

# CS 380 - GPU and GPGPU Programming

## Lecture 1: Introduction

Peter Rautek, KAUST

Markus Hadwiger, KAUST

# Lecture Overview and Ressources

Peter Rautek, KAUST

Markus Hadwiger, KAUST



# 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 3120, Bldg. 9

Webpage: [https://vccvisualization.org/CS380\\_GPU\\_and\\_GPGPU\\_Programming/](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
- **Xingdi Zhang** (programming questions): xingdi.zhang@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!

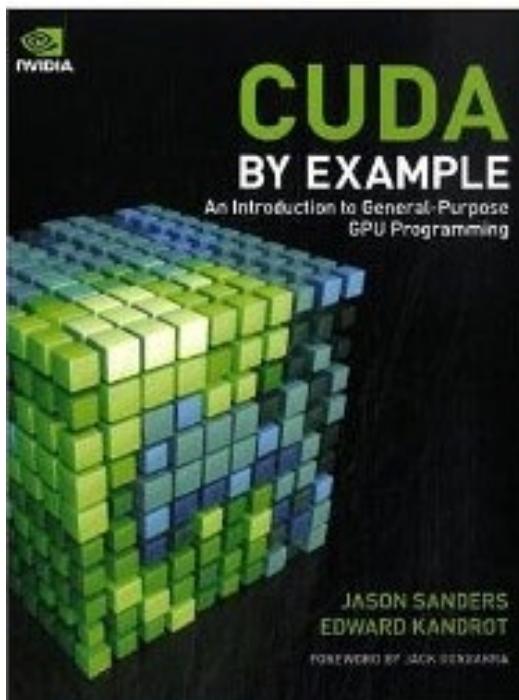
Grading: 40% programming assignments; 30% semester project; 30% quizzes

# Resources (1) – GPU Compute – Textbooks

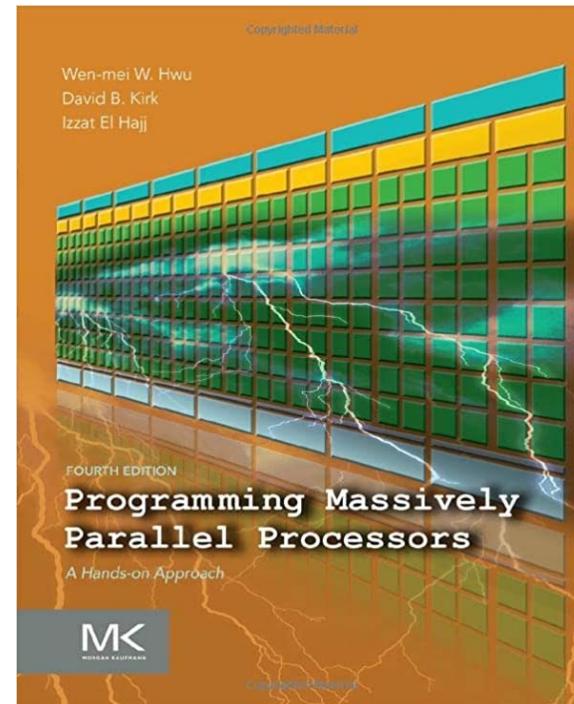


- Programming Massively Parallel Processors: A Hands-on Approach, 4<sup>th</sup> ed.
- CUDA by Example: An Introduction to General-Purpose GPU Programming, Jason Sanders, Edward Kandrot

[Online through KAUST Library](#)



[Online through KAUST Library](#)



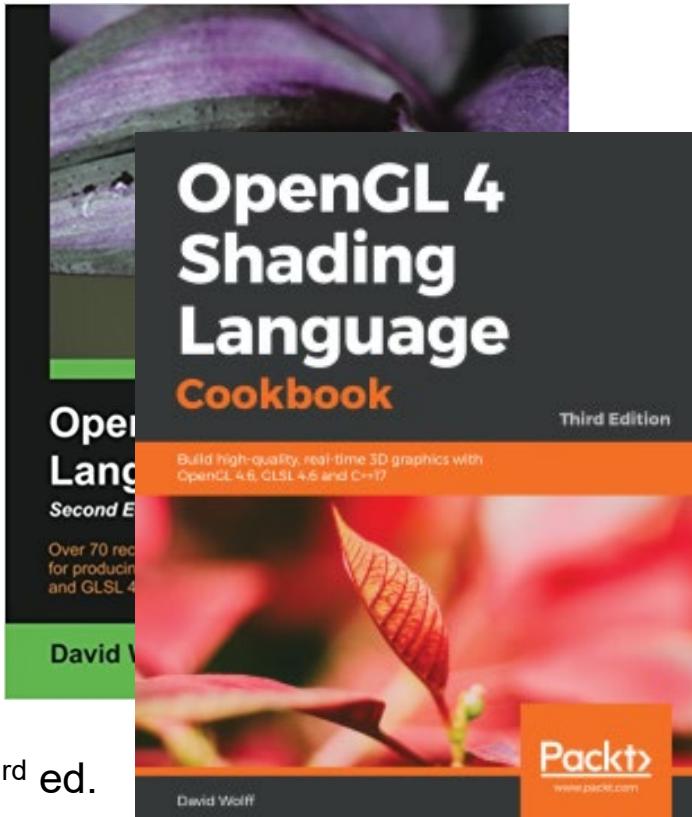
4<sup>th</sup> ed.

# Resources (2) – Graphics (OpenGL)

## Textbooks

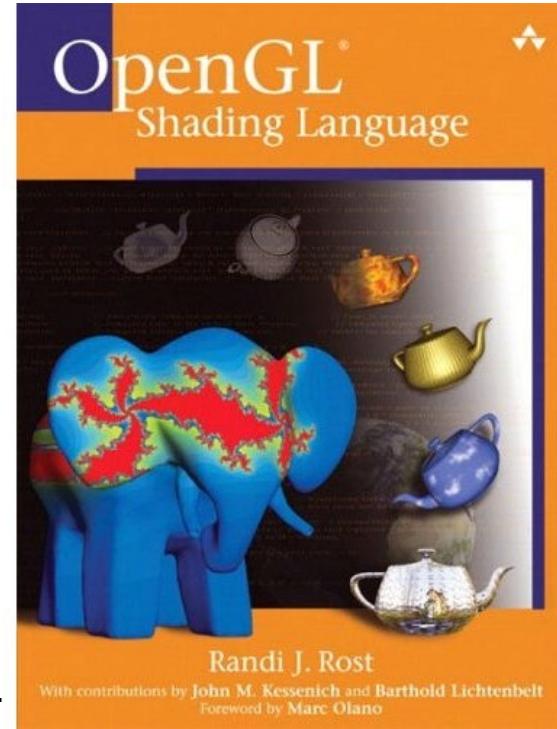


- OpenGL 4 Shading Language Cookbook, 2<sup>nd</sup> or 3<sup>rd</sup> ed.
- OpenGL Shading Language (orange book)



2<sup>nd</sup> ed.

