CS 380 - GPU and GPGPU Programming
Lecture 1: Introduction

Markus Hadwiger

Peter Rautek

King Abdullah University of Science and Technology
KAUST

Lecture Overview and Resources

Markus Hadwiger

Peter Rautek

Lecture Overview

Goals

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

Time and location

- Monday + Thursday, 10:00 – 11:30, Room 4227, Bldg. 9
- Default slot for make-up lectures: Tuesday, 14:30 – 16:00

Webpage: https://vccvisualization.org/CS380_GPU_and_GPGPU_Programming/

Contact:

- **Markus Hadwiger:** `markus.hadwiger@kaust.edu.sa`
- **Peter Rautek** (main contact assignments): `peter.rautek@kaust.edu.sa`
- **Julio Rey** (programming questions): `julio.reyramirez@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 Compute
- Part 3: GPUs for Graphics

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

Assignments

- 5 programming assignments
- Weekly reading assignments (required; also some optional)

Quizzes

- 4 quizzes, throughout the semester, 30 min each; announced at least a week in advance
- From lectures and (required) reading assignments

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

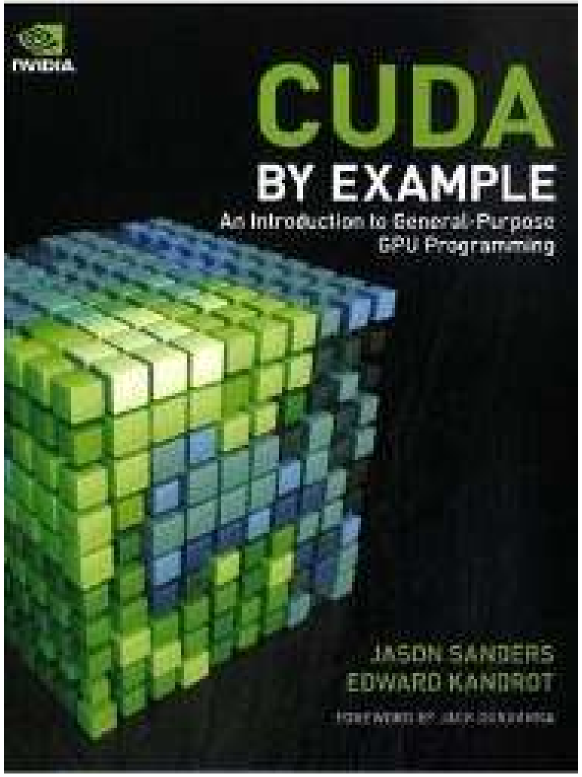
Grading: 40% programming assignments; 30% semester project; 30% quizzes

Lecture attendance is mandatory: skip up to 4 lectures unexcused, but not more

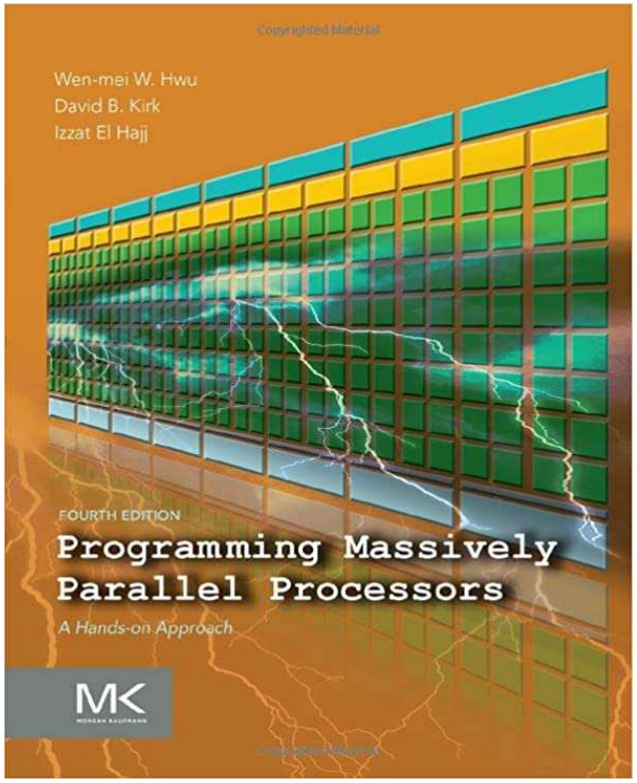
Resources (1) – GPU Compute – Textbooks

- Programming Massively Parallel Processors: A Hands-on Approach, 4th ed.
- CUDA by Example: An Introduction to General-Purpose GPU Programming, Jason Sanders, Edward Kandrot

Online through KAUST Library



Online through KAUST Library

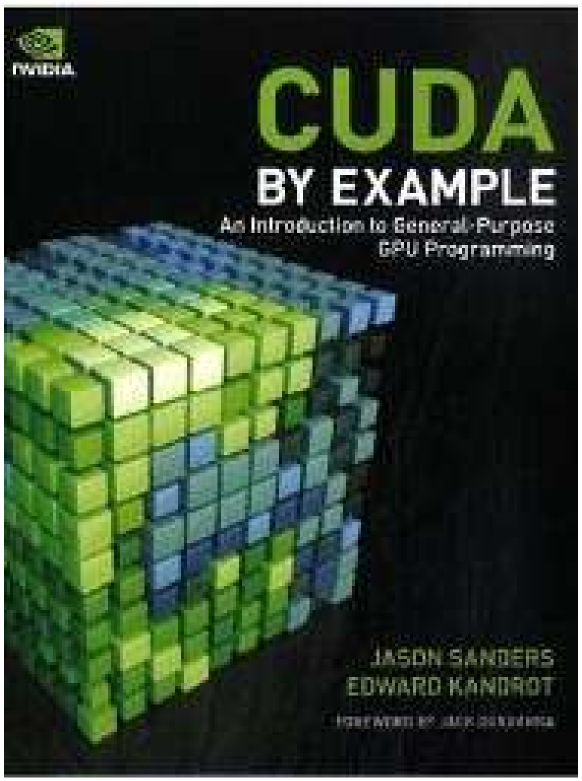


4th ed.

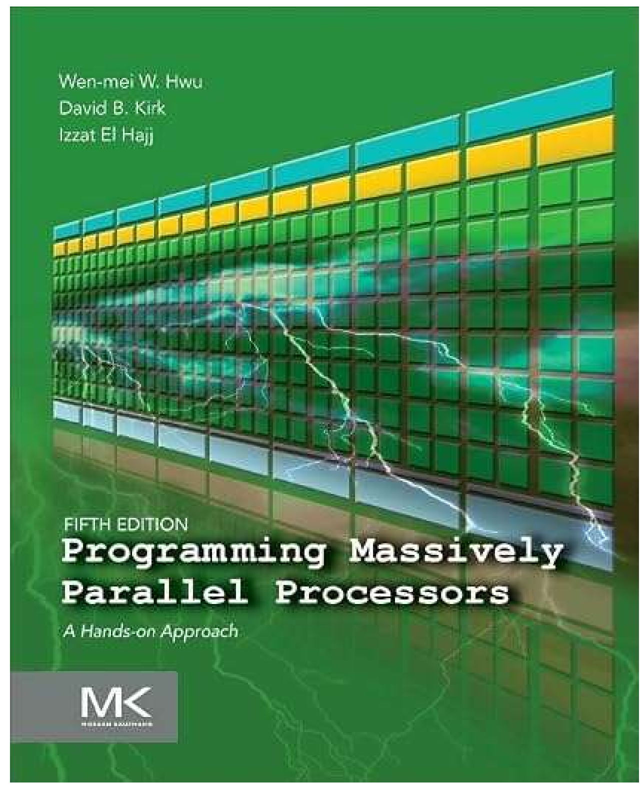
Resources (1) – GPU Compute – Textbooks

- Programming Massively Parallel Processors: A Hands-on Approach, 4th ed.
- CUDA by Example: An Introduction to General-Purpose GPU Programming, Jason Sanders, Edward Kandrot

Online through KAUST Library



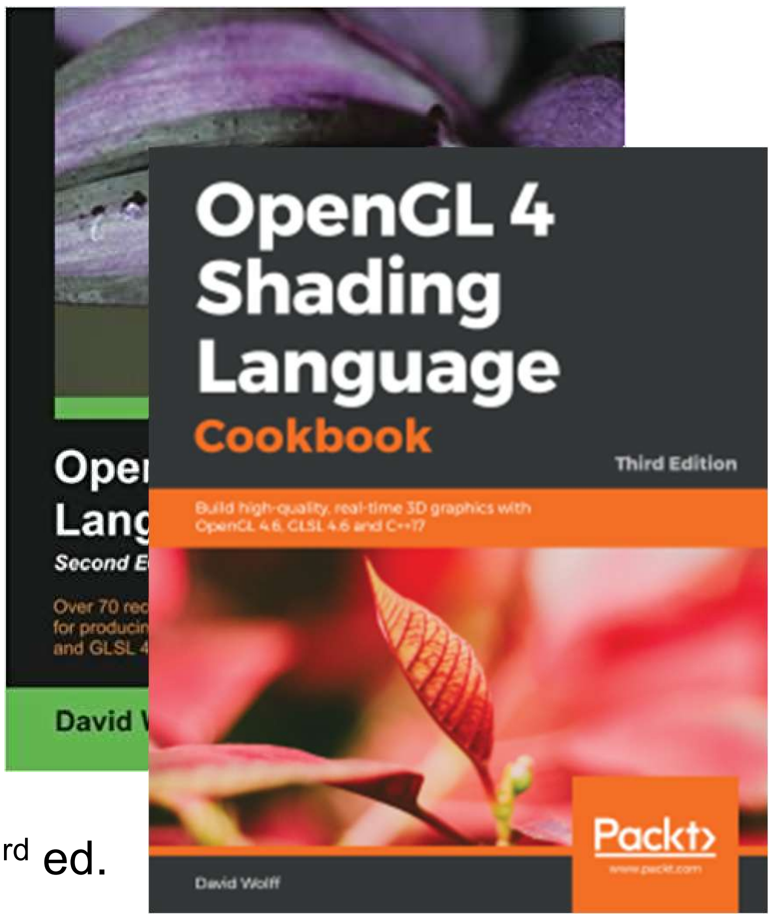
Online through KAUST Library



5th ed.

