Interactive Data Visualization on GPUs

Rüdiger Westermann

Lehrstuhl für Computer Graphik und Visualisierung

Overview

- Data types and representations, the visualization pipeline
- Volume rendering
  - Texture based ray-casting
- Data reconstruction from particle samples
  - Resampling techniques

Literature

- C. Hansen, C. Johnson (Ed.): *The handbook of Visualization*, Academic Press
- H. Schumann, W. Müller: *Visualisierung - Grundlagen und allgemeine Methoden*, Springer-Verlag
- G.M. Nielson, H.Hagen, H.Müller: *Scientific Visualization*, IEEE Computer Society Press
- Richard S. Gallagher (Ed.): *Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis*, CRC Press
- R. A. Earnshaw, N. Wiseman (Eds.): *An Introductory Guide to Scientific Visualization*, Springer-Verlag
- K.W. Brodlie u.a. (Eds.): *Scientific Visualization - Techniques and Applications*, Springer-Verlag
- R&D Agenda for Visual Analytics: *Illuminating the path*, http://nvac.pnl.gov/agenda.stm

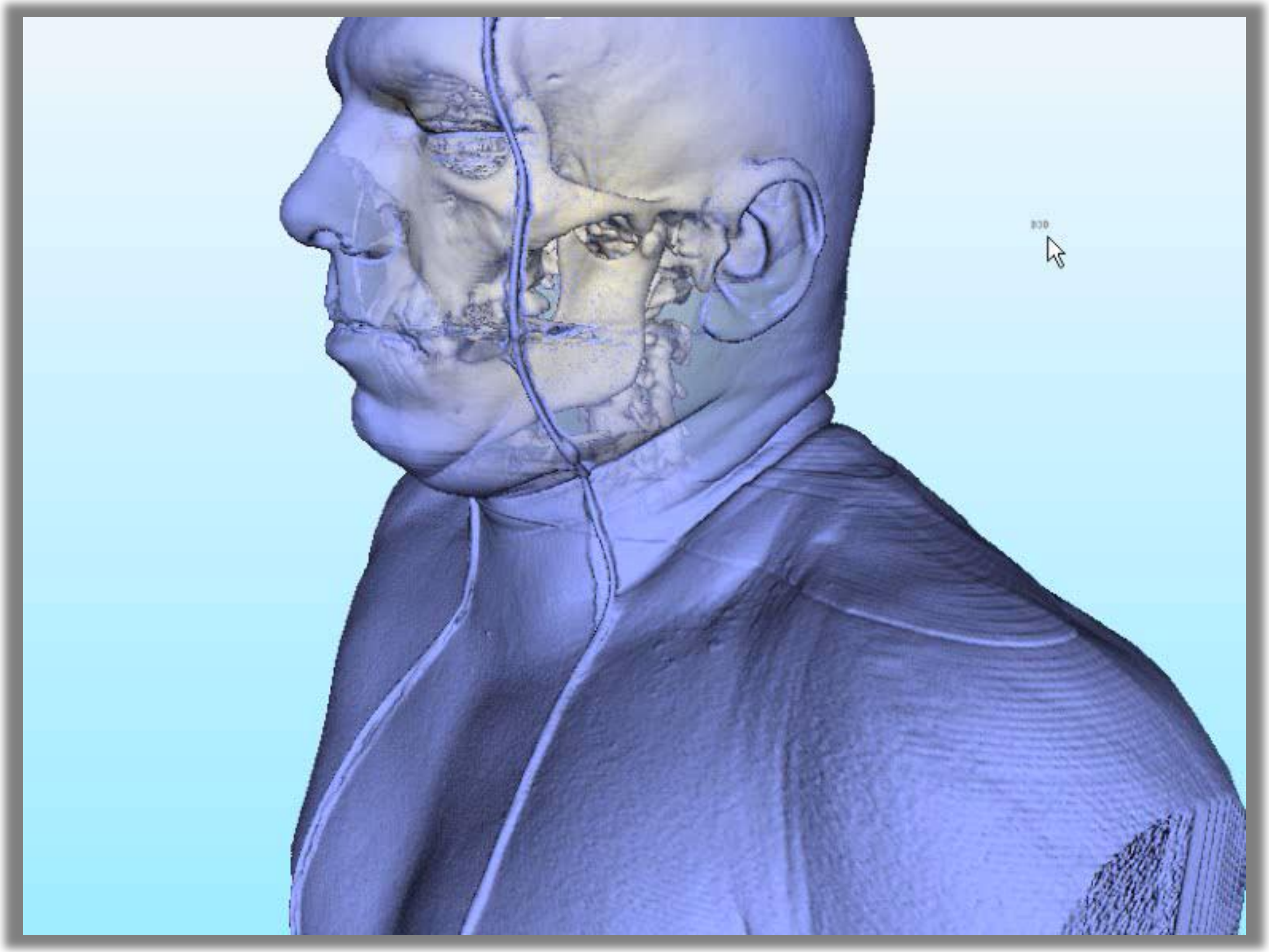
Definition

B. McCormick, T. DeFanti, and M. Brown:

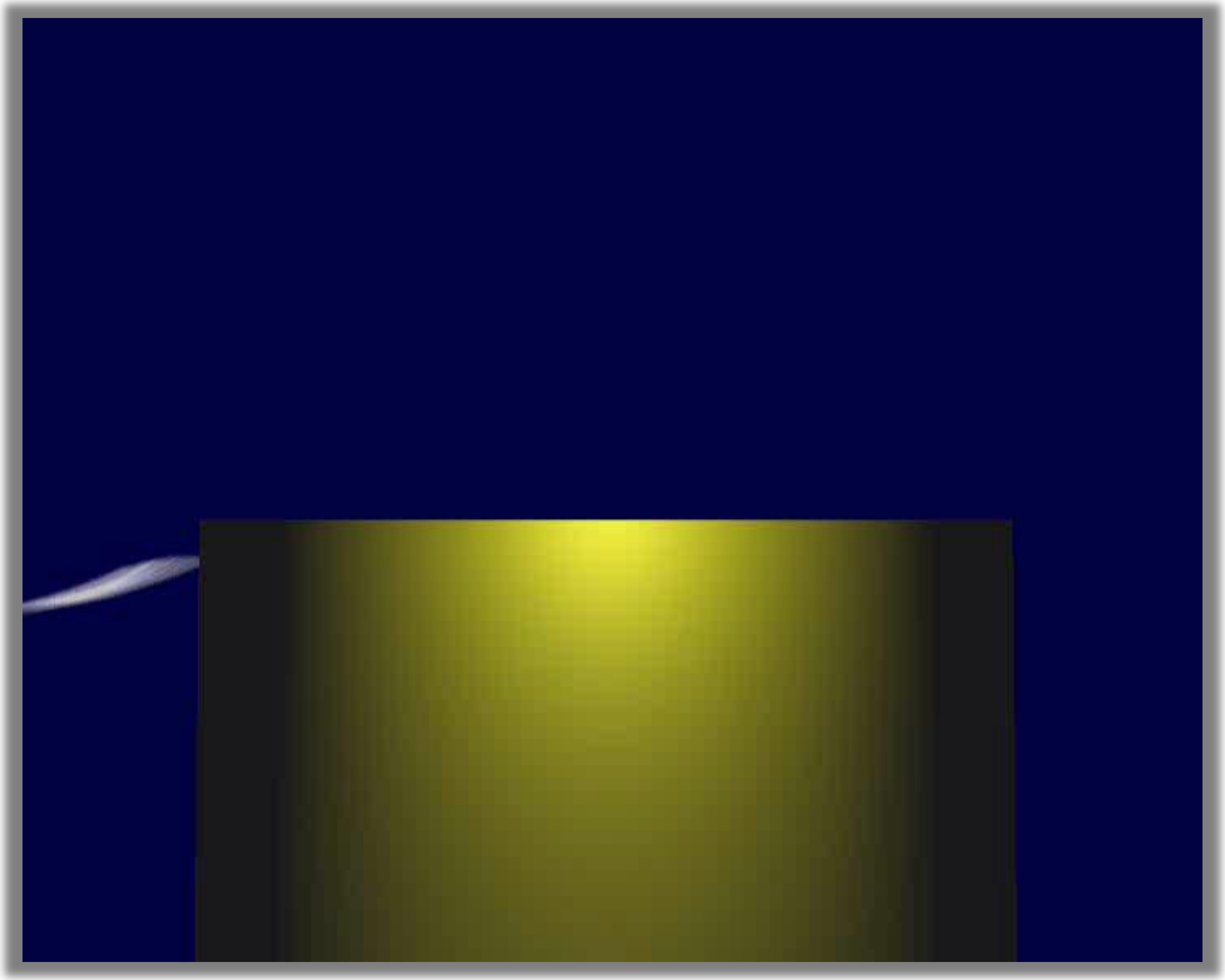
Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science.

McCormick, B.H., T.A. DeFanti, M.D. Brown, *Visualization in Scientific Computing*
Computer Graphics Vol. 21.6, November 1987