3<sup>rd</sup> ed.



3<sup>rd</sup> ed.

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outdated in several aspects (no geometry shaders)  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

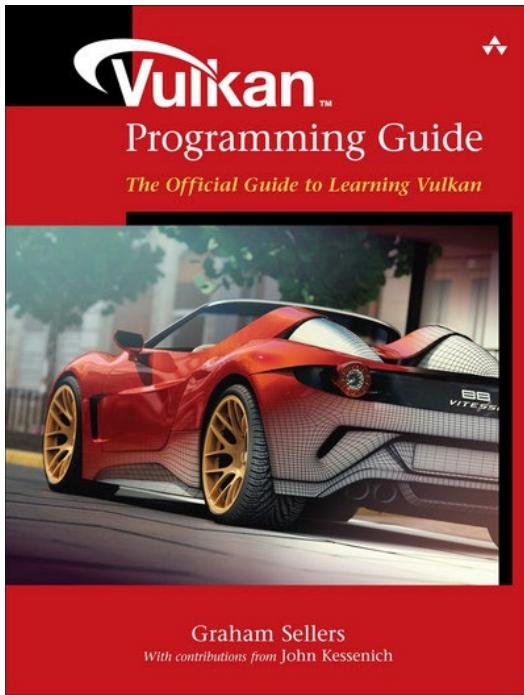
# Resources (2) – Graphics (Vulkan)

## Textbooks

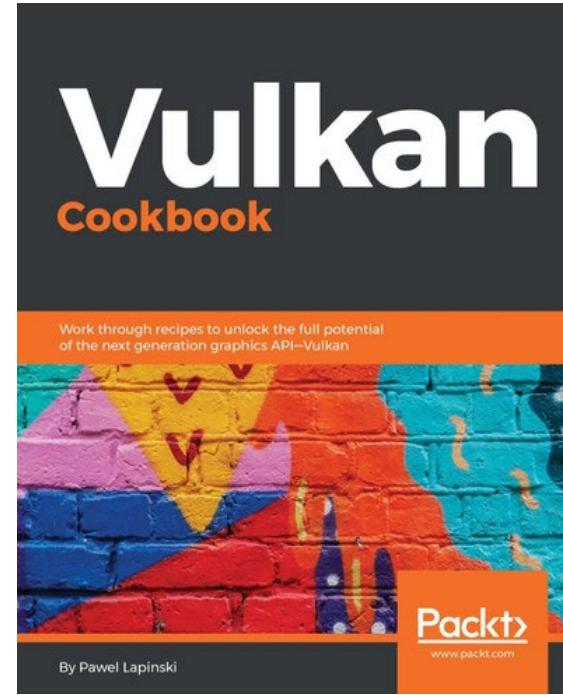


- Vulkan Programming Guide (2016)
- Vulkan Cookbook (2017)

[Online through KAUST Library](#)



[Online through KAUST Library](#)



