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

- Sunday + Wednesday, 13:00 – 14:30, Building 9, Room 4137

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

Contact

- Markus Hadwiger: markus.hadwiger@kaust.edu.sa
- Peter Rautek (main contact assignments): peter.rautek@kaust.edu.sa

Prerequisites

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

Lectures
- Part 1: GPU Basics and Architecture (both: 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 Wednesday
  (tentative dates: Sep 12, Sep 26, Oct 10, Oct 31, Nov 14, Nov 28)
  - 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, 2nd ed.
Resources (1)

Textbooks

- GPUs for Graphics: OpenGL 4.0 Shading Language Cookbook, 2nd ed.
Resources (2)

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)
http://www.opengl-redbook.com/

Computer graphics and OpenGL

Current edition: 9\(^{th}\)
OpenGL 4.5
contains extended chapters on GLSL

Available in the KAUST library
also electronically
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 (~August, September)

• 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 (~October)
- GPU texturing, filtering
- GPU (texture) memory management
- GPU frame buffers
- Virtual texturing
GPU Computing (~November)

- GPGPU, important parallel programming concepts
- CUDA memory access
- Reduction, scan
- Linear algebra on GPUs
- Deep learning on GPUs
- Combining graphics and compute
  - Display the results of computations
  - Interactive systems (fluid flow, ...)

Semester project presentations
Programming Assignments: Basics

4 assignments

• Based on C/C++, OpenGL, and CUDA

Organization

1. Explanation in readme, and during lecture (and Q&A sessions if required)
2. Get framework online \(\text{bitbucket+git}\)
3. Submit solution and report online \(\text{bitbucket+git}\) by submission deadline
4. Personal presentation after submission
Teaching Assistants:

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

CS380 machines

Markus Hadwiger office #2219
Peter Rautek office #2220

Visual Computing Center  Building 1
1. Google, Stackoverflow, ...

2. CS380 Forum:
   
   piazza.com/kaust.edu.sa/fall2018/cs380

3. Contact us:
   • Peter Rautek: peter.rautek@kaust.edu.sa
GPU programming comes in different flavors:
• Graphics: OpenGL, DirectX, Vulkan
• Compute: CUDA, OpenCL

In this course we will:
• Play with CUDA and OpenGL
• Wrap our heads around parallelism
• Learn the differences and commonalities of graphics and compute programming

Format:
• 4 Pre-specified programming assignments
• 1 Capstone (semester) project that you can define yourself
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/cs380-2018 (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)
• Contact Peter Rautek if you have problems or questions (peter.rautek@kaust.edu.sa)
Programming Assignment 1

Set up your development environment

- git (https://git-scm.com/downloads)
- Fork the CS380 repository (https://bitbucket.org/rautek/cs380-2018)
- Follow the readme and start coding

Query your graphics card for its capabilities (CUDA and OpenGL)
Programming Assignment 1 – Setup

- Programming
  - Query hardware capabilities (OpenGL and CUDA)
  - Instructions in readme.txt file
- Submission (via bitbucket)
  - Program
    - Short report (1-2 pages, pdf), including short explanation of program, problems and solutions, how to run it, screenshots, etc.
- Personal assessment
  - Meeting at Peter’s office (building 1, office 2220)
  - Max. 15 minutes, present program and source code
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 Sep 2

Assignment #2:
- Phong shading and procedural texturing (GLSL)  
  due Sep 24

Assignment #3:
- Image Processing with (a) GLSL, and (b) CUDA  
- Convolutional layers (CUDA)  
  due Oct 14

Assignment #4:
- Linear Algebra (CUDA)  
  due Nov 4
Semester Project

• Choosing your own topic encouraged!
  (we will 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 by end of Sep (announced later)
  • Talk to us before you start writing!
    (content and complexity should fit the lecture)
• Submit semester project with report (deadline: Dec 9)
• Present semester project (we will schedule event in final exams week)
Read (required):

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

Ray tracing in CUDA kernels, or ray tracing cores

- Microsoft DXR (DX12 API), Vulkan, NVIDIA OptiX / RTX
- NVIDIA Turing: “World’s First Ray Tracing GPU“ Quadro RTX, GeForce RTX
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: Machine Learning / Deep Learning

Perfect fit for massively parallel computation

- NVIDIA Volta Architecture: Tensor Cores (mixed-prec. 4x4 matrix mult plus add)
- NVIDIA Turing Architecture: Improved Tensor Cores, ...

Frameworks

- TensorFlow,
- Caffe,
- Pytorch,
- Teano, ...

WHY ARE GPUs GOOD FOR DEEP LEARNING?

<table>
<thead>
<tr>
<th>Neural Networks</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherently Parallel</td>
<td>✓</td>
</tr>
<tr>
<td>Matrix Operations</td>
<td>✓</td>
</tr>
<tr>
<td>FLOPS</td>
<td>✓</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>✓</td>
</tr>
</tbody>
</table>

GPUs deliver --
- same or better prediction accuracy
- faster results
- smaller footprint
- lower power
- lower cost

[Lee, Ranganath & Ng, 2007]
Example: GPU Data Structures

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

- “STL” for GPUs
- Virtual memory management

 Courtesy Lefohn et al.
Thank you.