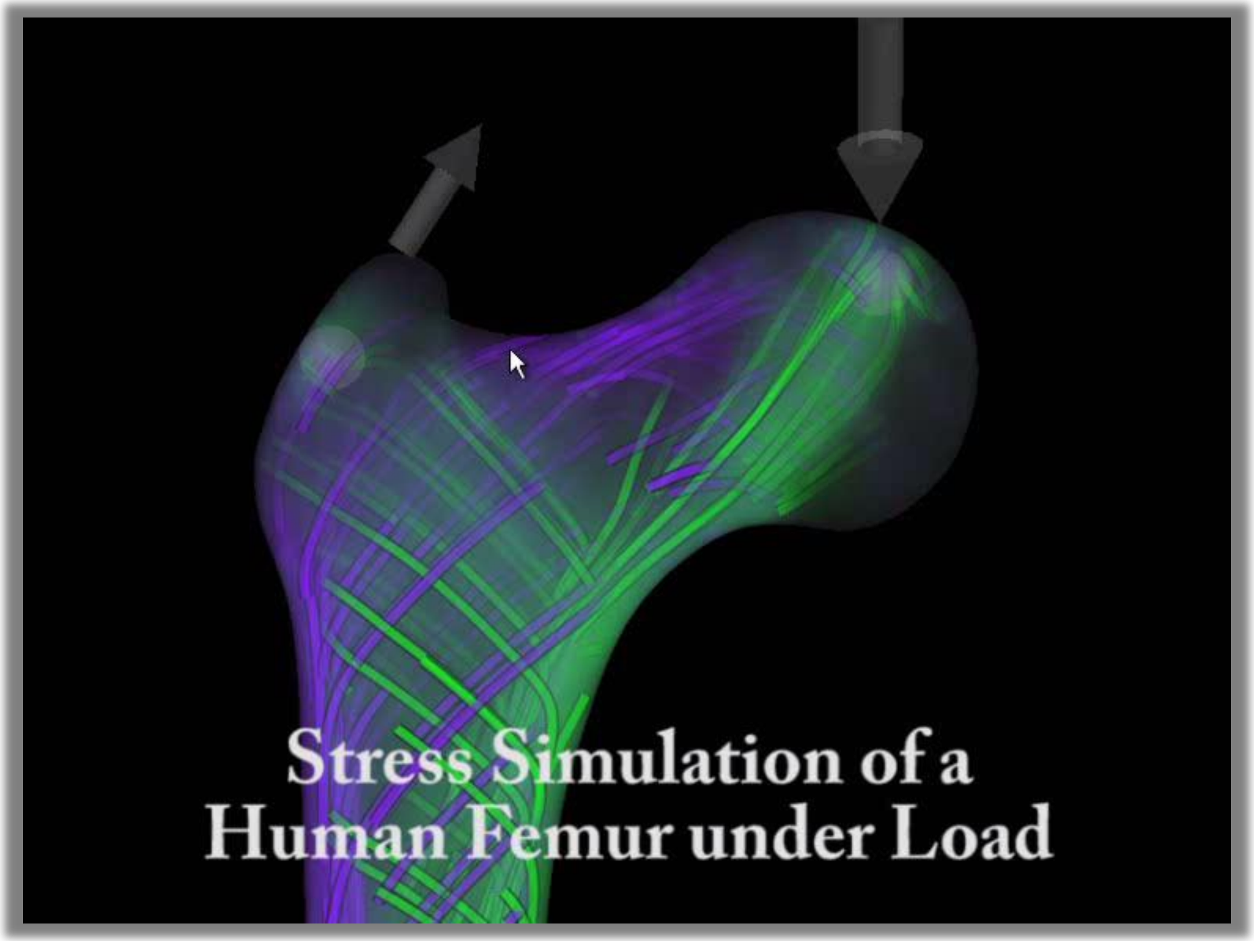
Data sources – medical imaging; volume rendering



Data sources – numerical simulation; particle tracing



Data sources – numerical simulation; tensor visualization



Data sources – remote sensing; terrain rendering

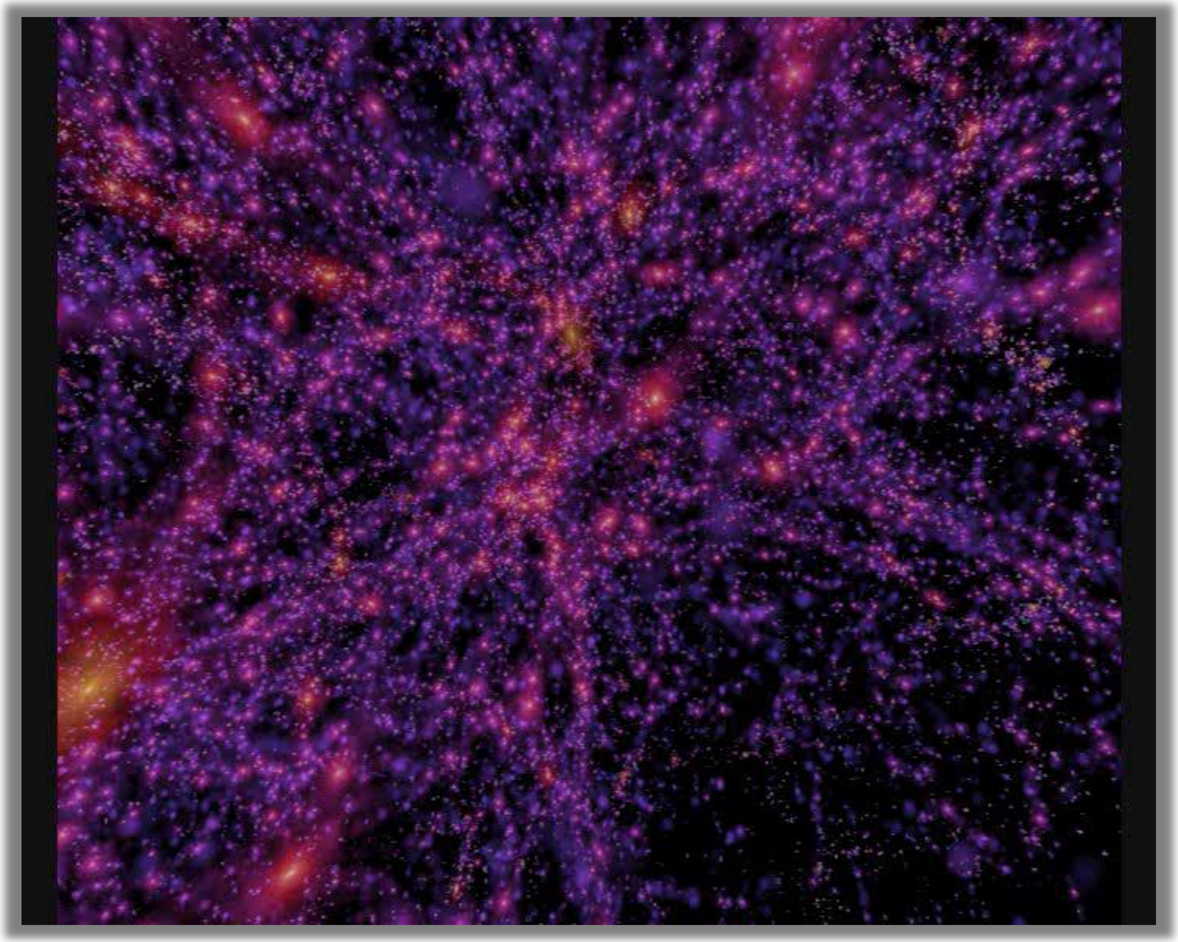
Vorarlberg

56km x 85km

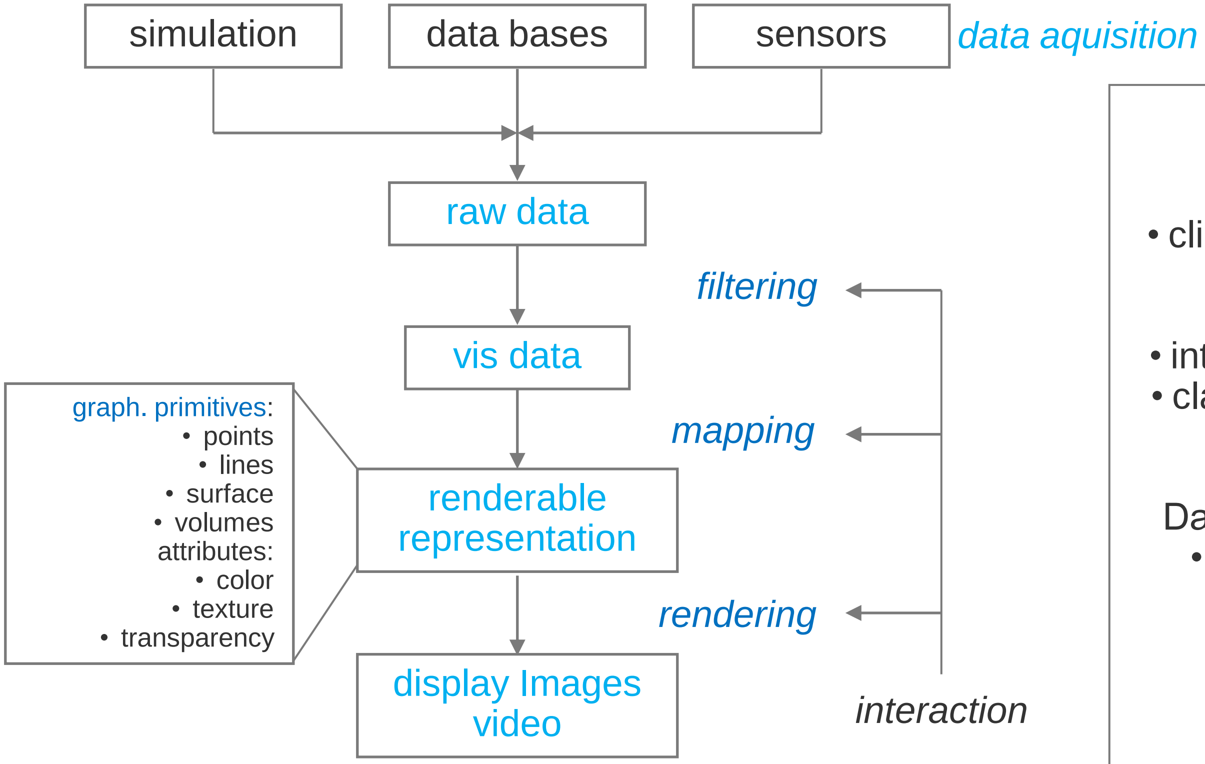
Geometry Spacing: 1m
Texture Spacing: 12.5cm
Raw Data Volume: 860GB

Geo Data © Land Vorarlberg

Data sources – numerical simulation; particle visualization



The visualization pipeline



simulation

data bases

sensors

data aquisition

raw data

filtering

vis data

mapping

renderable representation

rendering

display Images video

interaction

graph. primitives:
- points
- lines
- surface
- volumes

attributes:
- color
- texture
- transparency

filtering

data → data
- data format conversion
- clipping/cropping/denoising
- slicing
- resampling
- interpolation/approximation
- classification/segmentation