# Resources (3) – Graphics – Reference

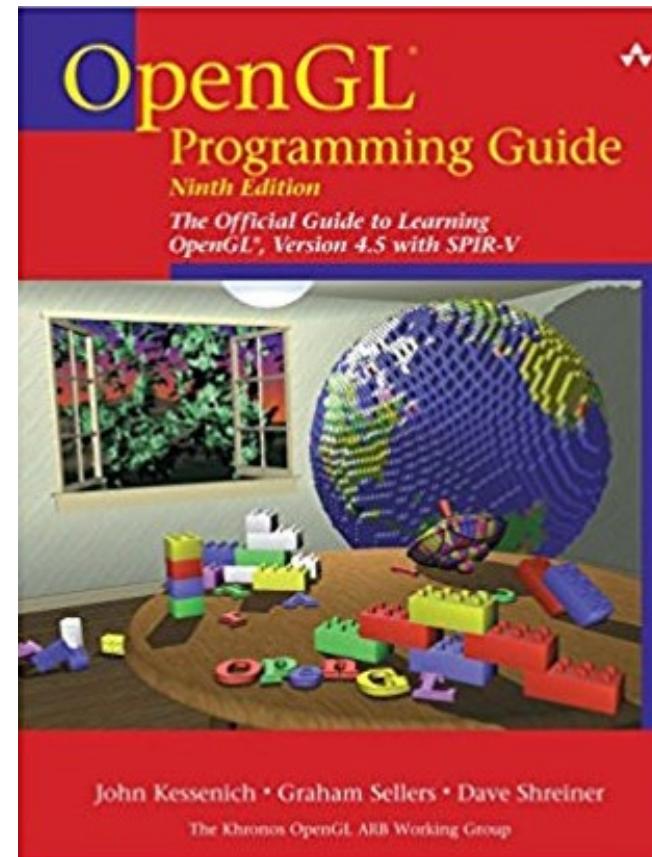


## OpenGL Programming Guide (red book)

<http://www.opengl-redbook.com/>

Computer graphics and OpenGL

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

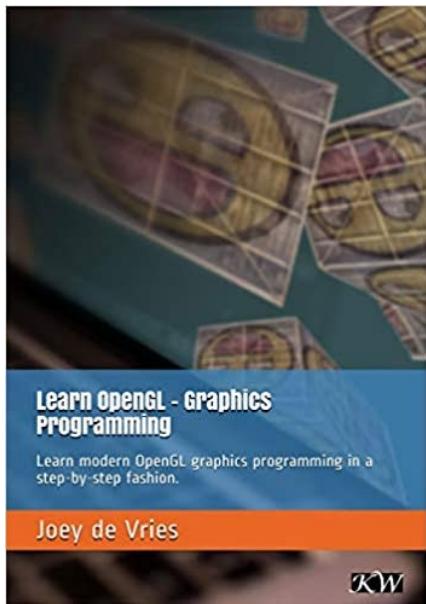


Available in the KAUST library  
and also electronically

# 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](https://learnopengl.com/book/book_pdf.pdf)

## YouTube lecture series on Vulkan:



**Introduction to Computer Graphics**  
186.832, 2021W, 3.0 ECTS

*Vulkan Lecture Series, Episode 1:  
Vulkan Essentials*

Johannes Unterguggenberger

Institute of Visual Computing & Human-Centered Technology  
TU Wien, Austria

<https://youtu.be/tLwbj9qys18>



# Resources (5) – Official Websites and Others



[https://vccvisualization.org/CS380\\_GPU\\_and\\_GPGPU\\_Programming/](https://vccvisualization.org/CS380_GPU_and_GPGPU_Programming/)

- OpenGL (4.6): [www.opengl.org](http://www.opengl.org)  
[www.khronos.org/files/opengl46-quick-reference-card.pdf](http://www.khronos.org/files/opengl46-quick-reference-card.pdf)
- CUDA (13.0): [developer.nvidia.com/cuda-toolkit/](http://developer.nvidia.com/cuda-toolkit/)
- Vulkan (1.4): [www.vulkan.org](http://www.vulkan.org)
- OpenCL (3.0): [www.khronos.org/opencl/](http://www.khronos.org/opencl/)

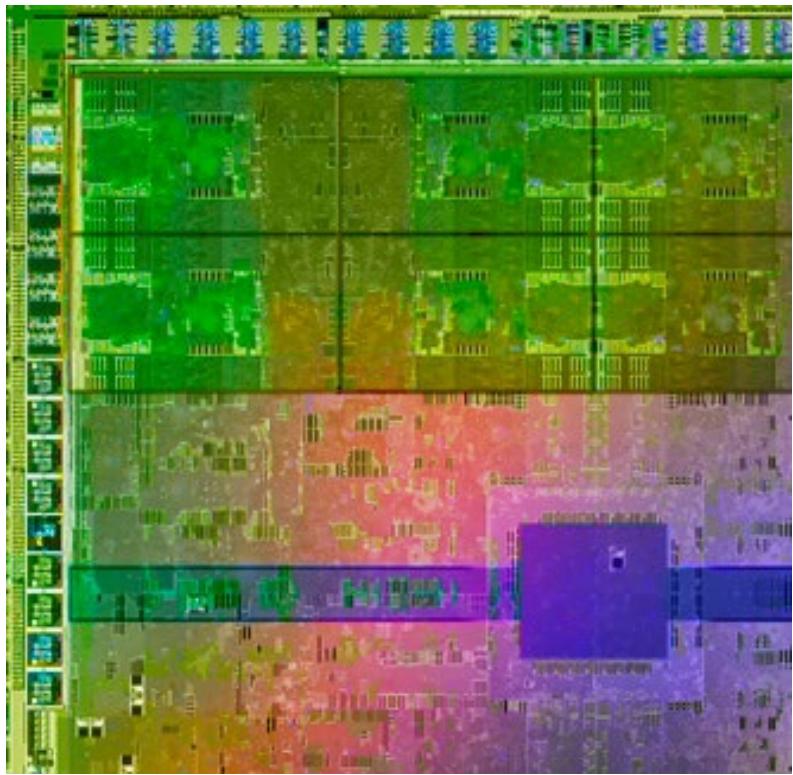
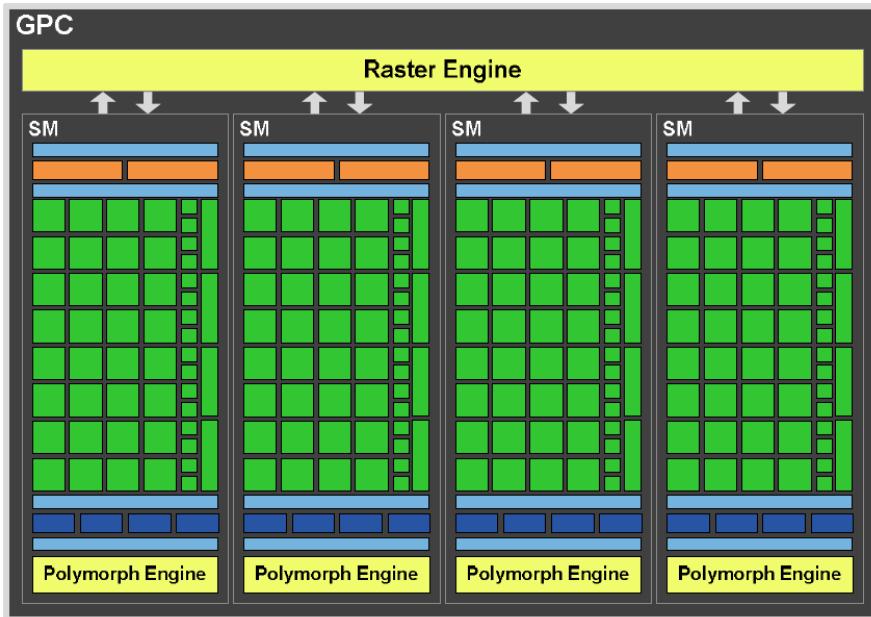
Very nice resources for techniques, algorithms and data structures:

- *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, ...)  
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]: 2020

- GA100, GA102, GA104, GA106, ...  
(A100, RTX 3070, 3080, 3090 (Ti), RTX A6000, ...)

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

- GH100, AD102, AD103, AD104, ...  
(H100, L40, RTX 4080 (12/16 GB), 4090, RTX 6000, ...)

**Blackwell** [CC 10.0]: 2024

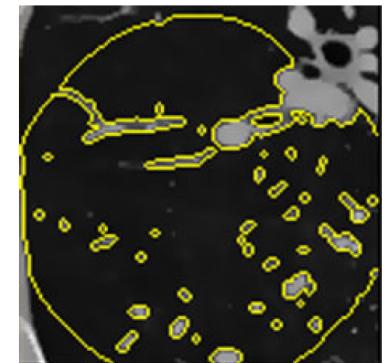
- GB200/GB202, ...  
(RTX 5080/5090, GB200 NVL72, HGX B100/200, ...?)