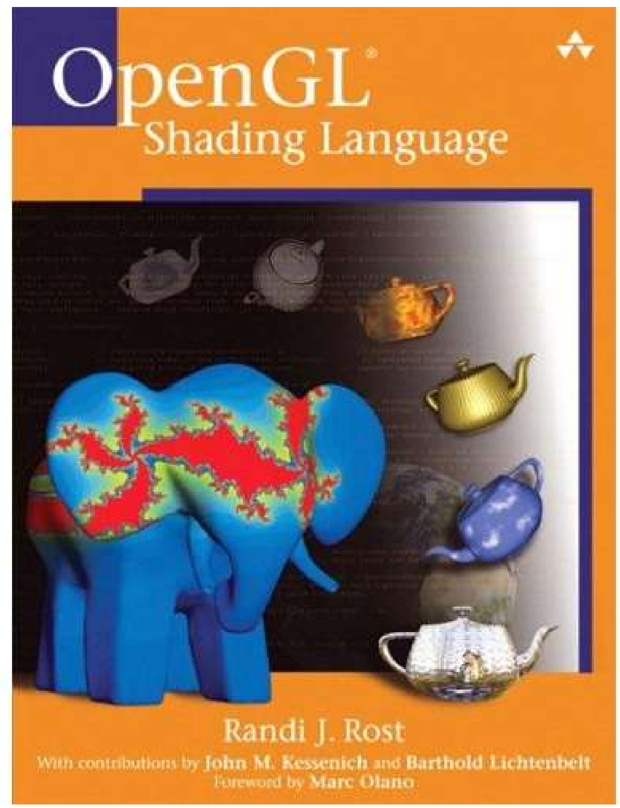
Resources (2) – Graphics (OpenGL) Textbooks

- OpenGL 4 Shading Language Cookbook, 2nd or 3rd ed.
- OpenGL Shading Language (orange book)



2nd ed.

3rd ed.



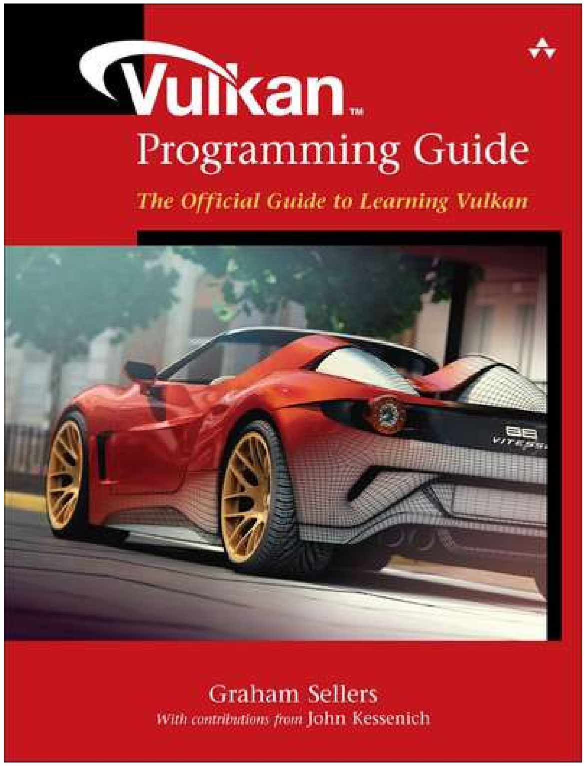
3rd ed.

OpenGL 3.1, GLSL 1.4
outdated in several aspects (no geometry shaders)
but the basics are still very nice

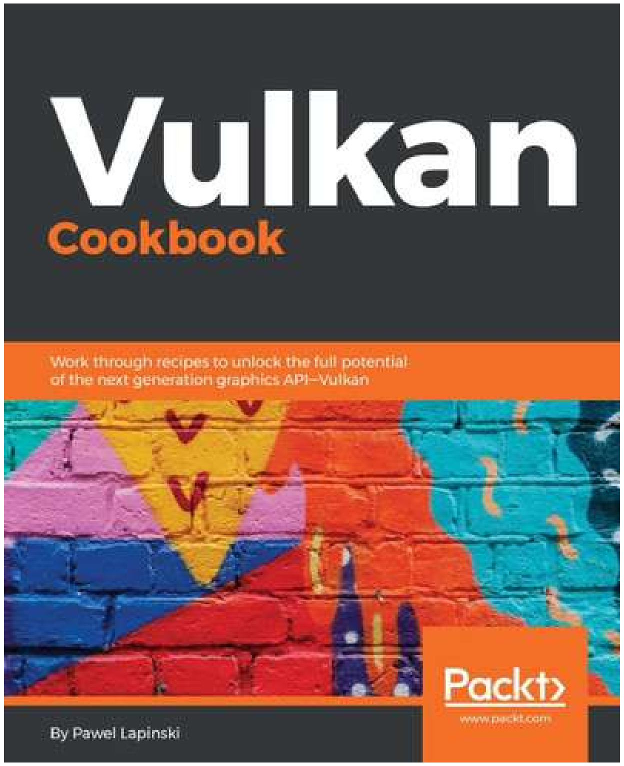
Resources (2) – Graphics (Vulkan) Textbooks

- Vulkan Programming Guide (2016)
- Vulkan Cookbook (2017)

Online through KAUST Library



Online through KAUST Library



Resources (3) – Graphics – Reference

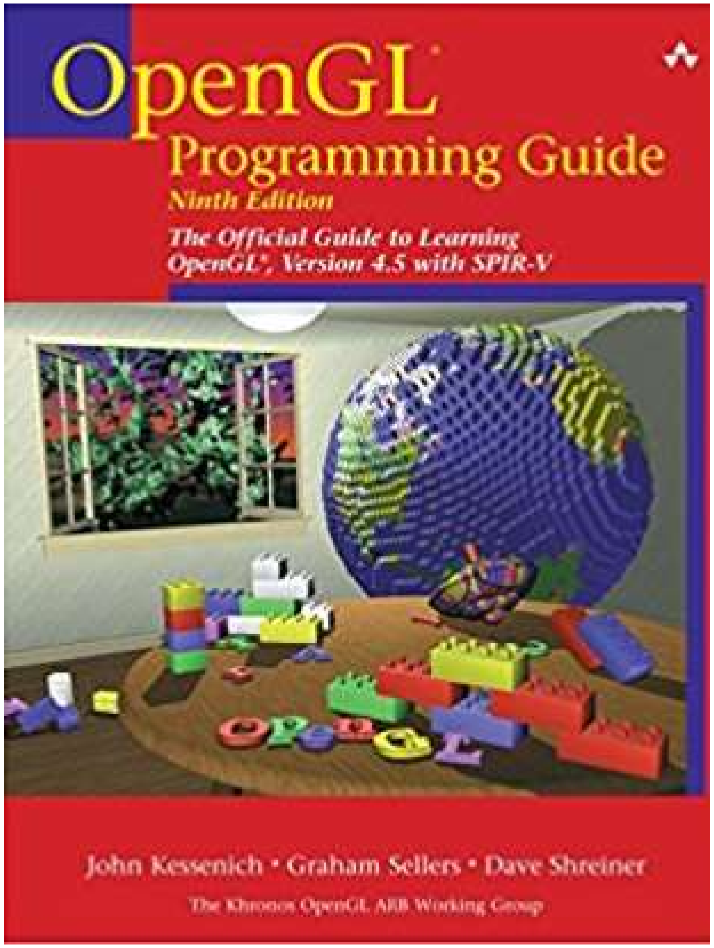
OpenGL Programming Guide (red book)

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

Computer graphics and OpenGL

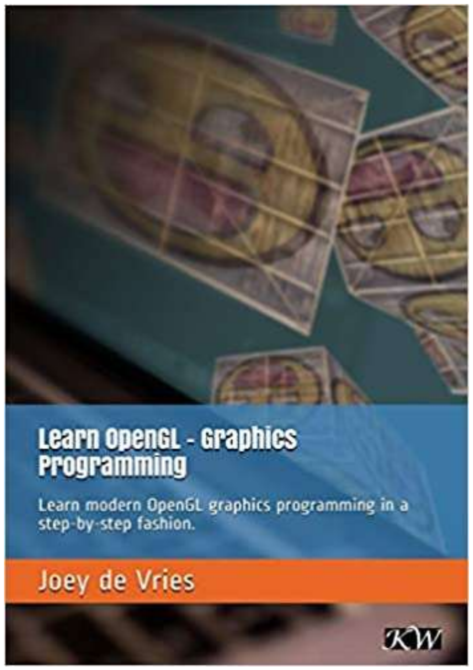
Current edition: 9th
OpenGL 4.5 (with SPIR-V)
contains extended chapters on GLSL