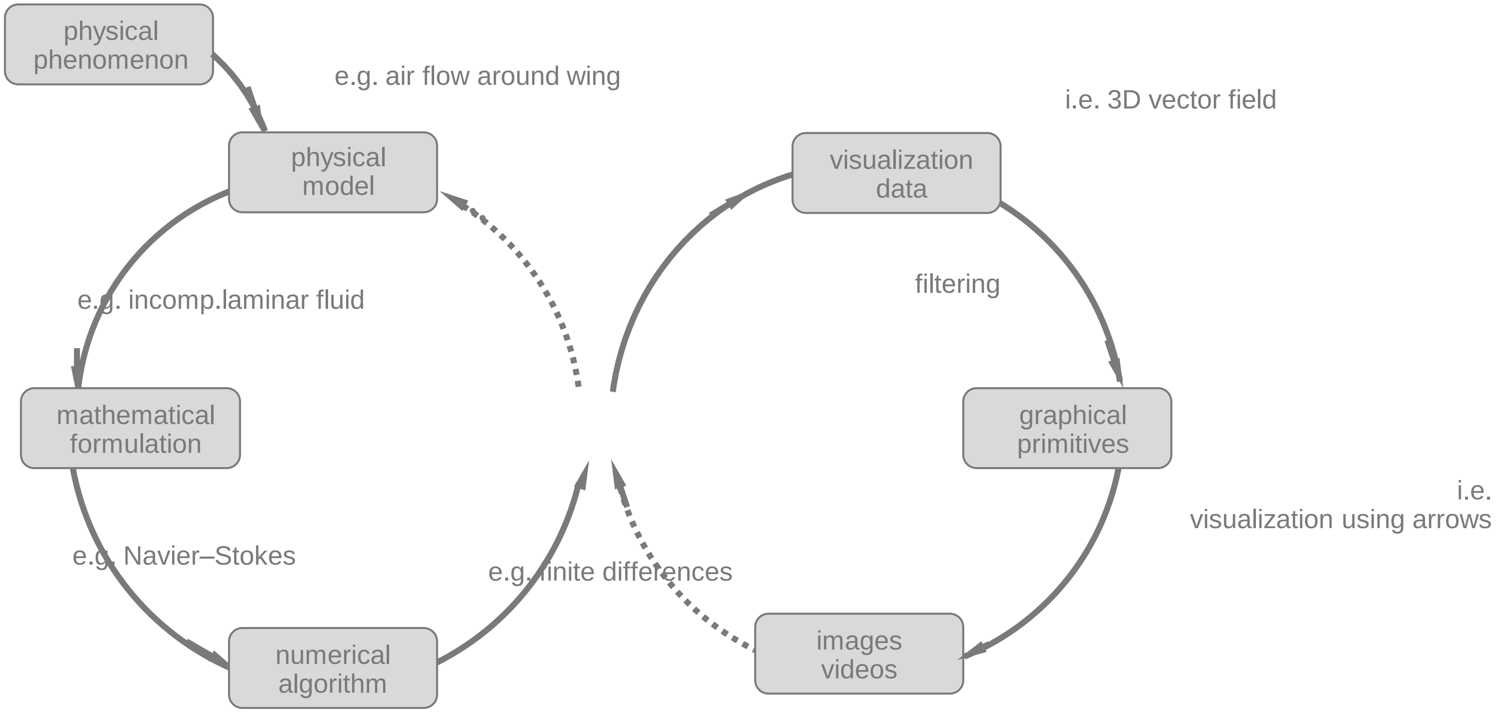
mapping

Data → graphical primitives
- scalar field → isosurface
- 2D field → height field
- vector field → vectors
- tensor field → glyphs
- 3D field → volume

rendering:
- geometry/images/volumes
- “realism“ – e.g. : shadows, lighting, shading

The visualization pipeline

- Visualization for computational steering



physical phenomenon

e.g. air flow around wing

physical model

e.g. incomp.laminar fluid

mathematical formulation

e.g. Navier–Stokes

numerical algorithm

e.g. finite differences

i.e. 3D vector field

visualization data

filtering

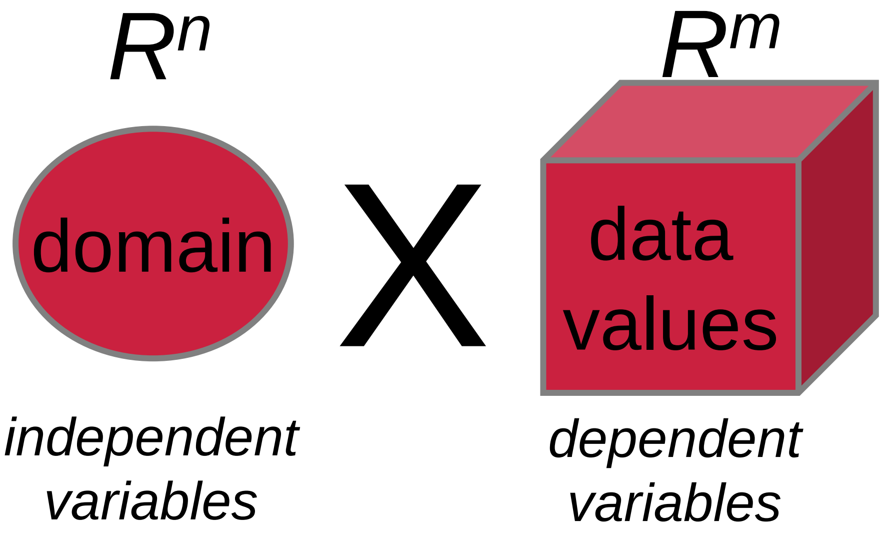
graphical primitives

i.e.
visualization using arrows

images videos

Data representation

- Classification of visualization techniques according to
  - Dimension of the domain of the problem (independent parameters)
  - Type and dimension of the data to be visualized (dependent parameters)



scientific data $\subseteq R^{n+m}$

Range of values

- Quantitative (measurable)
  - Metric scale – allows measure of distance
  - Discrete or continuous
- Qualitative
  - No metric scale
  - Ordinal (logical order relation)
  - Nominal (no order relation)
- Examples:
  - Eye color is nominal
    - Multi-valued: blue, green, brown, grey, pink, black
    - No metric scale
  - Survey: this is an interesting course
    - Ordinal: multi-valued with ordering
    - 1=Strongly disagree; 2=Disagree; 3=Neutral; 4=Agree; 5=Strongly agree

Data types

- Scalar data:
  given by a function $f(x_1,\ldots,x_n)\colon \mathbb{R}^n \rightarrow \mathbb{R}$ with n independent variables x_i
- Vector data
  represent direction and magnitude;
  given by a n-tupel $(f_1,\ldots,f_n)$ with $f_k = f_k(x_1,\ldots,x_n)$, *with* $1 \leq k \leq n$

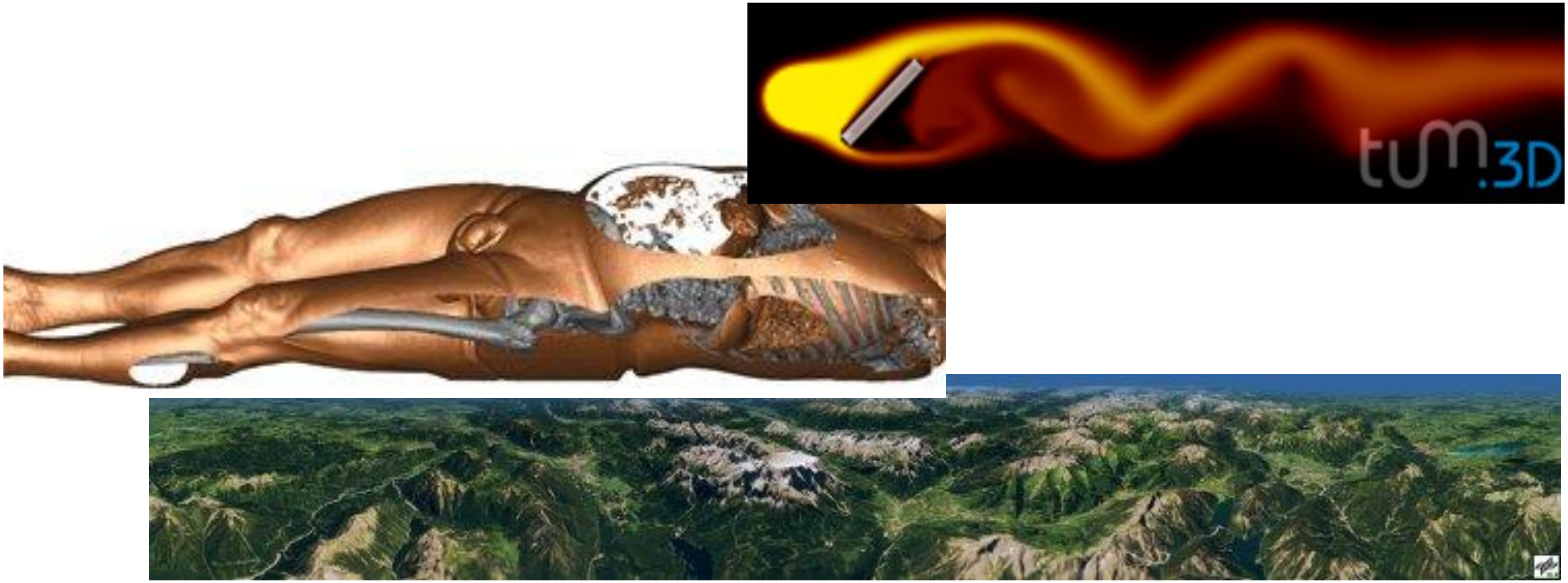
  Example: in a 2D vector field every sample represents a 2D vector (u,v) with $u = f(x,y)$ and $v = g(x,y)$.
- Tensor data
  a tensor of level k is given by $t_{i1,i2,\ldots,ik}(x_1,\ldots,x_n)$

Data representation

- Data in scientific visualization usually represents a continuous real object, e.g., an oscillating membrane, a velocity field around an obstacle, an human organ etc.
  - This object lives in an *n*-dimensional space - the domain
- Usually, the data is only given at a finite set of locations, or samples, in space and/or time
  - Remember numerical simulation techniques using grids or particles
- We call this a discrete structure, or a discrete representation of a continuous object