# 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

**SAXPY**  
 $\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \alpha \cdot \text{---} + \text{---} \end{array}$

**SGEMV**  
 $\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \cdot \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array}$

linear algebra

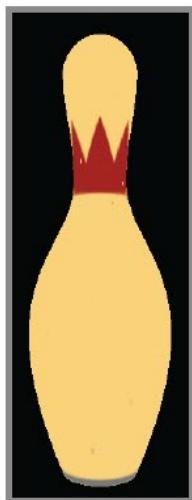
# Syllabus (3)

## GPU Graphics (~November)

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



## Semester project presentations



# GPGPU Examples

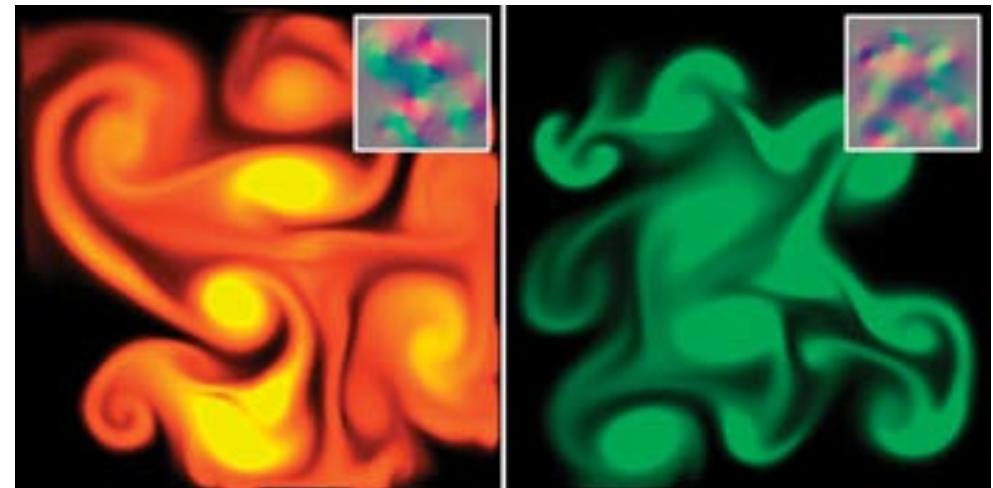
Peter Rautek, KAUST

Markus Hadwiger, 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



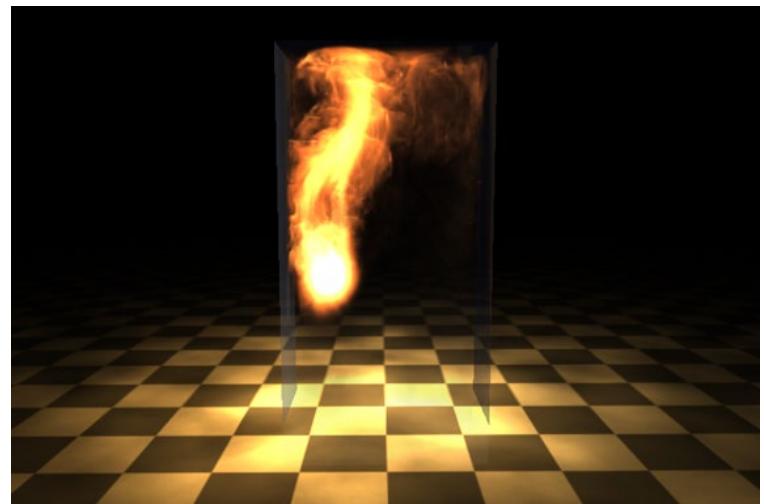
Courtesy Mark Harris





# 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



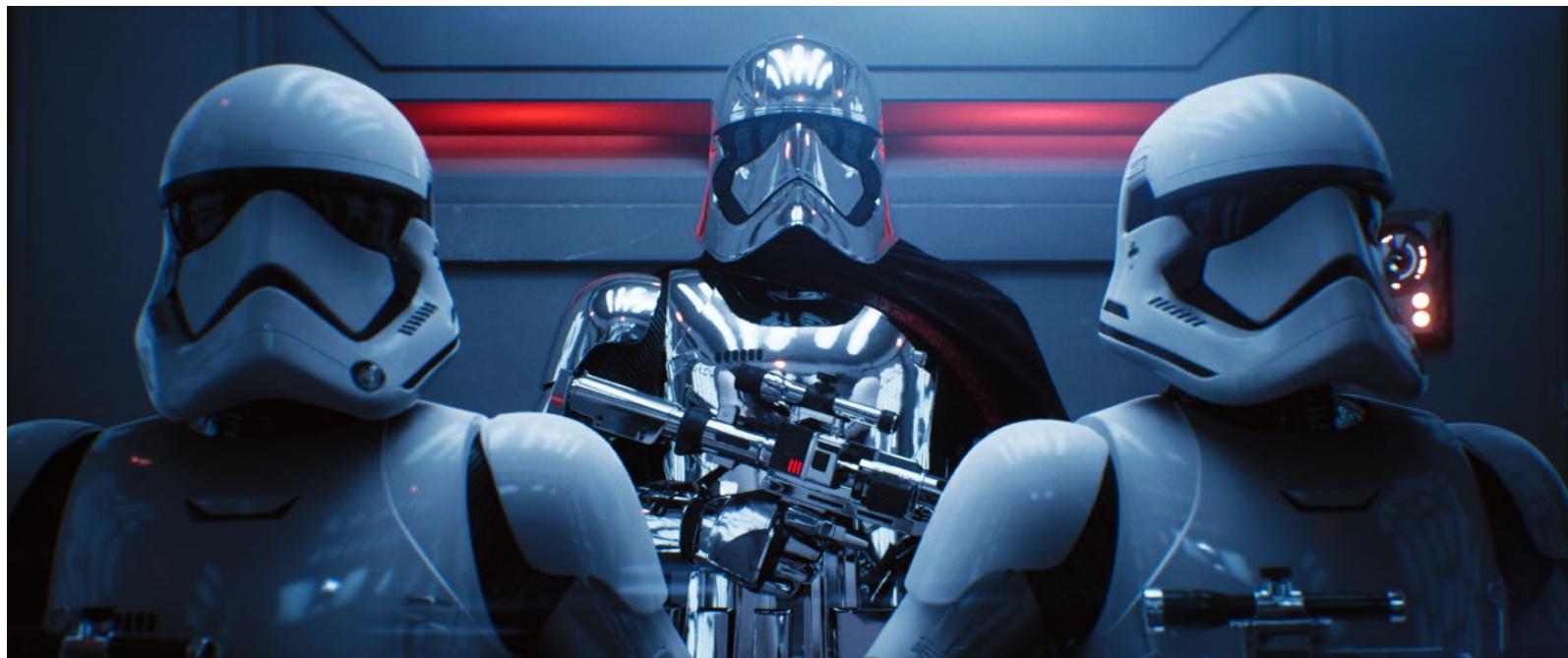




# Example: Ray Tracing

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

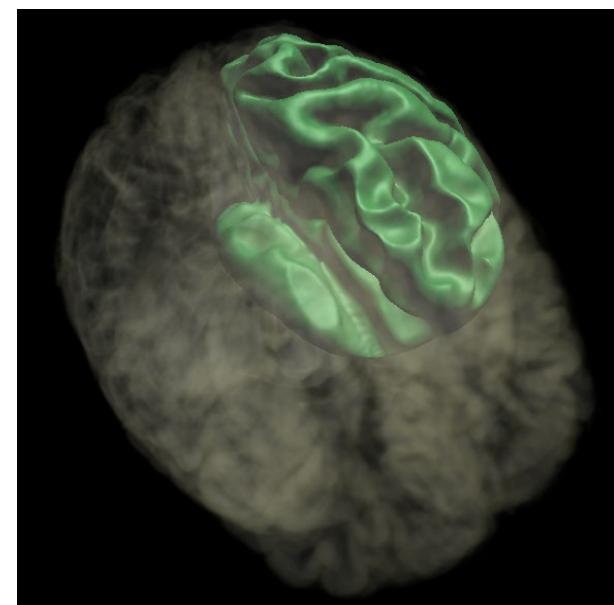
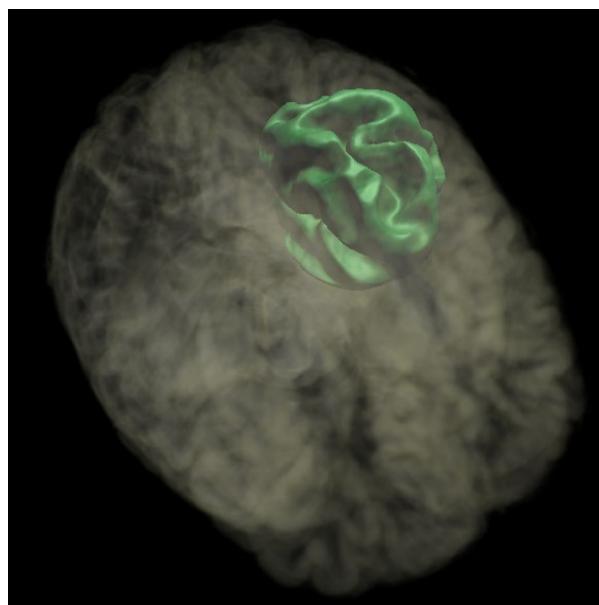
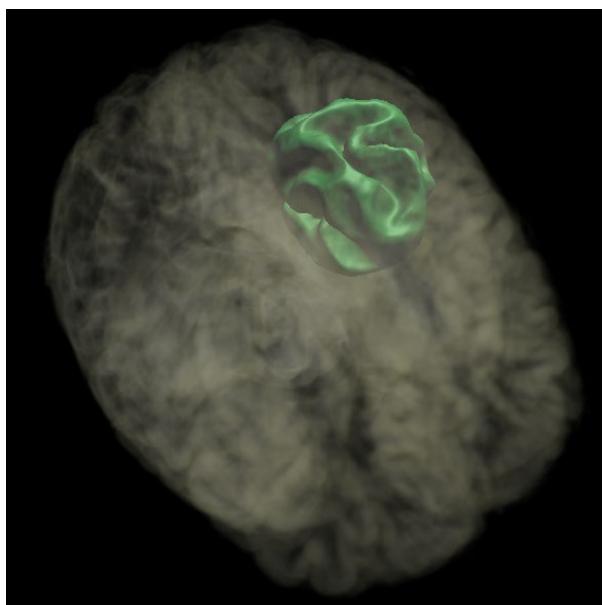