Available in the KAUST library
and also electronically



Resources (4) – Graphics – Websites/Tutorials

Learn **OpenGL**

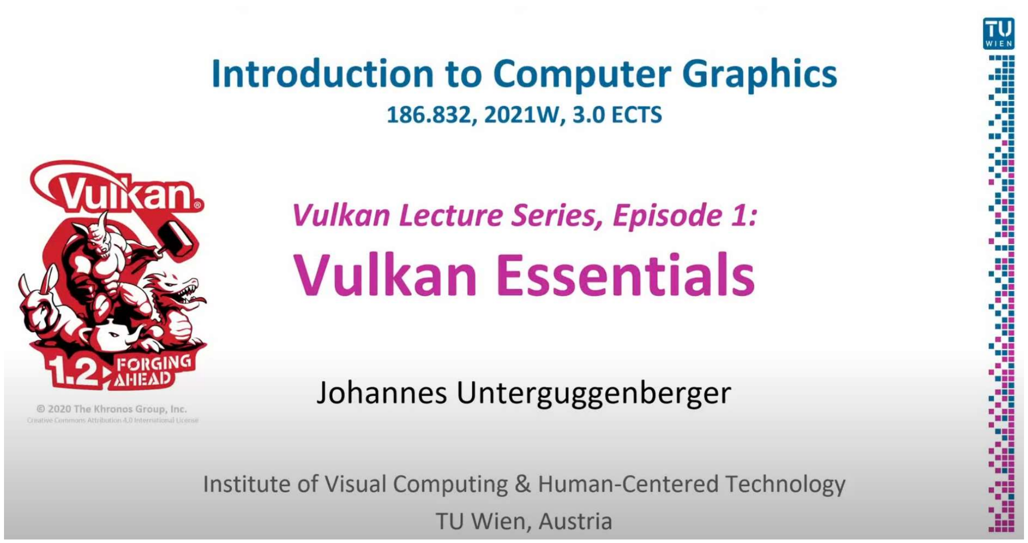


Nice introduction to modern OpenGL
`https://learnopengl.com/`

Free book as pdf:

`https://learnopengl.com/book/book_pdf.pdf`

YouTube lecture series on **Vulkan**:



`https://youtu.be/tLwbj9qys18`

Resources (5) – Official Websites and Others

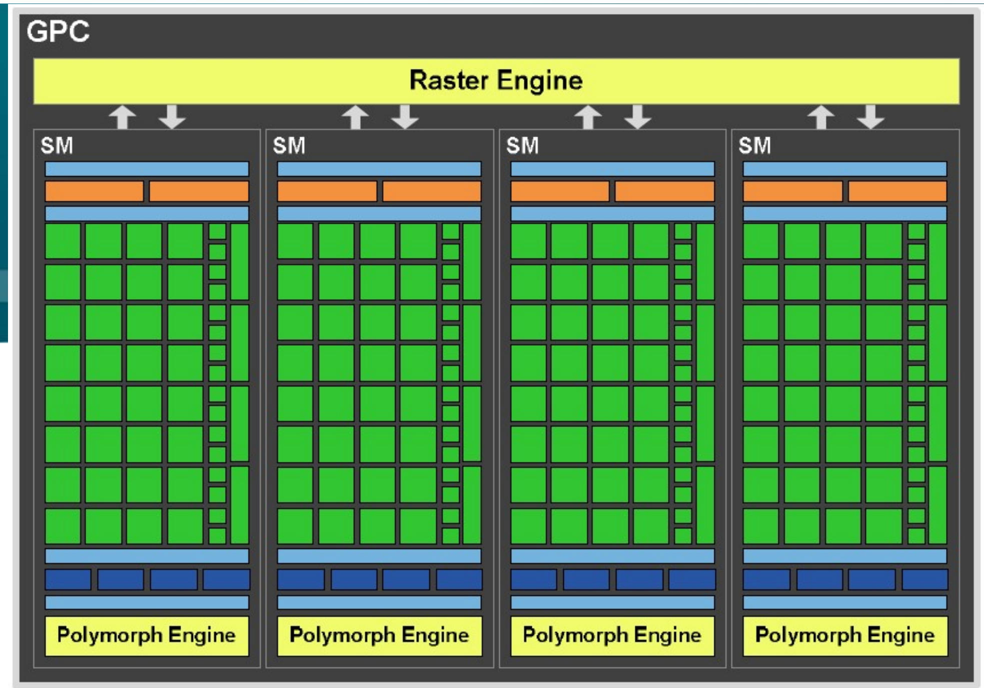
`https://vccvisualization.org/CS380_GPU_and_GPGPU_Programming/`

- OpenGL (4.6): www.opengl.org
  www.khronos.org/files/opengl46-quick-reference-card.pdf
- CUDA (13.3): developer.nvidia.com/cuda-toolkit/
- Vulkan (1.4): www.vulkan.org
- OpenCL (3.0): www.khronos.org/opencl/

Very nice resources for techniques, algorithms and data structures:

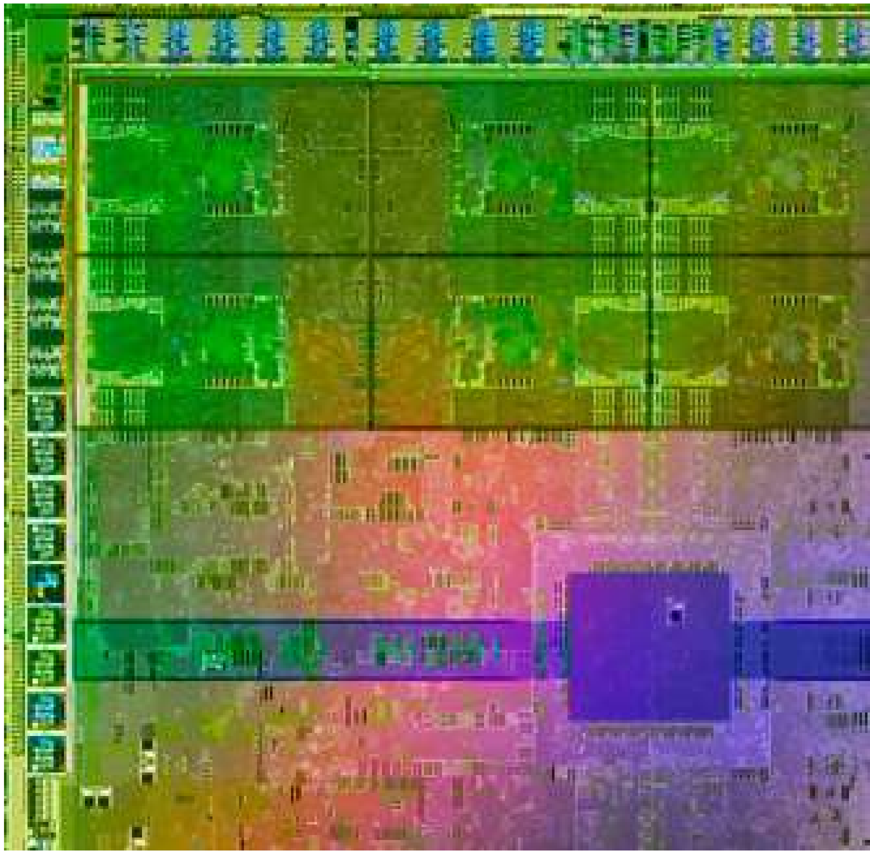
- *GPU Gems* books 1-3 (available online)
- *GPU Computing Gems*, Vol. 1 + 2 (Emerald/Jade edition)
- *Ray Tracing Gems* (2019) and *Ray Tracing Gems II* (2021)

Syllabus (1)



GPU Basics and Architecture (~September, early October)

- Introduction
- **GPU architecture**
- How compute/shader cores work
- GPU shading and GPU compute APIs
  - General concepts and overview
  - Learn syntax details on your own !
    - CUDA book
    - GLSL book
    - Vulkan tutorial
    - online resources, ...



NVIDIA Architectures (since first CUDA GPU)

Tesla [CC 1.x]: 2007-2009

- G80, G9x: 2007 (Geforce 8800, ...)
  GT200: 2008/2009 (GTX 280, ...)

Fermi [CC 2.x]: 2010 (2011, 2012, 2013, …)

- GF100, ... (GTX 480, ...)
  GF104, ... (GTX 460, ...)
  GF110, ... (GTX 580, ...)

Kepler [CC 3.x]: 2012 (2013, 2014, 2016, …)

- GK104, ... (GTX 680, ...)
  GK110, ... (GTX 780, GTX Titan, ...)

Maxwell [CC 5.x]: 2015

- GM107, ... (GTX 750Ti, ...); [Nintendo Switch]
  GM204, ... (GTX 980, Titan X, ...)

Pascal [CC 6.x]: 2016 (2017, 2018, 2021, 2022, …)

- GP100 (Tesla P100, ...)
- GP10x: x=2,4,6,7,8, ...
  (GTX 1060, 1070, 1080, Titan X *Pascal*,
  Titan Xp, ...)