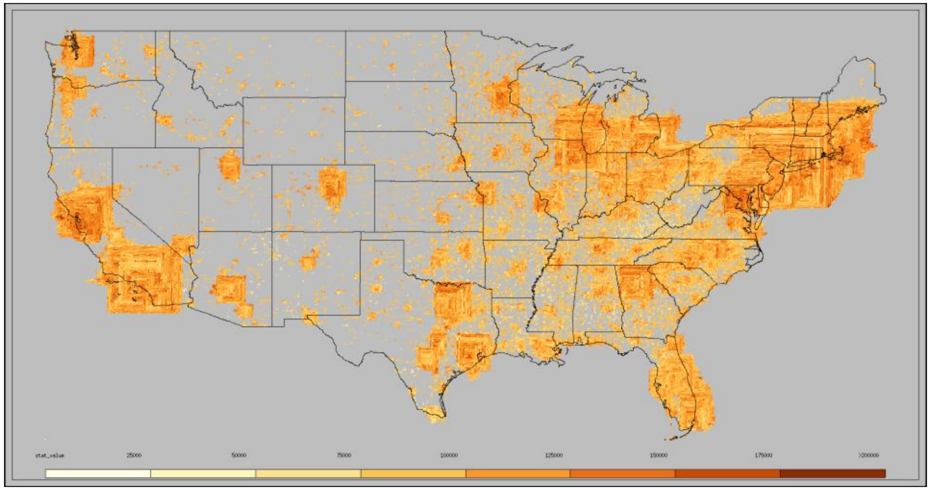
Data representation

- Discrete representations
  - In scientific visualization we usually deal with the reconstruction of a continuous real object from a given discrete representation



Data representation

- Discrete representations
  - There are examples, where the data to be visualized is discrete and abstract



Average income
in the US

  - The branch of visualization that deals with such data is called Information Visualization

Volume visualization

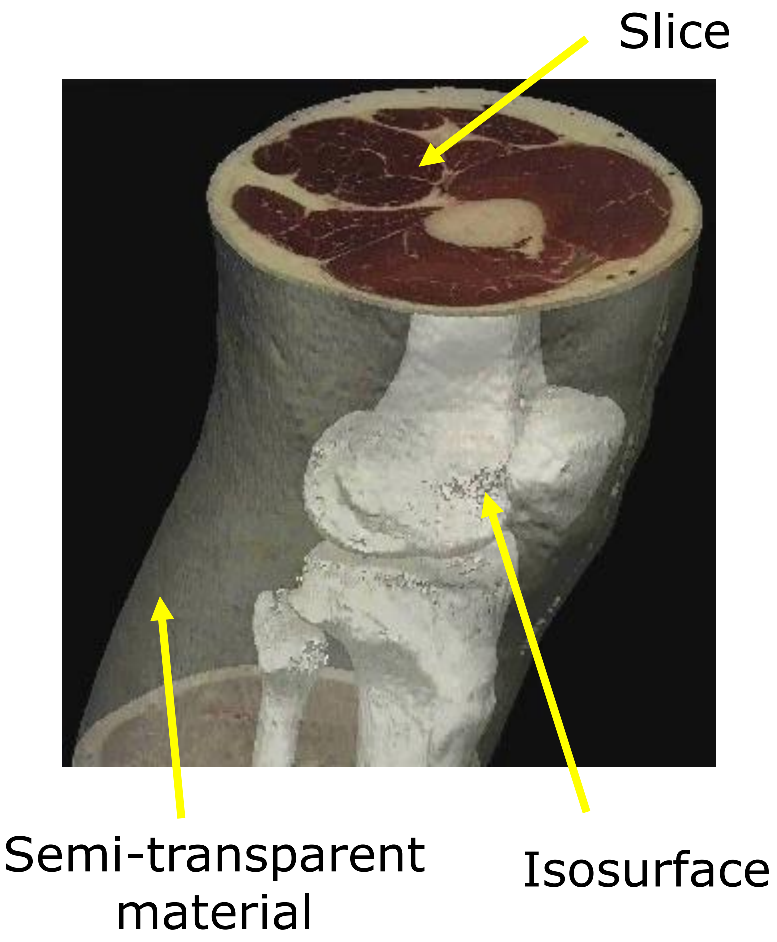
- Focus in this tutorial: rendering techniques for volumetric data sets
  - Some characteristics of volume data
    - Essential information in the interior
    - Cannot be described by a surface in general (e.g. fire, clouds, gaseous phenomena)

Volume visualization

- Volume rendering techniques
  - Techniques for 2D scalar fields
  - Indirect volume rendering techniques (e.g. surface fitting)
    - Convert/reduce volume data to an intermediate representation (surface representation), which can be rendered with traditional techniques – the MC-algorithm
  - Direct volume rendering techniques
    - Consider the data as a semi-transparent gel with physical properties and directly get a 3D representation of it

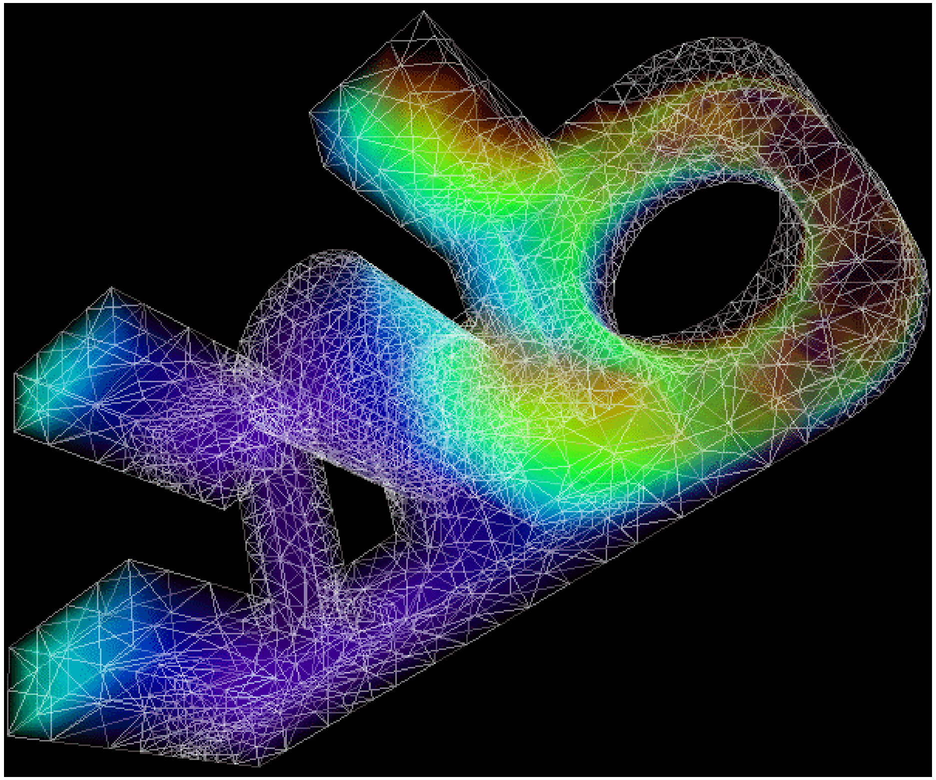
Volume visualization

- **Slicing**:
  Display the data values, mapped to colors, on a slice plane
- **Isosurfacing**:
  Generate opaque/semi-opaque surfaces
- **Direct volume rendering**:
  Volume material attenuates reflected or emitted light



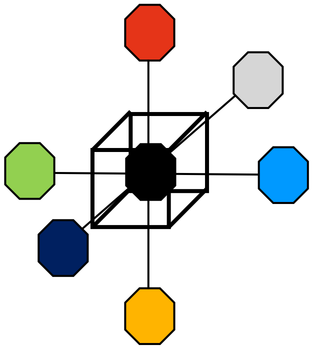
Volume visualization

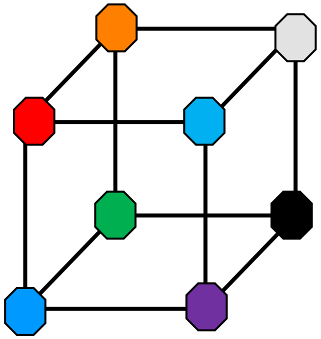
- Volumetric data sets
  - Consist of a 3D grid defining the shape
  - … and the data values given at the grid vertices

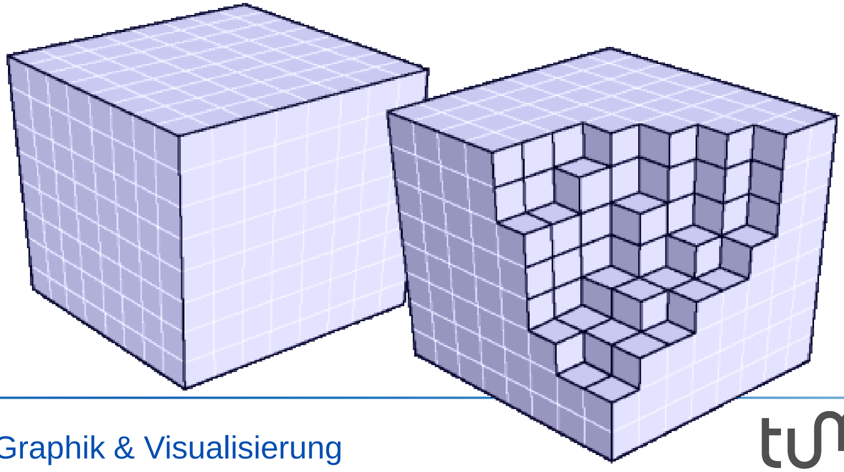


Volume visualization

- Volumetric data sets on Cartesian (uniform, orthogonal) grids
  - Data values are given at the grid vertices
  - These are called voxels (volume elements)
  - Data values are mapped to color and opacity (= 1.0 - Transparency) via a transfer function
  - “Adjacent” grid vertices make up a cell; use interpolation for data reconstruction in a cell