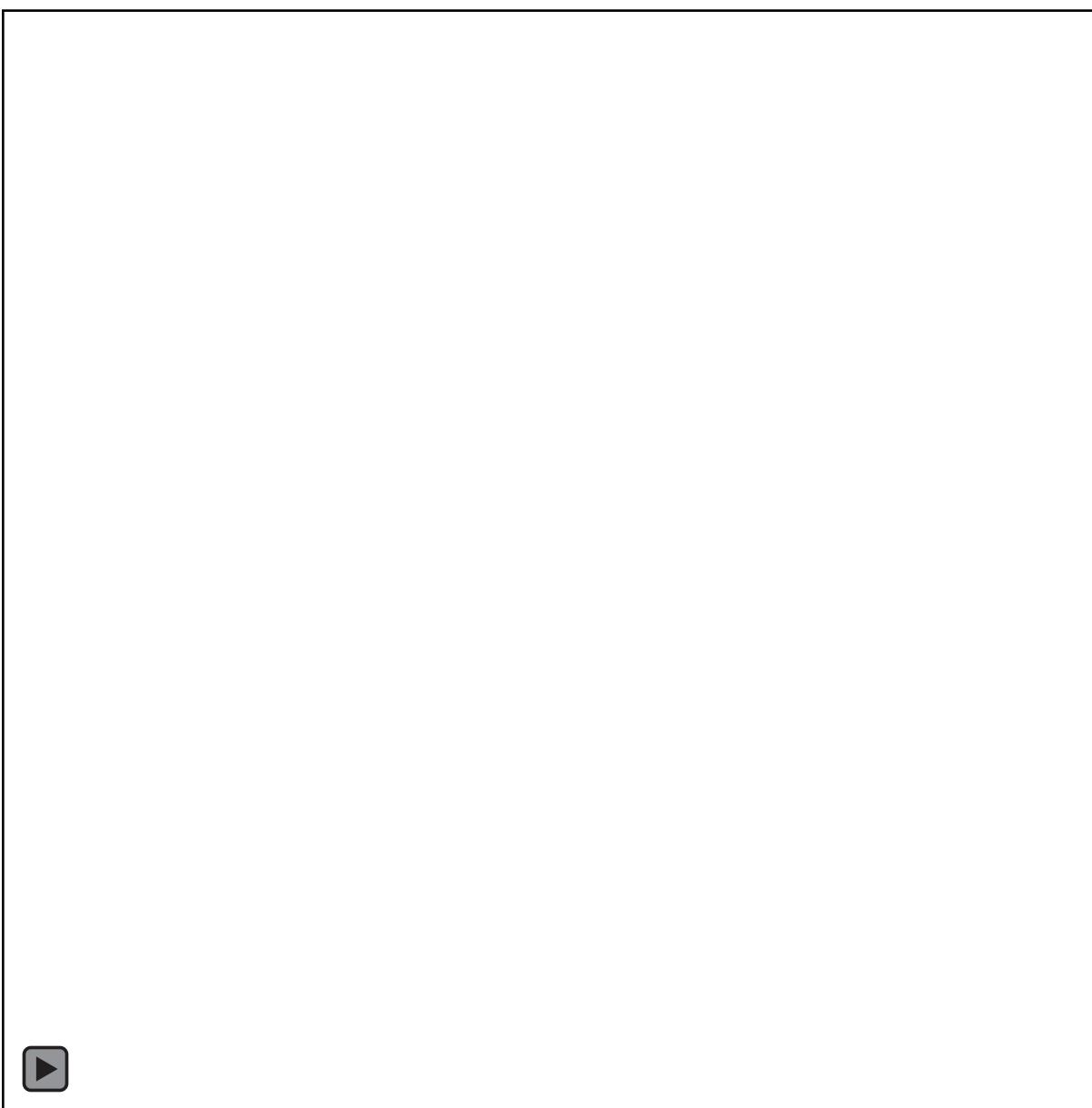
Unreal Engine 4 (2018, [youtube](#)), Nvidia RTX (2021, [youtube](#)), Unreal Engine 5 (2025, [youtube](#)) 23