Volta [CC 7.0, 7.2]: 2017/2018

- GV100, ...
  (Tesla V100, Titan V, Quadro GV100, ...)

Turing [CC 7.5]: 2018/2019

- TU102, TU104, TU106, TU116, TU117, ...
  (Titan RTX, RTX 2070, 2080 (Ti), GTX 1650, 1660, ...)

Ampere [CC 8.0, 8.6, 8.7, 8.8]: 2020

- GA100, GA102, GA104, GA106, ...; [Nintendo Switch 2]
  (A100, RTX 3070, 3080, 3090 (Ti), RTX A6000, ...)

Hopper [CC 9.0], Ada Lovelace [CC 8.9]: 2022/23

- GH100, AD102/103/104/106/107, ...
  (H100, H200, GH200, L20, L40, L40S, L2, L4,
  RTX 4080 (12/16 GB), RTX 4090, RTX 6000 (Ada), ...)

Blackwell [CC 10.0, 10.1(→11.0), 10.3, 12.0, 12.1]: 2024/2025

- GB100, GB200, GB202/203/205/206/207, G10, ...
  (RTX 5080/5090, HGX B200/B300, GB200/GB300 NVL72,
  RTX 4000/5000/6000 PRO Blackwell, B40, ...)

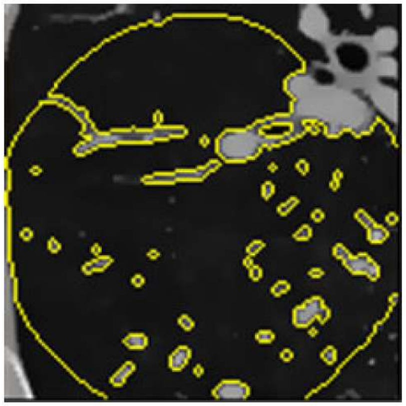
Rubin: 2026 ?   Feynman: 2028 ?

see https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units
and https://en.wikipedia.org/wiki/CUDA

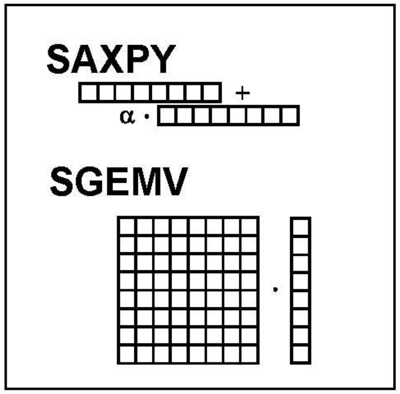
Syllabus (2)

GPU Computing (~October)

- 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, ...)



segmentation



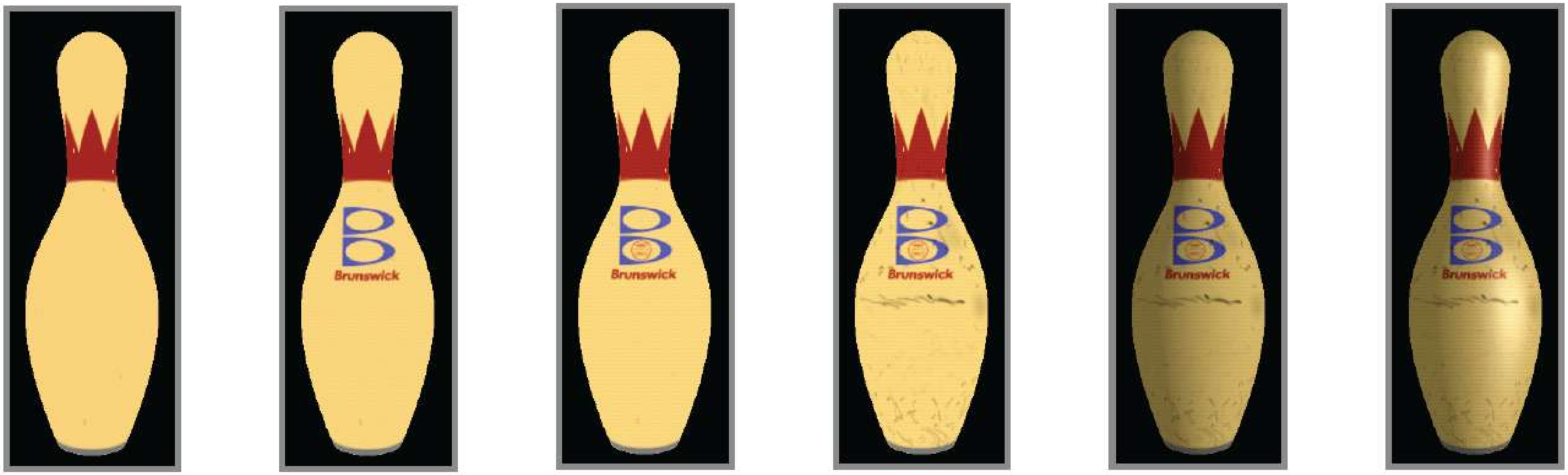
linear algebra

Syllabus (3)

GPU Graphics (~November)

- GPU (virtual) texturing, filtering
- GPU (texture) memory management
- Modern game engine technologies

Semester project presentations



GPGPU Examples

Markus Hadwiger

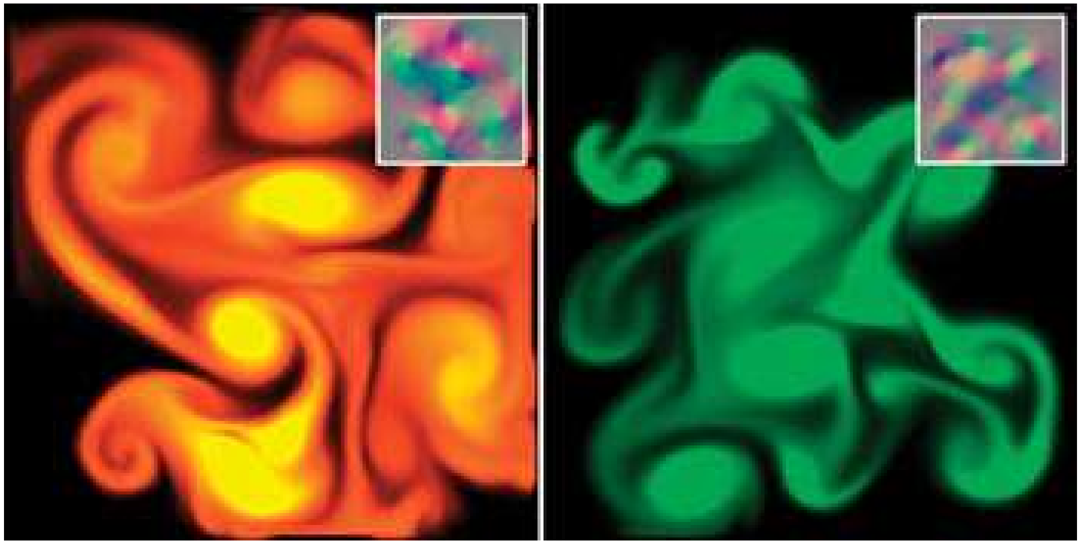
Peter Rautek

King Abdullah University of Science and Technology

KAUST

Example: Fluid Simulation and Rendering

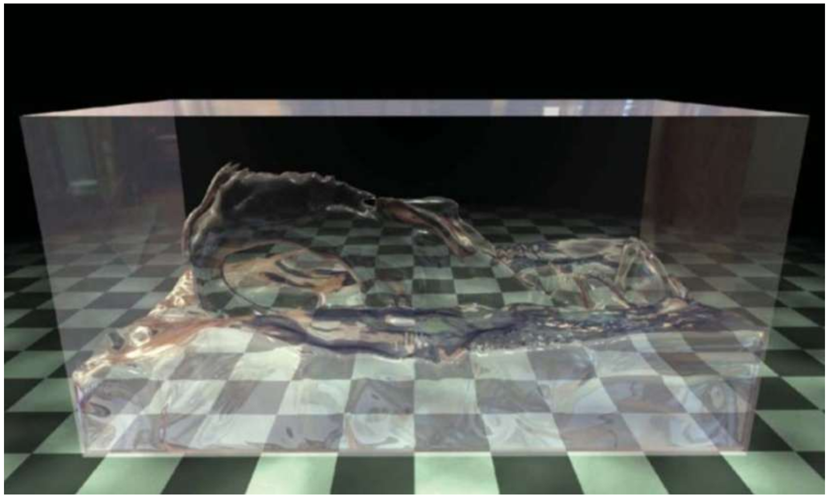
- Compute advection of fluid
  - (Incompressible) Navier-Stokes solvers
  - Lattice Boltzmann Method (LBM)
- Discretized domain; stored in 2D/3D textures
  - Velocity, pressure
  - Dye, smoke density, vorticity, …
- Updates in multi-passes
- Render current frame

