GPU Data Structures

Glift: Generic, Efficient, Random-Access GPU Data Structures

• “STL“ for GPUs

• Virtual memory management

Courtesy Lefohn et al.
Thank you.