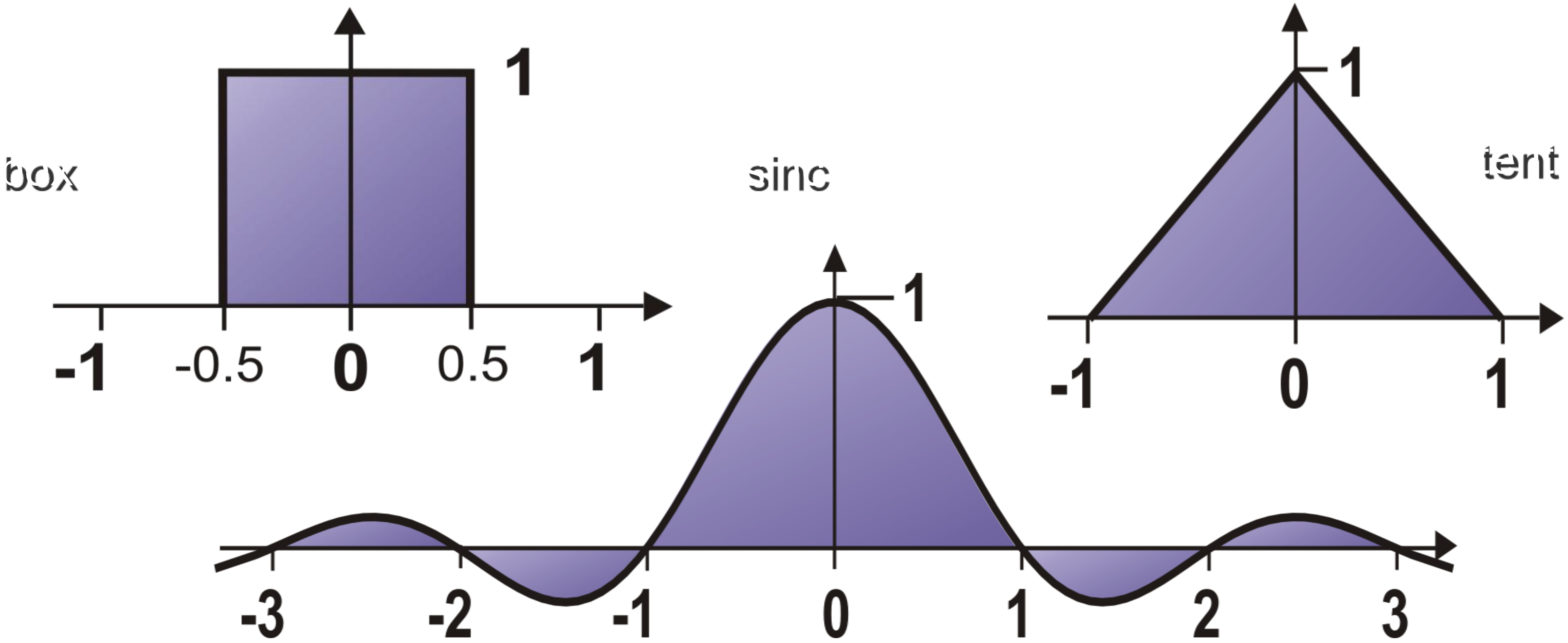






Volume visualization

- Reconstruction
  - Rectilinear 3D grid; scalar values
  - Convolution of samples with reconstruction filter (box, tent, …)



How do we determine what is seen along a ray through a volumetric body?

Physical model: emission and absorption, no scattering

Absorption along the ray segment s_0 - s

Initial intensity at s_0

$$I(s) = \boxed{I(s_0)}\boxed{e^{-\tau(s_0,s)}}$$

Extinction τ

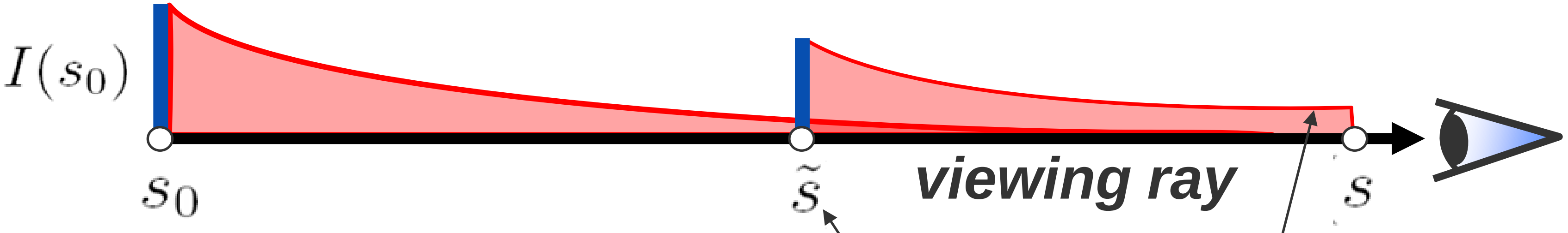
Absorption κ

Without absorption all the initial radiant energy would reach the point ***s.***

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)\, ds.$$

How do we determine what is seen along a ray through a volumetric body?

Physical model: emission and absorption, no scattering



One point $\tilde{s}$ along the viewing ray emits additional radiant energy

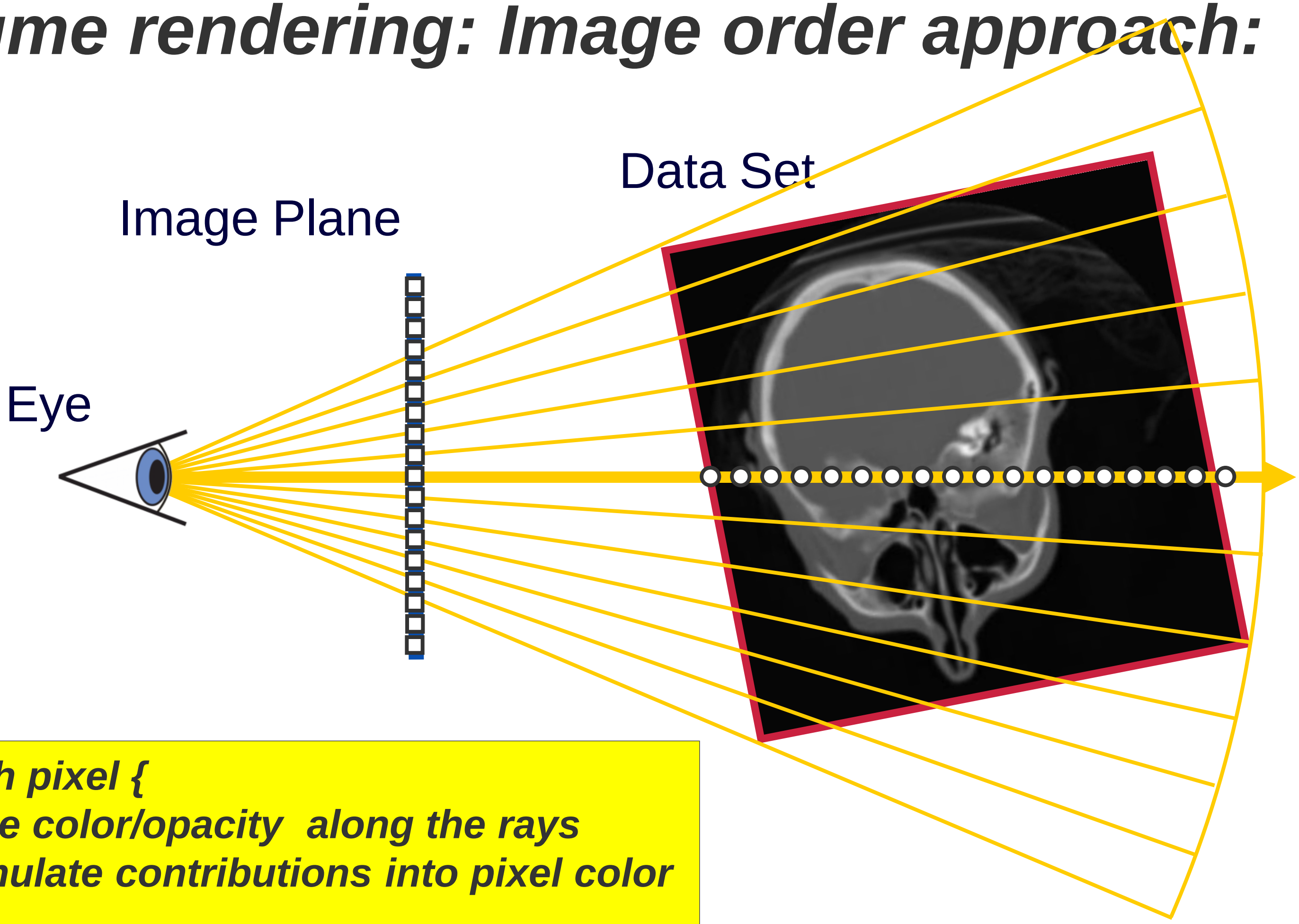
Every point $\tilde{s}$ along the viewing ray emits additional radiant energy.

Active emission
at point $\tilde{s}$

Absorption along the
distance $s - \tilde{s}$

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} \boxed{q(\tilde{s})}\,\boxed{e^{-\tau(\tilde{s},s)}}\, d\tilde{s}$$

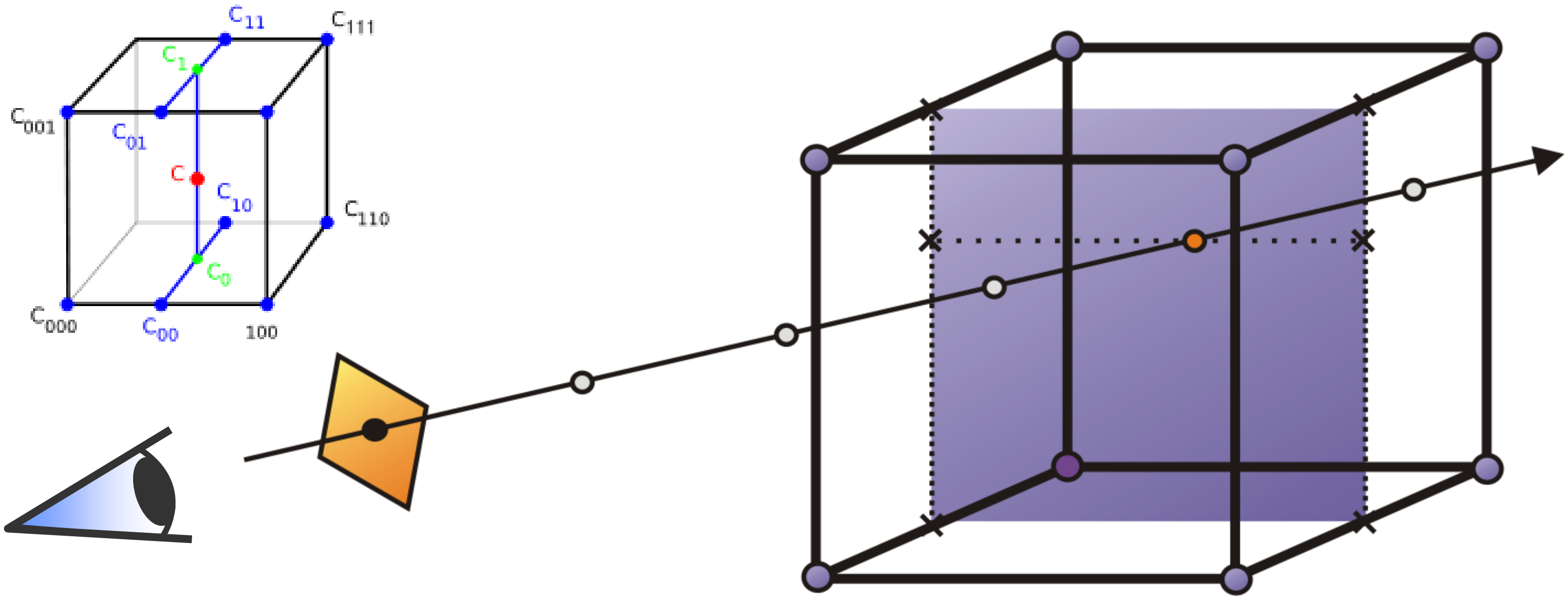
Volume rendering: Image order approach:



```
For each pixel {
   sample color/opacity  along the rays
   accumulate contributions into pixel color
}
```