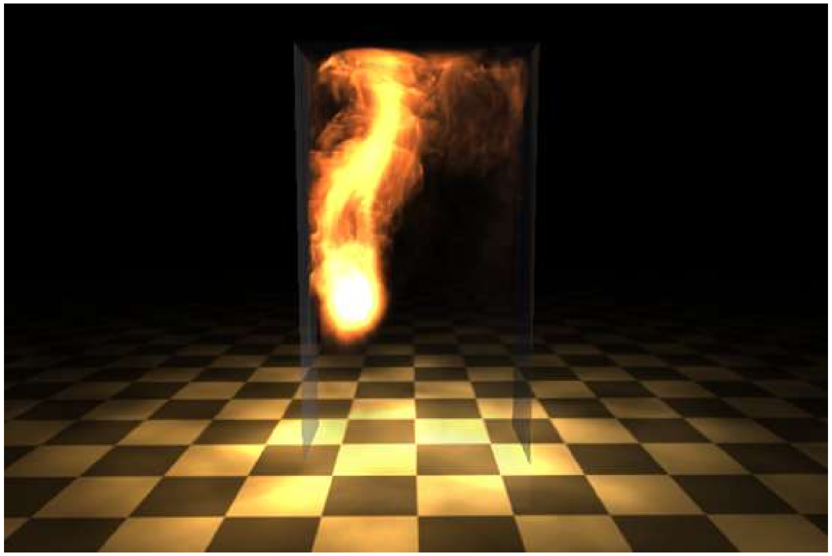


Courtesy Mark Harris

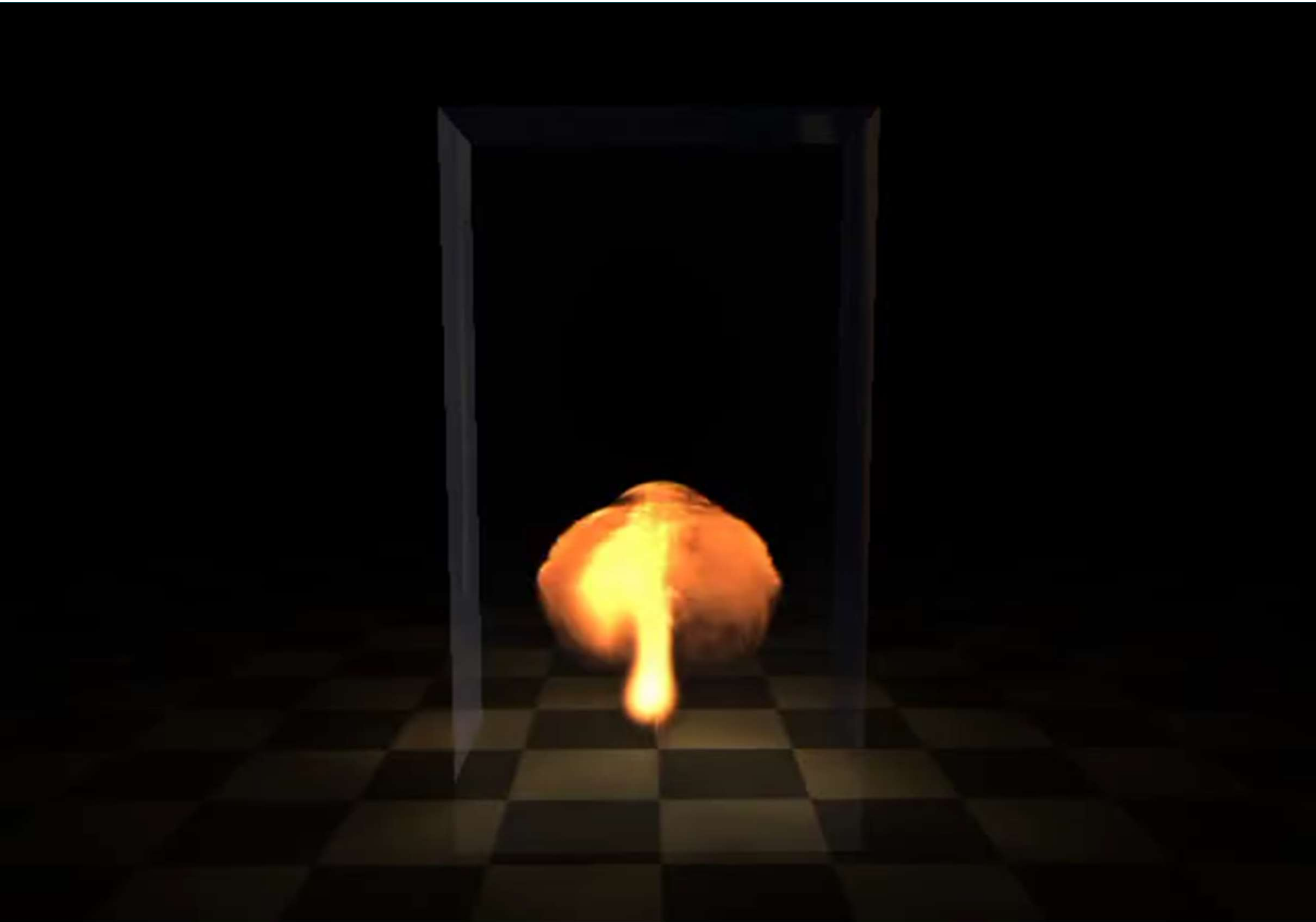
Example: Volumetric Special Effects

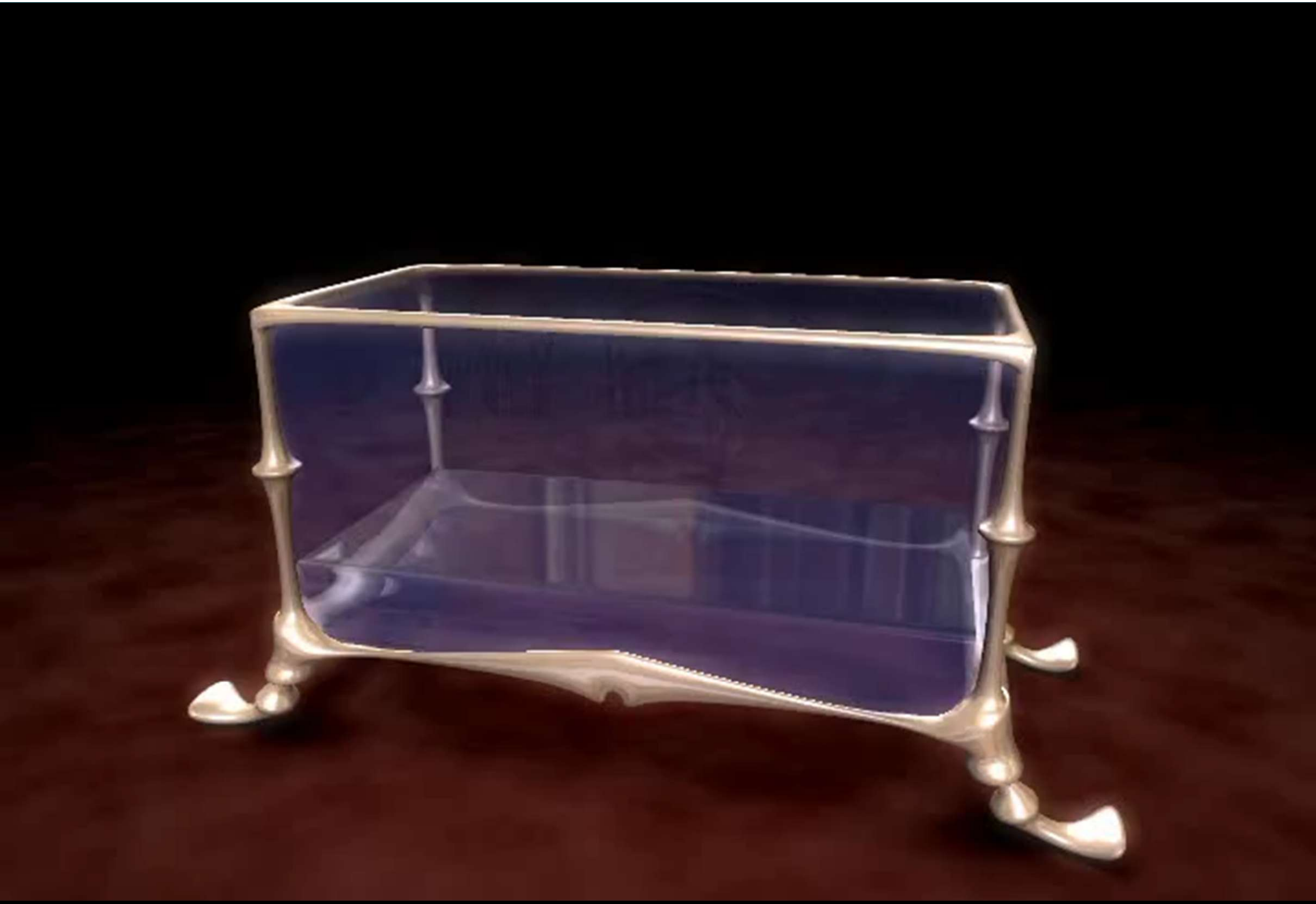
- NVIDIA Demos
  - Smoke, water
  - Collision detection with voxelized solid (Gargoyle)
- Ray-casting
  - Smoke: direct volume rendering
  - Water: level set / isosurface