# 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



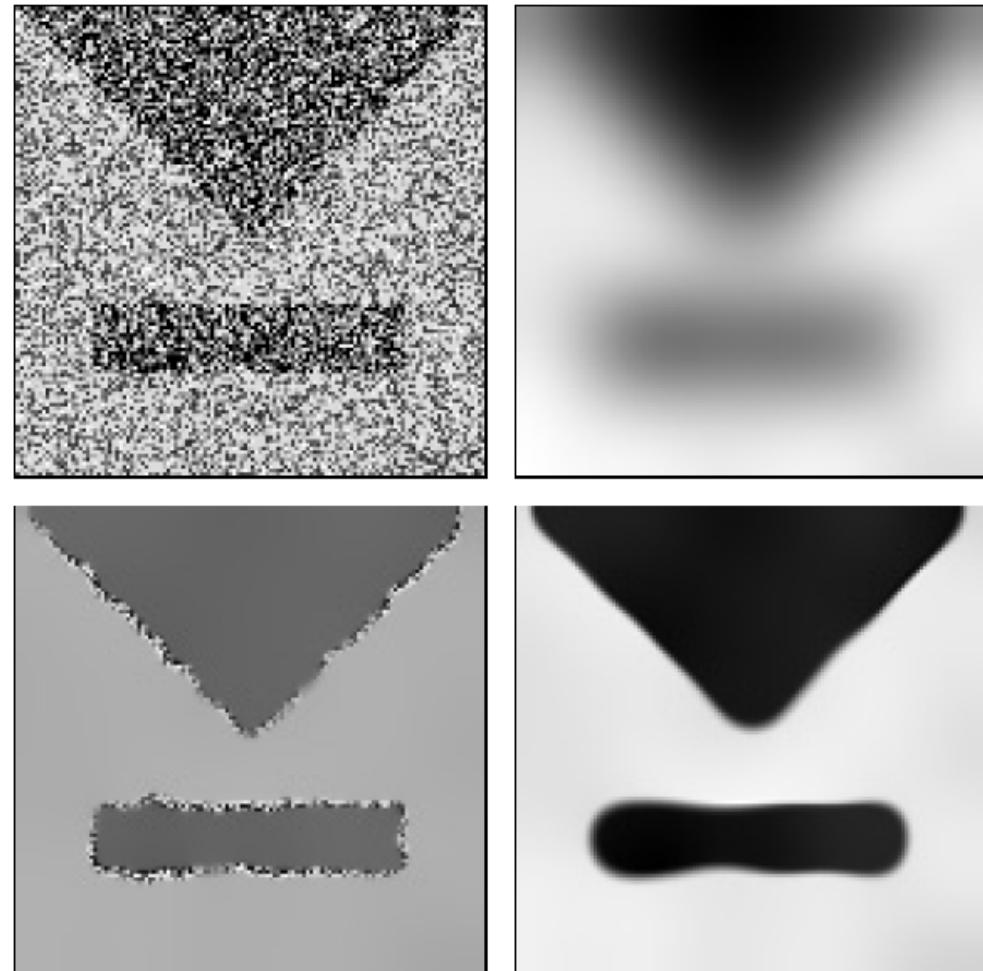




# Example: Diffusion Filtering

## De-noising

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

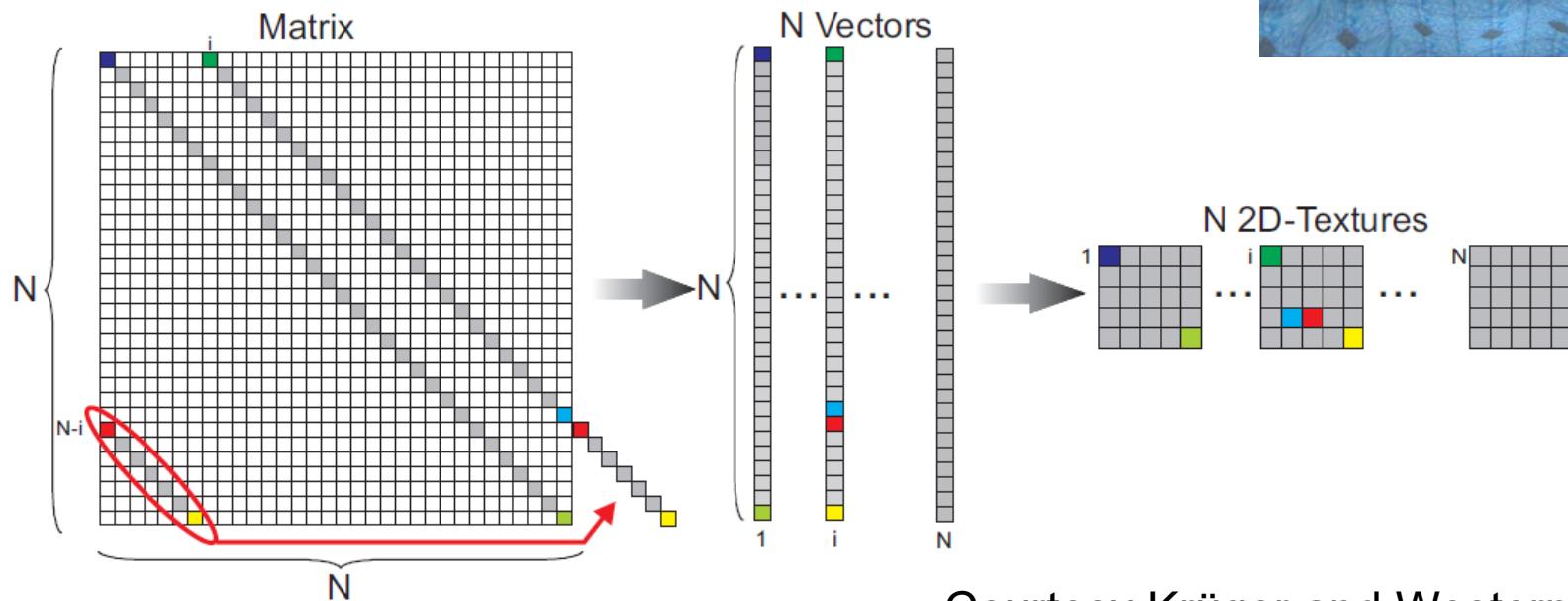
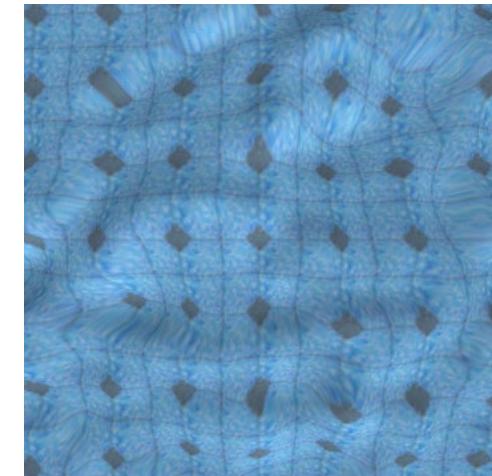




# Example: Linear Algebra Operators

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  
27

# 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

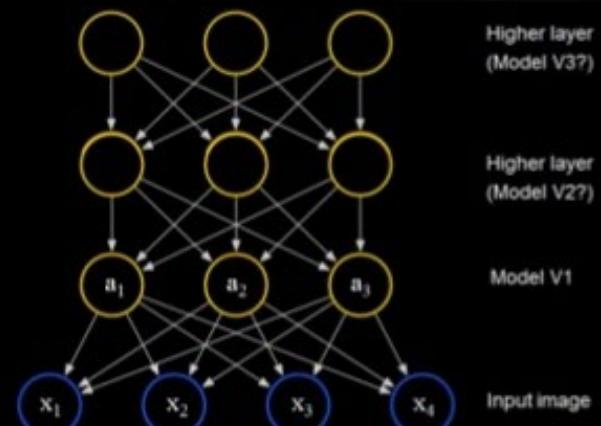
- TensorFlow,  
PyTorch,  
Caffe,  
...

## WHY ARE GPUs GOOD FOR DEEP LEARNING?

	Neural Networks	GPUs
Inherently Parallel	✓	✓
Matrix Operations	✓	✓
FLOPS	✓	✓
Bandwidth	✓	✓

*GPUs deliver --*

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



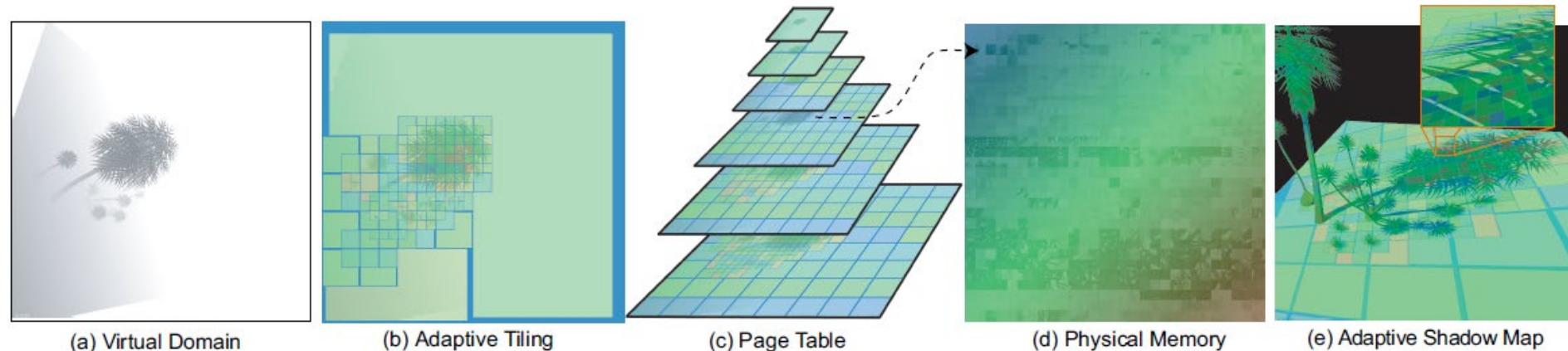
[Lee, Ranganath & Ng, 2007]



# Example: GPU Data Structures

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

- “STL“ for GPUs
- Virtual memory management



Courtesy Lefohn et al.