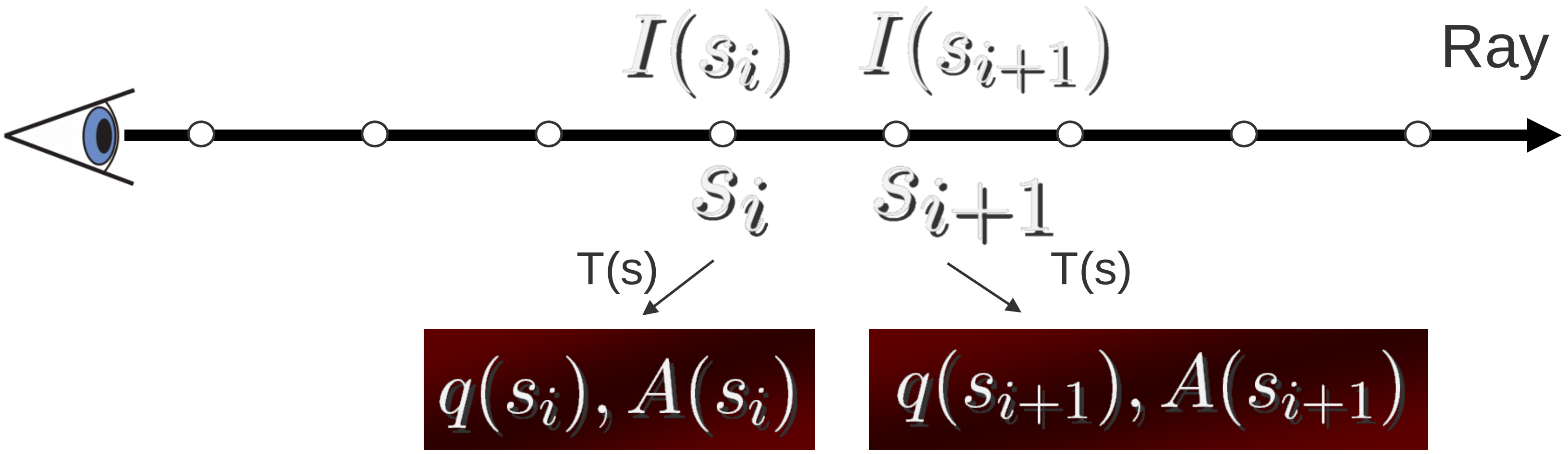
Volume ray casting

- Numerical approximation of the volume rendering integral
- Resample volume at equi-spaced intervals along the ray
- Tri-linear interpolation of color/opacity at grid vertices



Discrete Solution

Resample the scalar field at discrete locations along the viewing ray:



$I(s_i)$ $I(s_{i+1})$ Ray

s_i s_{i+1}

T(s) T(s)

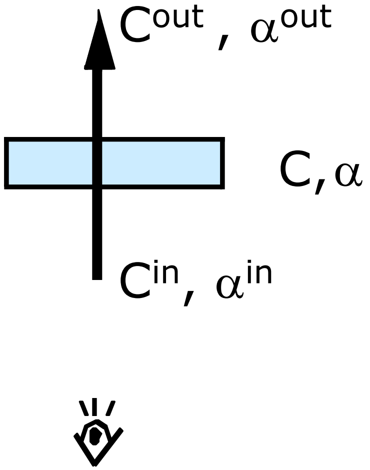
$q(s_i), A(s_i)$ $q(s_{i+1}), A(s_{i+1})$

Back-to-front Compositing with $\alpha = A(s_i)$

$$\begin{aligned} I(s_{i+1}) &= \alpha \cdot q(s_{i+1}) + (1-\alpha) I(s_i) \\ &= q(s_{i+1}) \text{ OVER } I(s_i) \end{aligned}$$

Volume ray casting

- Compositing of semi-transparent segments
  - Physical model: emissive gas with color C and opacity α
  - Front-to-back strategy
  - $C^{out} = C^{in} + (1 - \alpha^{in})\ \alpha C$
  - $\alpha^{out} = \alpha^{in} + (1 - \alpha^{in})\ \alpha$

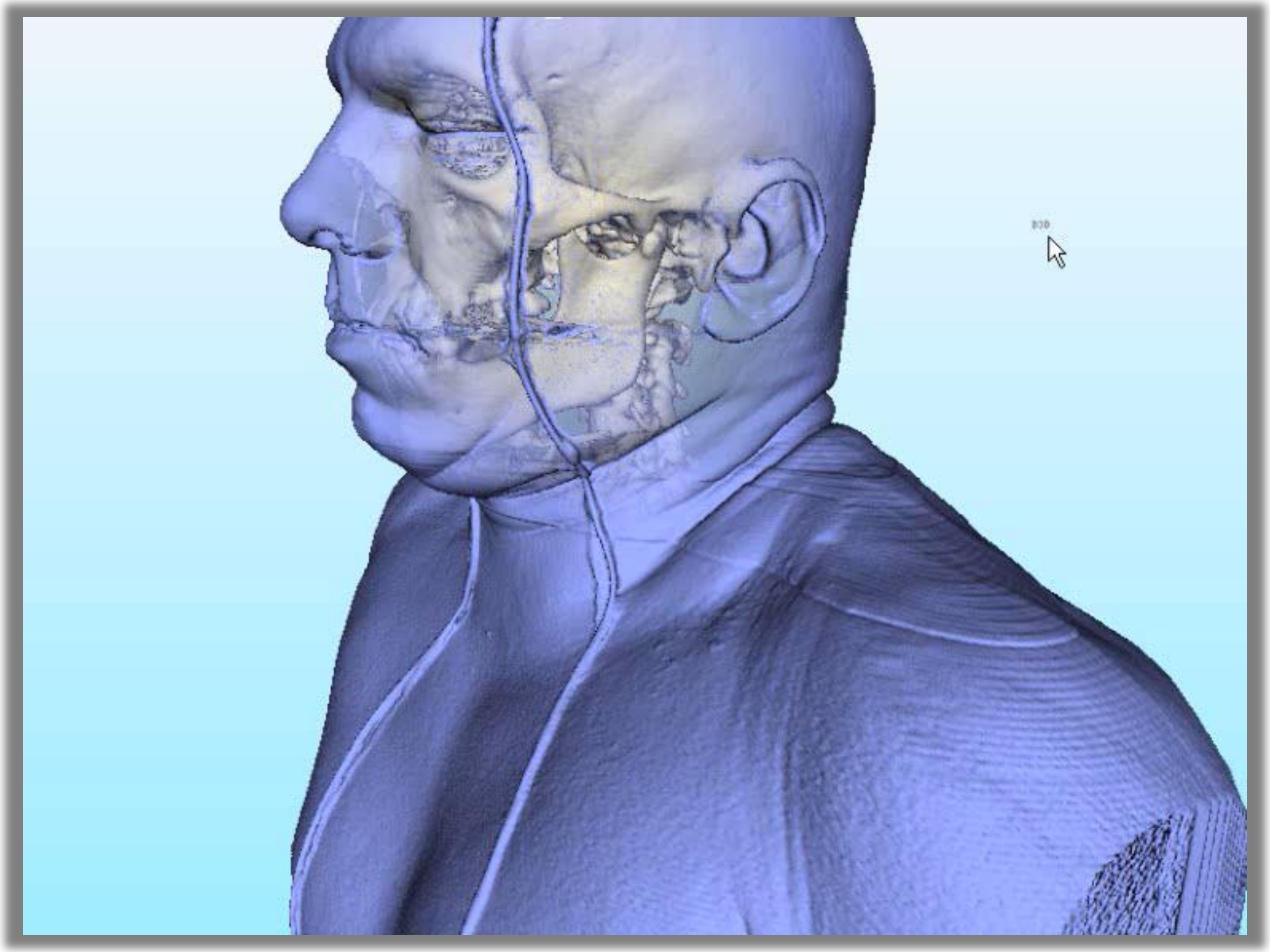


front-to-back strategy

Volume ray casting

- Volume ray casting on GPUs - takes advantage of
  - Hardware accelerated texture mapping
    - Either in CUDA or Graphics APIs
    - Including tri-linear texture interpolation and a fast texture cache
  - Framebuffer hardware for blending
    - Compositing of color/opacity in framebuffer
  - Many parallel compute units
    - Perform many numerical ray integrations in parallel