Courtesy Keenan Crane





Example: Particle Simulation and Rendering

- NVIDIA Particle Demo

Example: Ray Tracing

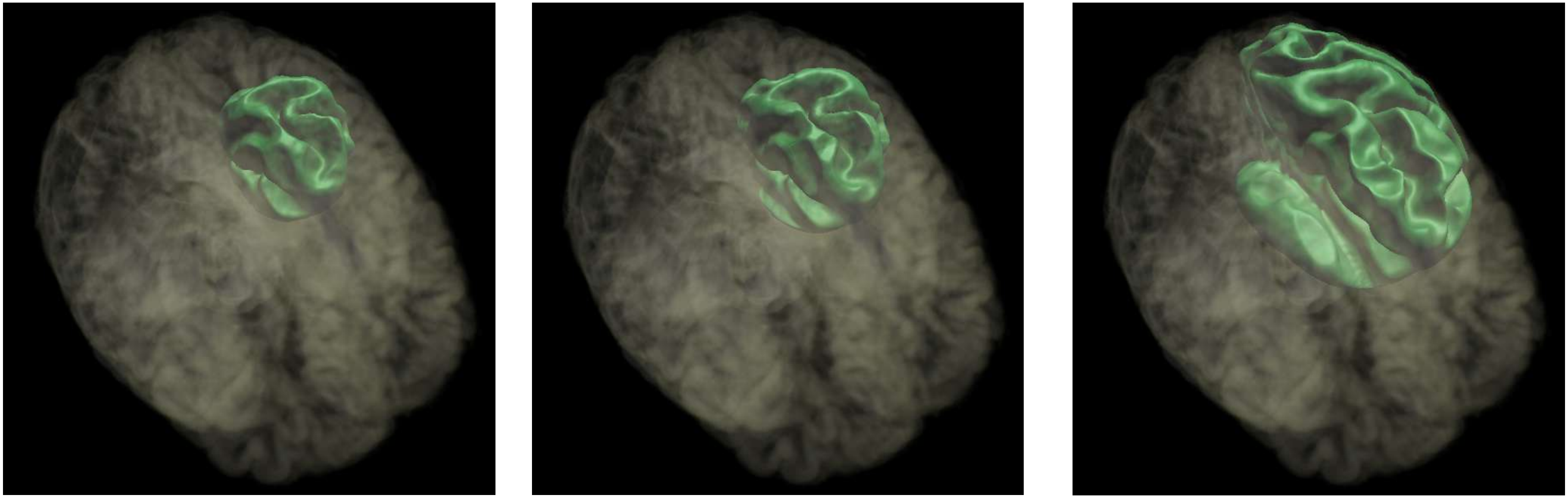
Ray tracing in hardware (ray tracing cores: ray/triangle isect, BVH)

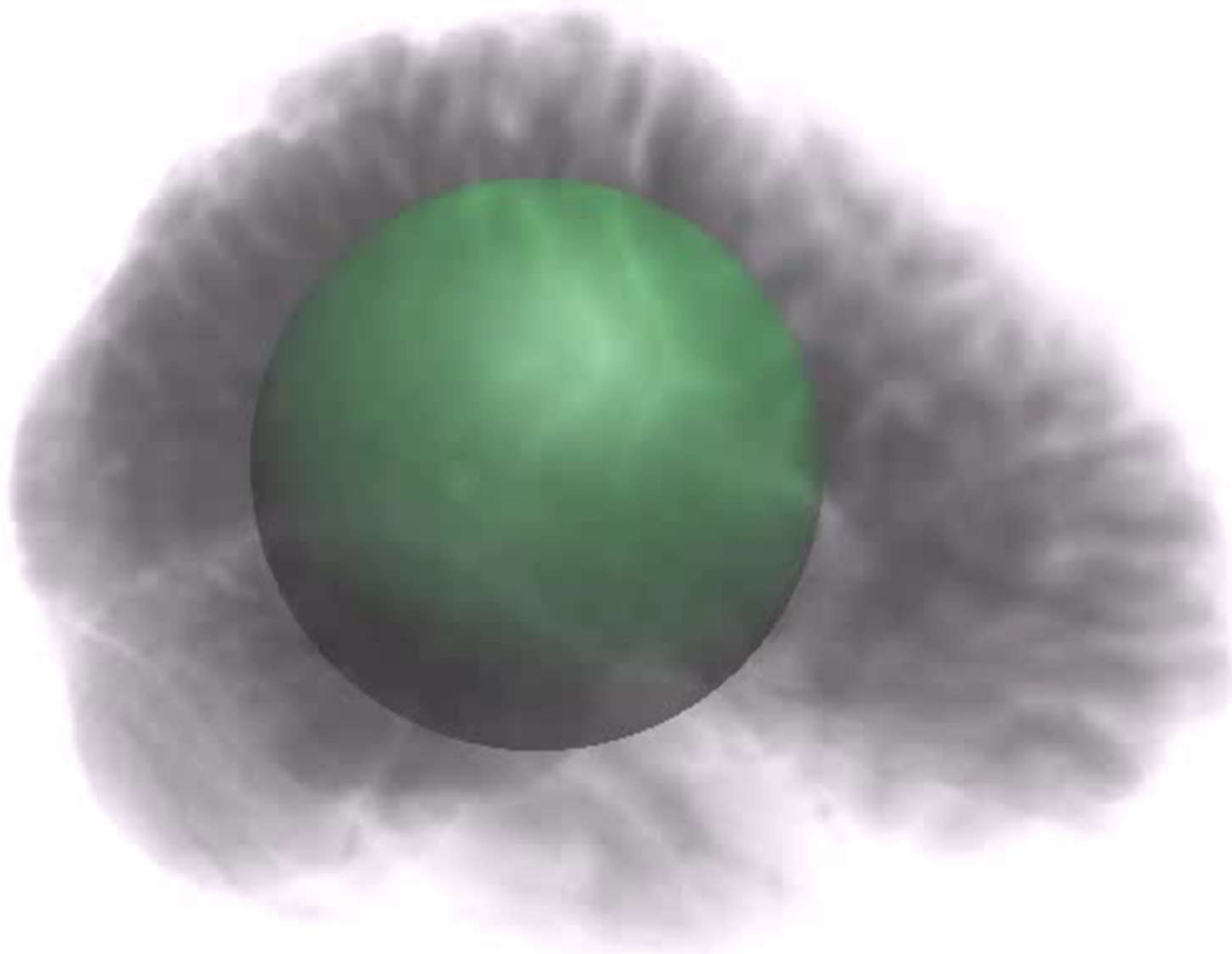
- Microsoft DXR (DX12 Ultimate API), Vulkan, NVIDIA OptiX
- NVIDIA Turing: “World‘s First Ray Tracing GPU“ Quadro RTX, Geforce RTX
- AMD RDNA 2 (also in PS5, Xbox Series X), upcoming Intel Arc (Alchemist, 2022)

Unreal Engine 4 (2018, youtube), Nvidia RTX (2021, youtube), Unreal Engine 5 (2025, youtube)