# Programming Assignments - Organization

Peter Rautek, KAUST

Markus Hadwiger, KAUST

# 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
- Xingdi Zhang (**xingdi.zhang@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](mailto:peter.rautek@kaust.edu.sa)

Xingdi: [xingdi.zhang@kaust.edu.sa](mailto:xingdi.zhang@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](mailto: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 13.0 (<https://developer.nvidia.com/cuda-downloads>)
- git (<https://git-scm.com/downloads>)
- Fork the CS 380 repository  
(<https://bitbucket.org/rautek/cs380-2024/src/main>)
- Follow the readme and start coding

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



```
C:\Users\rautek\Documents\GitHub\CS380\src>nvcc -V
nvcc: NVIDIA (R) Cuda compiler version 13.0.60 (git: 21d79436e0b8)
Copyright (c) 2022, NVIDIA Corporation.  All rights reserved.
https://github.com/NVIDIA/cuda-compiler-tools-13.0/releases
Detected 1 CUDA capable device(s)

Device #0: "GeForce GTX 1660"
  CUDA Driver Version / Runtime Version : 11.0 / 11.0
  CUDA Capabilities Mapped : 6.1
  Total Clock Rate (MHz) : 1389
  GPU Clock Speed : 1389 MHz
  Max Clock Speed : 1389 MHz
  Compute Mode : Default
  Max Dynamic Parallelism : 256
  Max Dimension Size (x,y,z) : 65536, 65536, 32<-4096, 65536, 65536
  Maximum Shared 2D Texture Size : 32768<-16384, 32768<-16384, 20480 bytes
  Maximum Shared 3D Texture Size : 32768<-16384, 32768<-16384, 20480 bytes
  Total amount of memory available per block : 49152
  Total amount of shared memory available per block : 23392
  Maximum number of threads per multiprocessor : 2048
  Maximum number of threads per block : 1024
  Max dimension size of a grid (x,y,z) : 65536, 65536, 32<-4096
  Dimension size of a thread block (x,y,z) : 1024, 1024, 16<-4096
  Concurrent kernel and kernel execution : Yes
  Host pointer copy : Yes
  Host pointer lock : Yes
  Support host pointer lock for memory migration : Yes
  Support host pointer lock for shared memory migration : Yes
  Support host pointer lock for texture memory migration : Yes
  Support host pointer lock for constant memory migration : Yes
  Support host pointer lock for uniform memory migration : Yes
  Support host pointer lock for atomic memory migration : Yes
  Device PCI bus ID / PCI location ID : 0
  Device PCI bus ID / PCI location ID : 0
  Device can use user CUDA context with serial (multithreaded) : Yes
  Device can use user CUDA context with serial (multithreaded) : Yes
Device0 = 0

C:\Users\rautek\Documents\GitHub\CS380\src>
```

# 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

```
\\10.68.74.73\10_gpgpu\CS380_2012_Assignment_1_Solution\CS380_2012_Assignment_1\bin\Release\

-> OpenGL Check Driver Supports and Information
GL Vendor : NVIDIA Corporation
GL Renderer : Quadro 6000/PCI/SSE2
GL Version : 4.1.0
GLEW Version : 1.7.0

3D Texture : Supported
1D Texture Array : Supported
2D Texture Array : Supported
2D Texture Size : 16384
3D Texture Size : 1048
Framebuffer Objects : Supported

Max Draw Buffers : 8
Max Tex Units Vert : 32
Max Tex Units Geom : 32
Max Tex Units Frag : 32
Max Vertex Attributes : 16
Max Varying Floats : 60

GLSL : Supported
GLSL Version : 4.10 NVIDIA via Cg compiler
GLSL Geom Shader <ARB> : Supported
GLSL Geom Shader <EXT> : Supported

-> CudaCheck There are 2 devices supporting CUDA
--> Device 1 Quadro 6000
CUDA Capability : 2.0
CUDA MP Count : 14
CUDA Cores : 448

Global Memory : 4.000 GB
Shared Memory : 48.00 KB
Registers / Block : 32768

Clock rate GPU : 1.147 GHz
Clock rate Memory : 1.149 GHz
Warp Size : 32

CUDA Threads / Block : 1024
CUDA Threads / Block : 1024 x 1024 x 64
CUDA Blocks / Grid : 65535 x 65535 x 65535

2D Texture Size : 65536 x 65535
3D Texture Size : 2048 x 2048 x 2048
CUDA Timeout : true

--> Device 2 Quadro 6000
CUDA Capability : 2.0
CUDA MP Count : 14
CUDA Cores : 448

Global Memory : 4.000 GB
Shared Memory : 48.00 KB
Registers / Block : 32768

Clock rate GPU : 1.147 GHz
Clock rate Memory : 1.149 GHz
Warp Size : 32

CUDA Threads / Block : 1024
CUDA Threads / Block : 1024 x 1024 x 64
CUDA Blocks / Grid : 65535 x 65535 x 65535

2D Texture Size : 65536 x 65535
3D Texture Size : 2048 x 2048 x 2048
CUDA Timeout : true

-> CudaCheck Driver Supports and Information
CUDA Driver Version : 4.0
CUDA Driver Version : 4.0
```

# 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 or friends!  
You have to understand what you program:  
your explanations during the presentations will be part of the grade!

# Programming Assignments: Schedule (tentative)



Assignment #1:

- **Querying the GPU (Graphics and Compute APIs)** due Sep 7

Assignment #2:

- **GPU Compute – Data Parallel Processing** due Sep 21

Assignment #3:

- **GPU Compute – Porting Sequential to Parallel Code** due Oct 5

Assignment #4:

- **Graphics – Rasterization Pipeline** due Oct 26

Assignment #5:

- **Graphics – Compute Shaders (SPH Simulation)** due Nov 16

# 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 14)
- Present semester project, event in final exams week: Dec 15 (tentative!)

# Reading Assignment #1 (until Sep 4)



## Read (required):

- Programming Mass. Parallel Proc. book, 4<sup>th</sup> ed., Chapter 1 (*Introduction*)
- Programming Mass. Parallel Proc. book, 2<sup>nd</sup> 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.