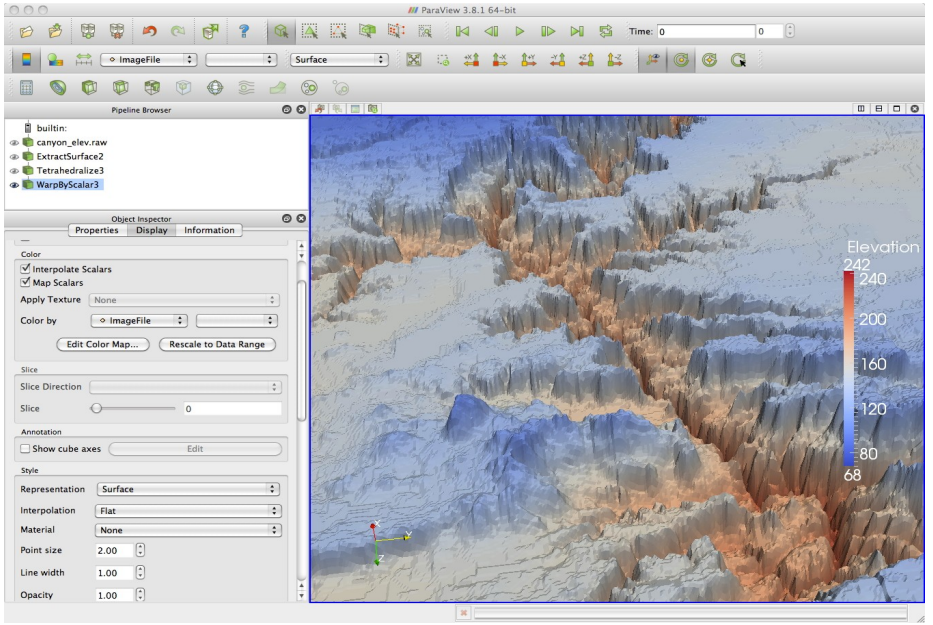
Introduction to scientific visualization with ParaView



Paul Melis
SURFsara Visualization group

paul.melis@surfsara.nl

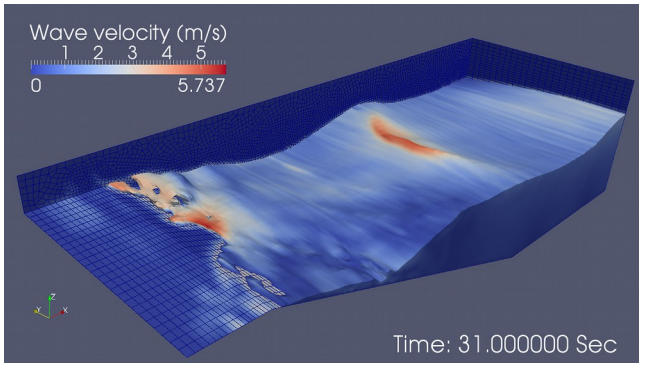
(some slides courtesy of Robert Belleman, UvA)