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

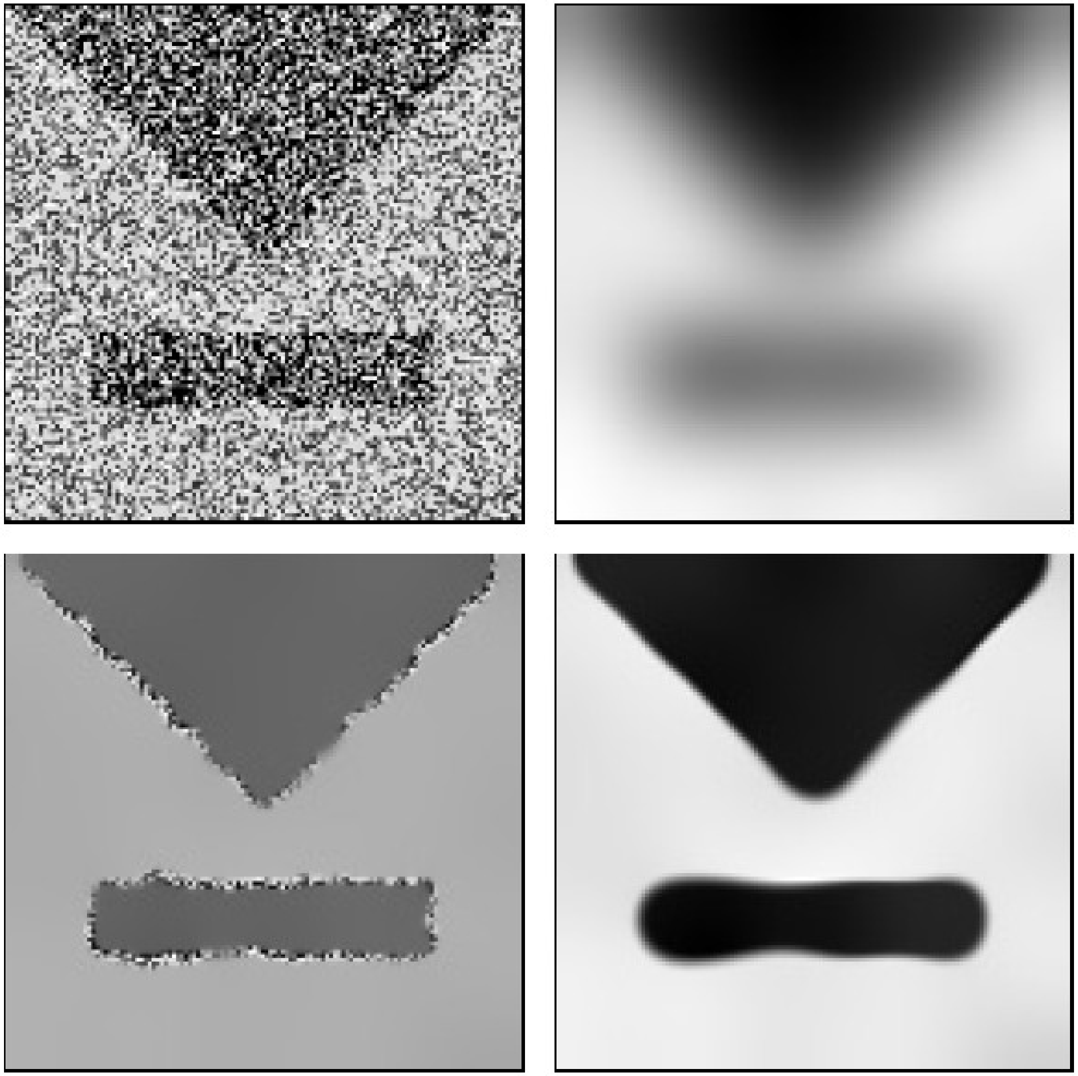




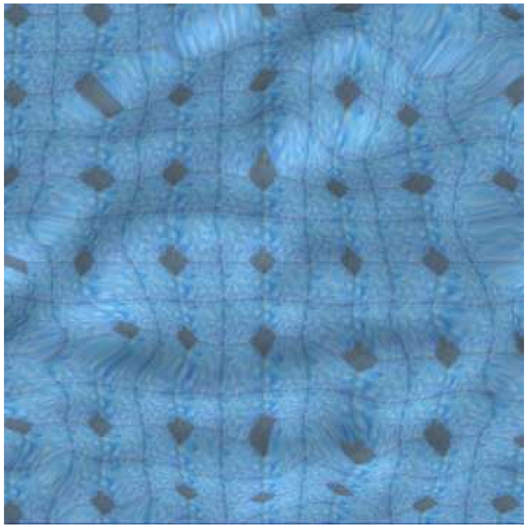
Example: Diffusion Filtering

De-noising

- Original
- Linear isotropic
- Non-linear isotropic
- Non-linear anisotropic

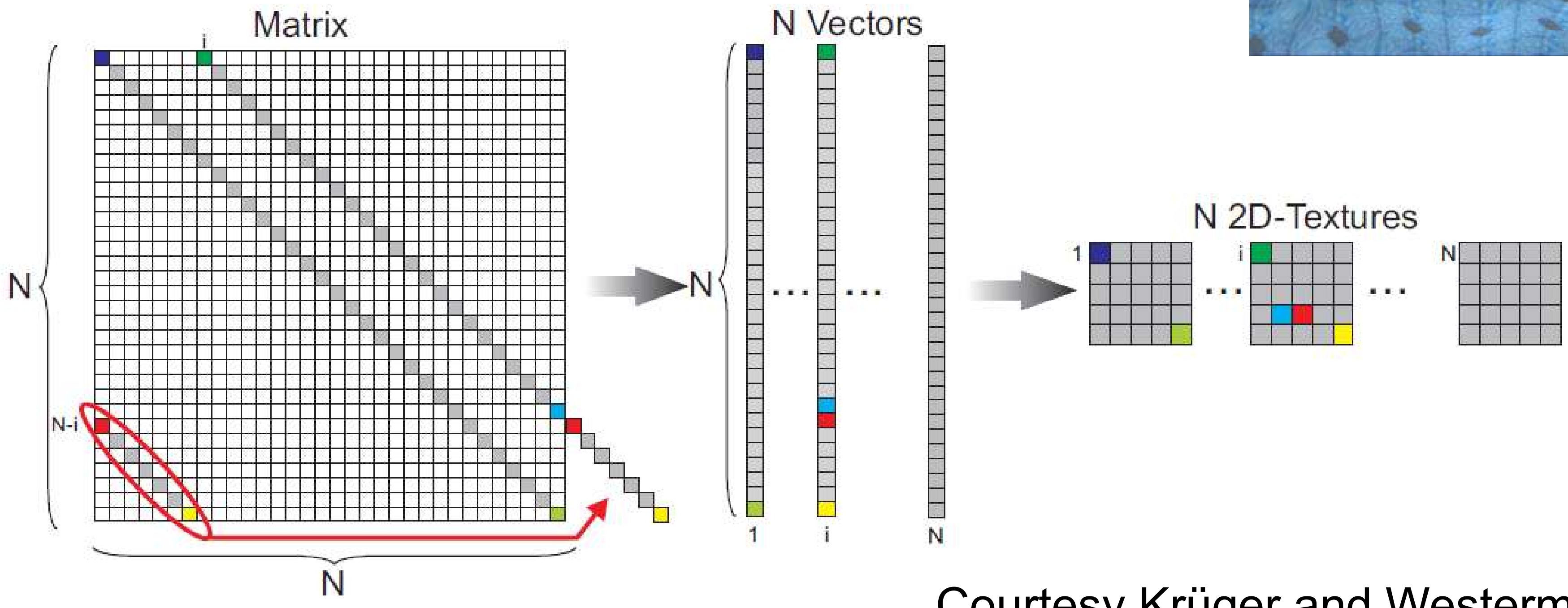


Example: Linear Algebra Operators



Vector and matrix representation and operators

- Early approach based on graphics primitives
- Now CUDA makes this much easier (+ lots of libraries)
- Linear systems solvers



Courtesy Krüger and Westermann

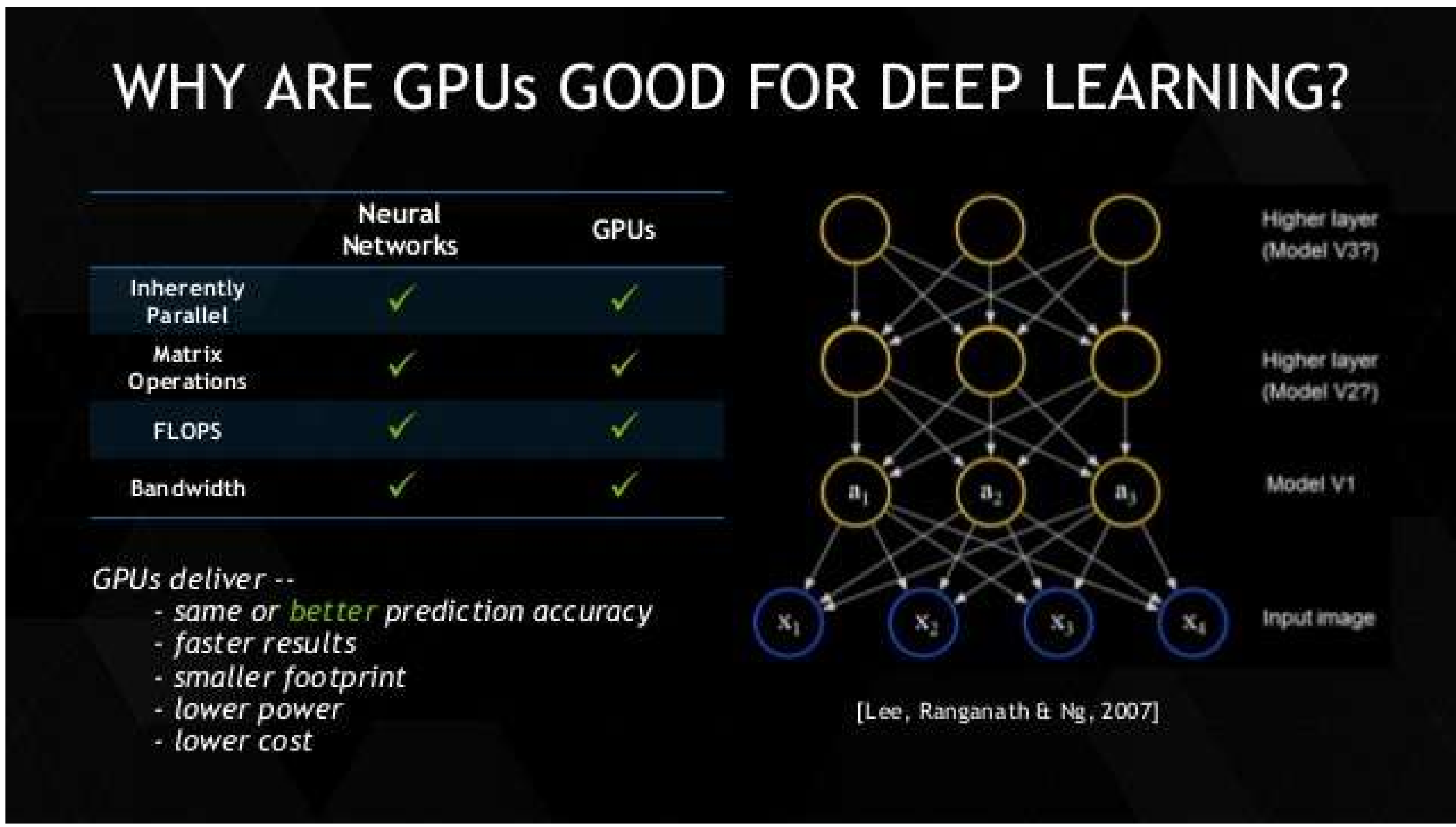
Example: Machine Learning / Deep Learning