Volume ray casting



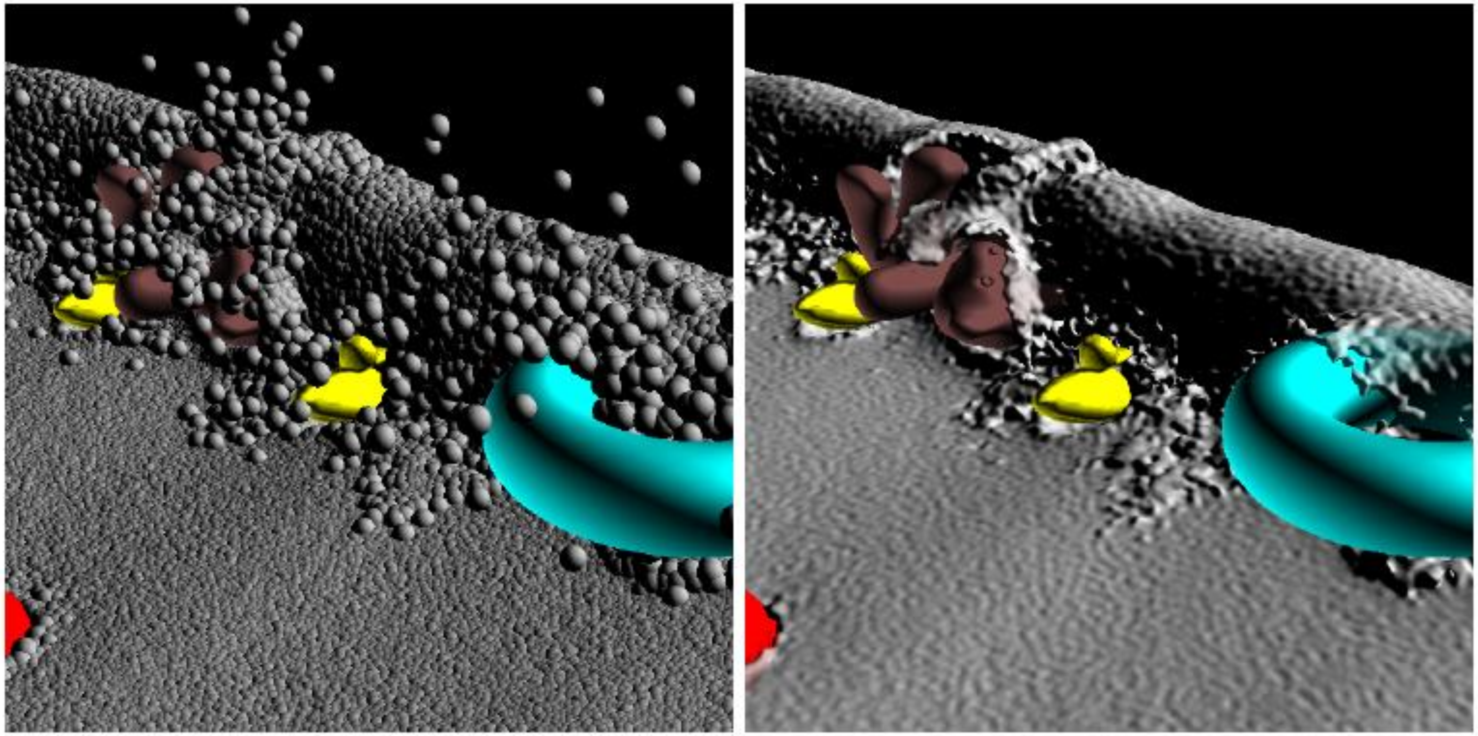
Data set:
512x512x512

Viewport:
1024x1024

Frame rate:
25 fps

GPU rendering of SPH simulations

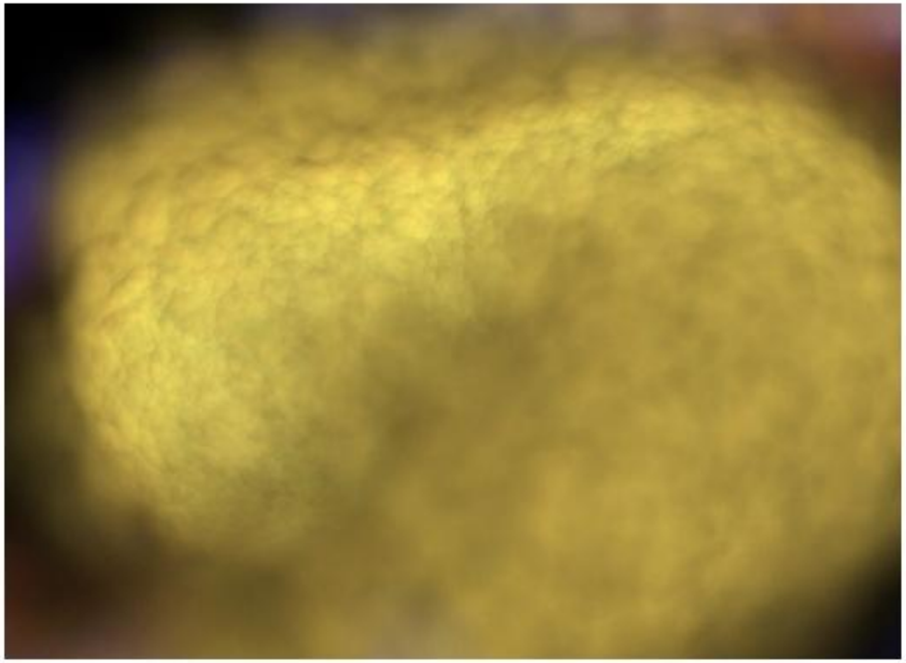
- Rendering of SPH particle simulations
  - Naive: render particles as opaque spheres



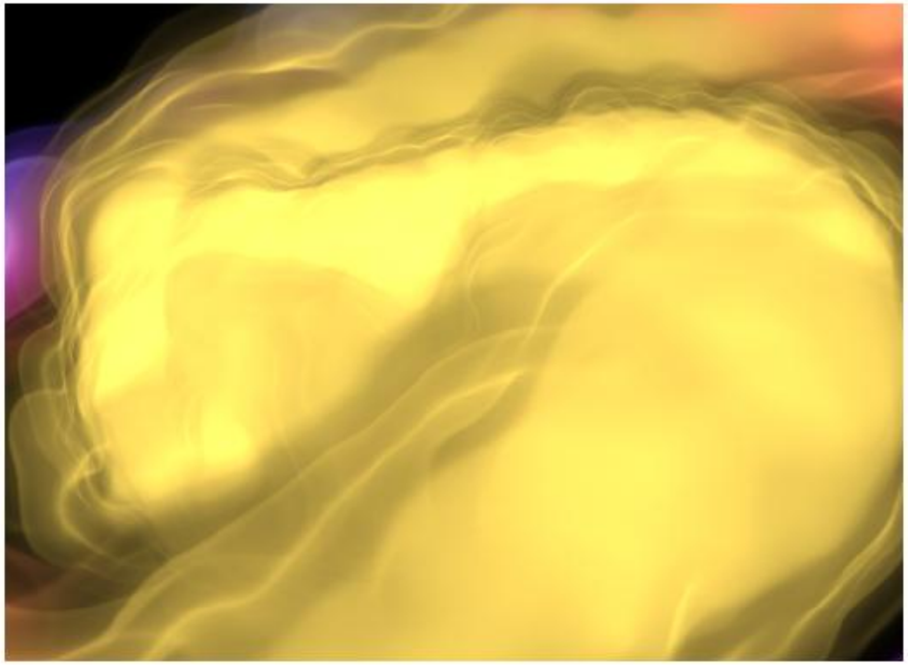
GPU rendering of SPH simulations

- Rendering of SPH particle simulations
  - Naive: sort particle wrt. viewer and splat as transparent circles

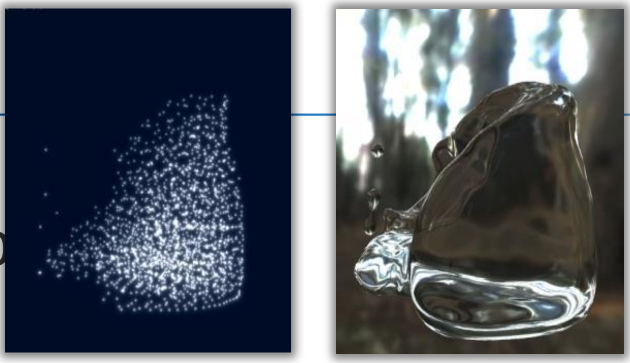
Splatting



This is what we might want to see

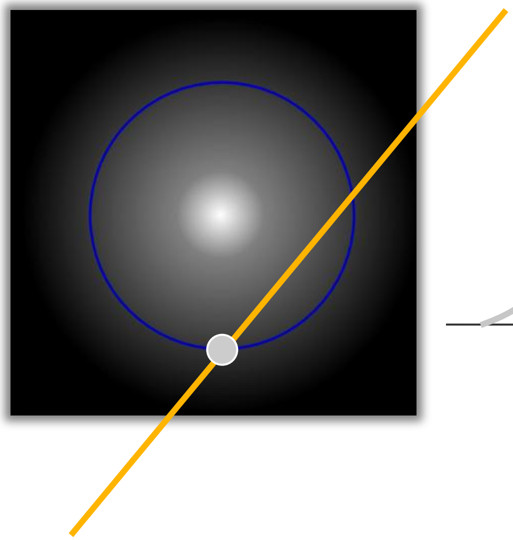


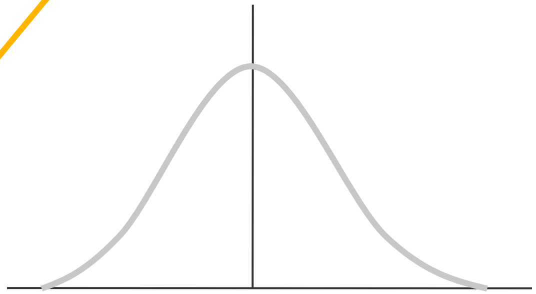
GPU rendering of SPH simulatio

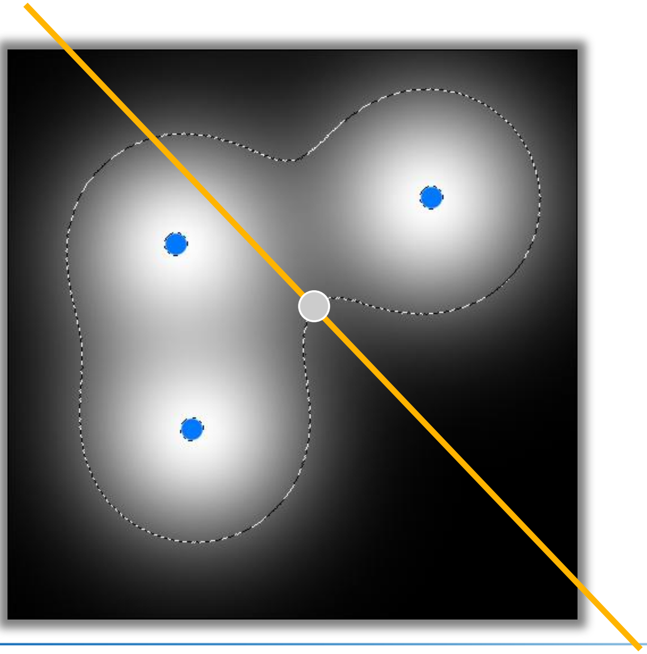


- High quality rendering of SPH fluid simulations
  - Traversal of view rays until intersection with fluid surface
  - Test for intersection:

    accumulated contributions of all particles > threshold

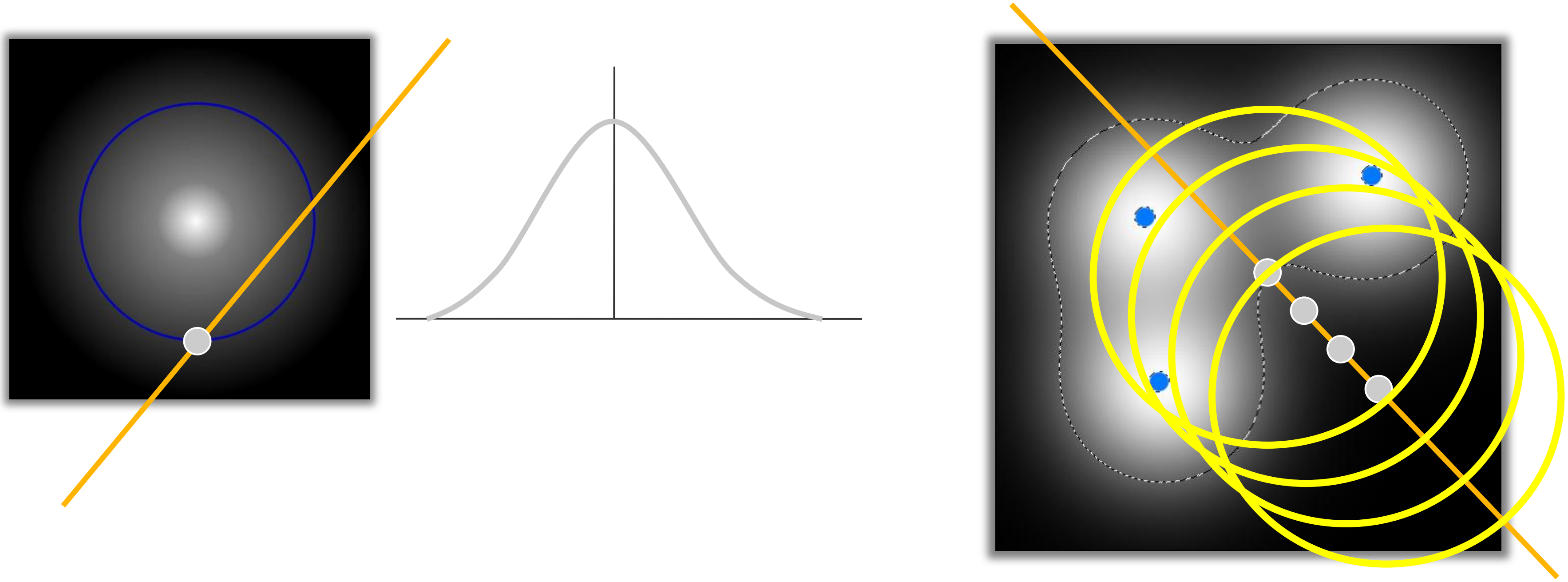






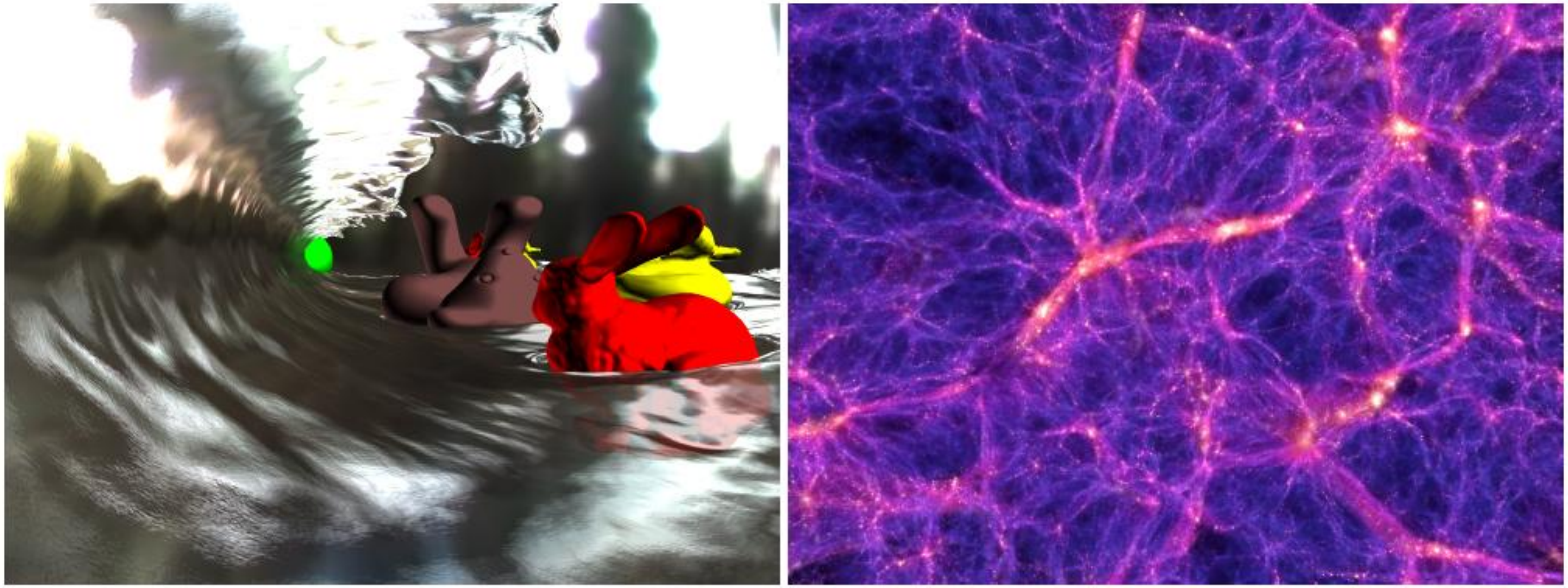
GPU rendering of SPH simulations

- High quality rendering of SPH fluid simulations
  - General approach: neighbor search at each sample point along the ray
  - Too costly!



GPU rendering of SPH simulations

- GPU friendly high quality rendering of SPH fluid simulations
  - Particle quantities are first resampled onto a 3D Cartesian grid
  - Use fast volume rendering on the GPU
    - Allows arbitrary rendering options

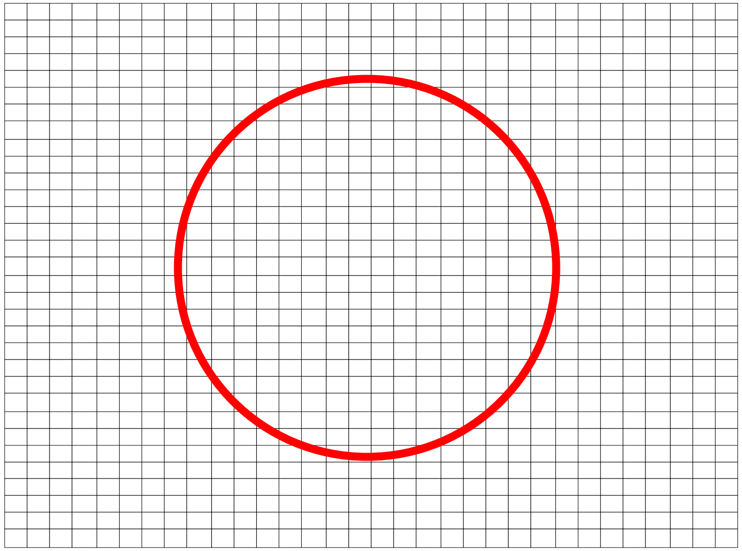


GPU rendering of SPH simulations

- GPU based particle resampling
  - Particle resampling into a 3D Cartesian grid, represented as a 3D texture map on the GPU
    - Exploits the GPU‘s capability to efficiently render into slices of a 3D texture
    - Particles overlapping a grid vertex are blended accumulatively

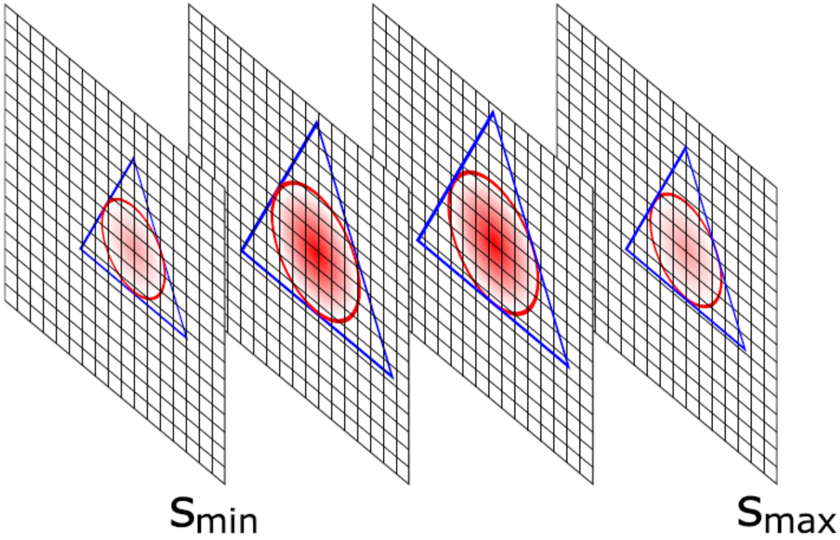
Foreach vertex covered by the particle:

- compute distance to particle center
- evaluate kernel function
- blend particle quantity weighted by kernel function into the texture



GPU rendering of SPH simulations

- GPU based particle resampling :
  - a) Determine slices S_{min} – S_{max} coverd by the particle
  - b) Determine bounding triangle of the particle-slice intersections
  - c) Render triangles with render target set to 3D texture slice



GPU rendering of SPH simulations

- Performance

Particles:
2.6M

Particle size:
r = 4 cells

Slices:
512 x512

Performance:
90ms

GPU rendering of SPH simulations

- Further examples

GPU rendering of SPH simulations

- Lessons learned:
    - GPU based volume rendering techniques for 3D scalar fields and scattered particle distributions
    - Exploits texture mapping and blending hardware for reconstruction and accumulation
    - Uses polygon throughput to resample particles into 2D texture slices
    - Enables high quality rendering including surface structures