Perfect fit for massively parallel computation

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

Frameworks

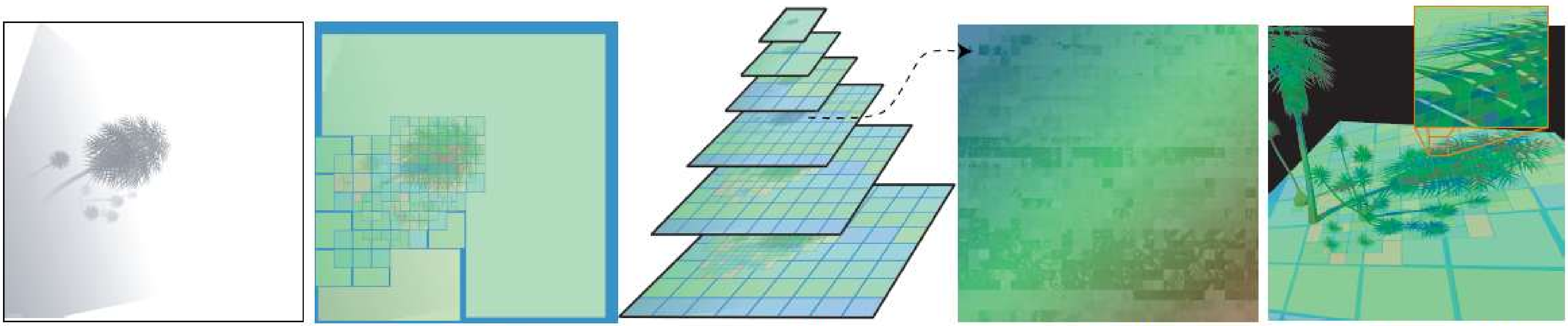
- TensorFlow,
  PyTorch,
  Caffe,
  ...



Example: GPU Data Structures

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

- “STL“ for GPUs
- Virtual memory management



(a) Virtual Domain (b) Adaptive Tiling (c) Page Table (d) Physical Memory (e) Adaptive Shadow Map

Courtesy Lefohn et al.

Programming Assignments - Organization

Markus Hadwiger

Peter Rautek

Programming Assignments: Basics

5 assignments

- Framework based on C/C++ and several GPU APIs (**CUDA**, **Vulkan**, OpenGL, OpenCL)

Organization

1. Explanation in readme, and during lecture (and Q&A sessions if required)
2. Get framework online (*github+git*)
3. Submit solution and report online (*github+git*) by submission deadline
4. Personal presentation and assessment after submission

Programming Assignments: People

Teaching Assistants:

- Peter Rautek (**`peter.rautek@kaust.edu.sa`**)
  programming assignments, assignment presentations
- Julio Rey Ramirez (**`julio.reyramirez@kaust.edu.sa`**)
  programming questions, general help

Need Help?

1. Google, Stackoverflow, ChatGPT, …

2. Ask your fellow students
   Discussions and explanations are encouraged
   (but: copying code is not allowed!)

3. Contact us:
   Peter: peter.rautek@kaust.edu.sa
   Julio: julio.reyramirez@kaust.edu.sa

Playing with the GPU

GPU programming comes in different flavors:

- Compute: CUDA, OpenCL, HIP;
  compute API parts of Vulkan, OpenGL, etc.
- Graphics: Vulkan, OpenGL, DirectX

In this course we will:

- Learn to use compute APIs like CUDA and OpenCL and graphics APIs like Vulkan and OpenGL
- Wrap our heads around parallelism
- Learn the differences and commonalities of graphics and compute programming

Format:

- 5 Pre-specified programming assignments
- 1 Capstone (semester) project that you can define yourself

Programming Assignments: Where to Start

- Source code is hosted on *github.com*
- Go to the github repo (Peter will send you info)
- 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`)

C++ Programming and Graphics API Tutorial

Optional and on-demand:

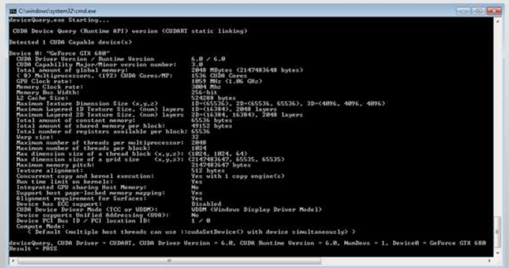
- Short tutorials and tutor sessions
(attendance optional, but recommended)
- To make it easier to get started with C++, Vulkan/OpenGL
- If you have questions/problems when you come to the tutorial, that's even better!

Programming Assignment 1

Set up your development environment

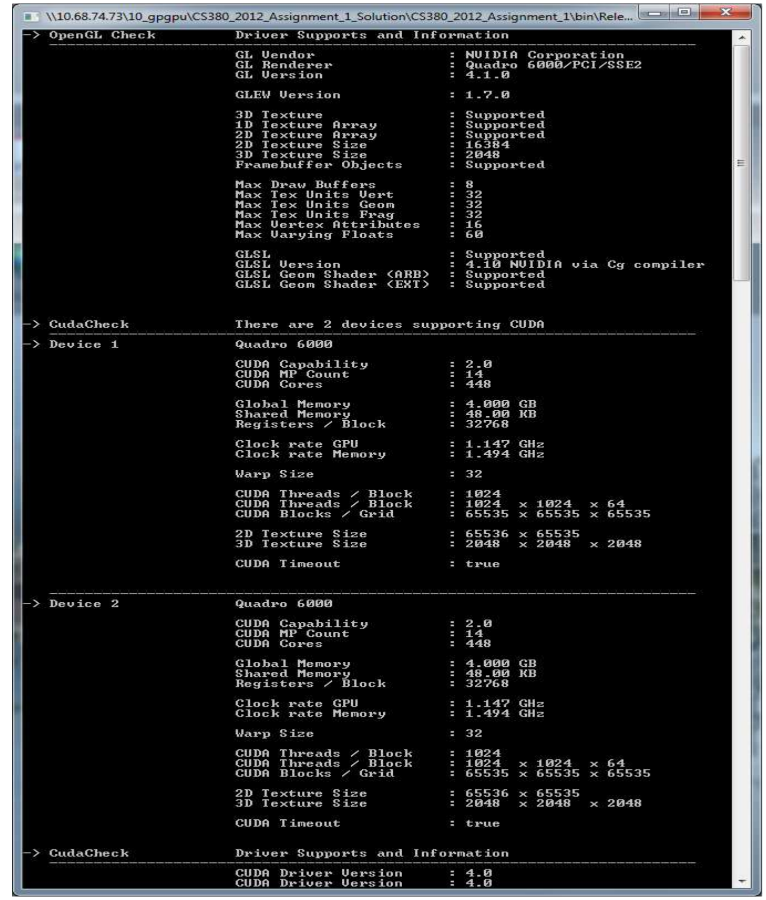
- Visual Studio (either 2019 or 2022) (https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=16)
- CUDA 13.3 Update 1 (https://developer.nvidia.com/cuda-downloads)
- git (https://git-scm.com/downloads)
- Fork the CS 380 repository (https://github.com/rautekp/cs380-2026)
- Follow the readme and start coding

Query your graphics card for its capabilities (CUDA and OpenGL)



Programming Assignment 1 – Setup

- Programming
  - Query hardware capabilities (Vulkan, OpenGL, and CUDA)
  - Instructions in readme.txt file
- Submission (via github)
  - Program
  - Short report (1-2 pages, pdf), including short explanation of program, problems and solutions, how to run it, screenshots, etc.
- Personal assessment
  - Meeting with Peter
  - Max. 15 minutes, present program + 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%
- You have to *understand* your program:
  Your explanations during the assignment presentations will determine your grade!

Programming Assignments: Schedule (tentative)

Assignment #1:

- **Querying the GPU (Graphics and Compute APIs)** due Sep 6

Assignment #2:

- **GPU Compute – Data Parallel Processing** due Sep 20

Assignment #3:

- **GPU Compute – Porting Sequential to Parallel Code** due Oct 4

Assignment #4:

- **Graphics – Rasterization Pipeline** due Oct 25

Assignment #5:

- **Graphics – Compute Shaders (SPH Simulation)** due Nov 15

Semester / Capstone 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 CS 380 frameworks, NVIDIA OpenGL SDK, CUDA SDK, ...
- Write short (1-2 pages) project proposal by early Oct *(announced later)*
  - Talk to us before you start writing!
    (content and complexity should fit the lecture)
- Submit semester project with report (deadline: Dec 13)
- Present semester project, event in final exams week: Dec 14 (tentative!)

Reading Assignment #1 (until Sep 7)

Read (required):

- Programming Mass. Parallel Proc. book, 4th ed., Chapter 1 (*Introduction*)
- Programming Mass. Parallel Proc. book, 2nd ed., Chapter 2 (*History of GPU Computing*)
- OpenGL Shading Language (orange) book, Chapter 1 (*Review of OpenGL Basics*)

Read (optional):

- OpenGL Shading Language 4.6 (current: Aug 14, 2023) specification: Chapter 2

  `https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.60.pdf`

- Download OpenGL 4.6 (current: May 5, 2022) specification

  `https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf`

Thank you.