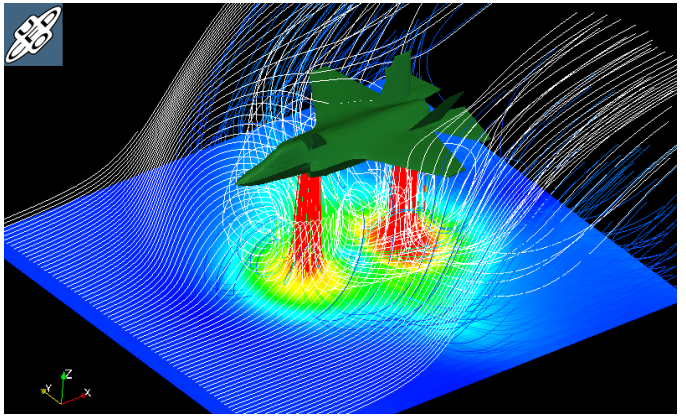
Outline

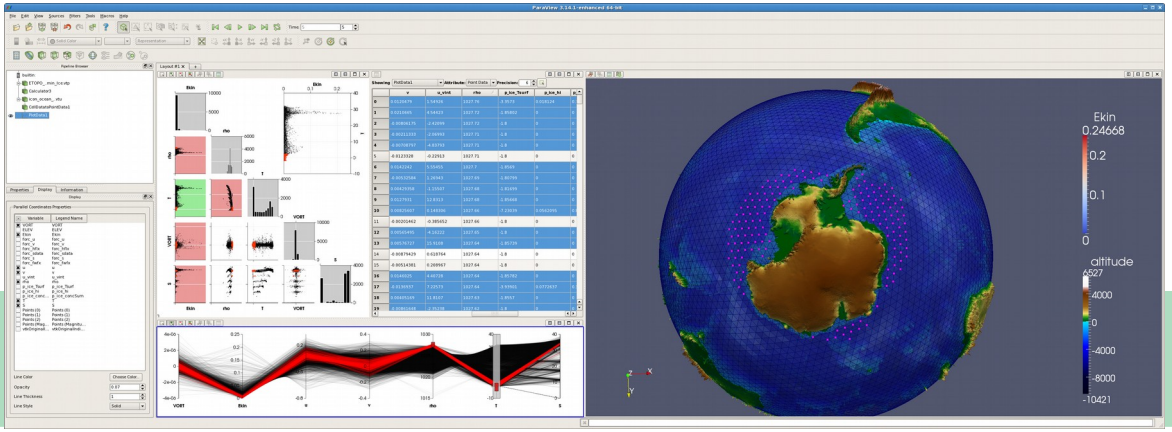
- Introduction, pipeline and data model (10 min)
- ParaView overview & walk-through (5-10 min)
- ParaView hands-on (60 min)
- ParaView wrap-up (5 min)

ParaView

- General scientific visualization package
  - Usable in many scientific fields
  - 2D/3D datasets
  - Data visualization & interactive exploration
  - Image/animation rendering
  - Not the best tool for:
    - information visualization, GIS, web-based visualizations
- Similar scientific visualization packages
  - VisIt, MayaVi, DeVIDE (TU Delft)
  - Knowledge about ParaView transfers mostly to other packages

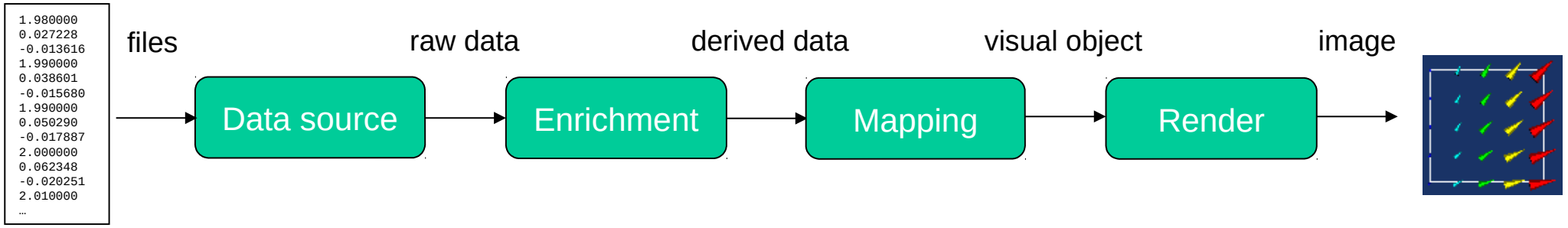






SURF SARA

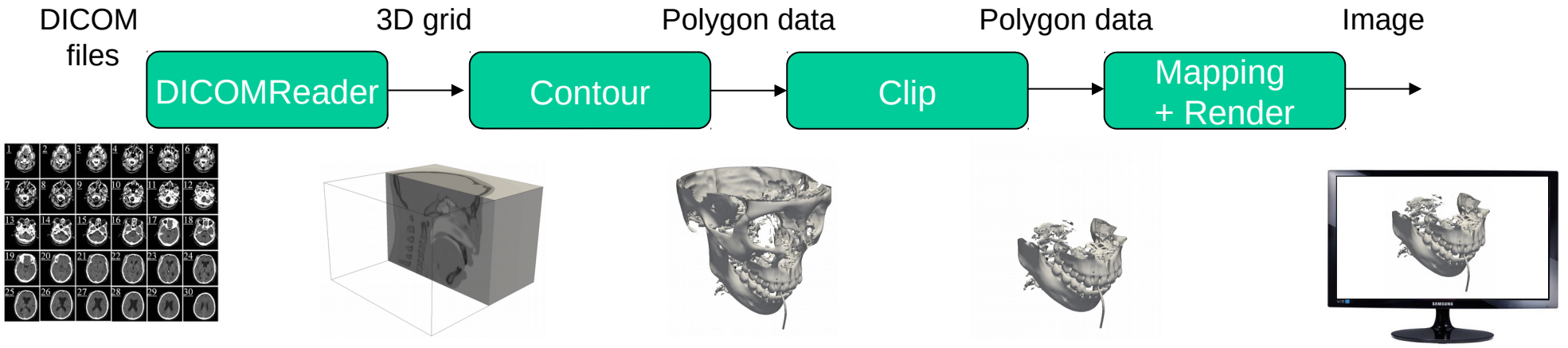
The scientific visualization pipeline



Haber and McNabb reference model

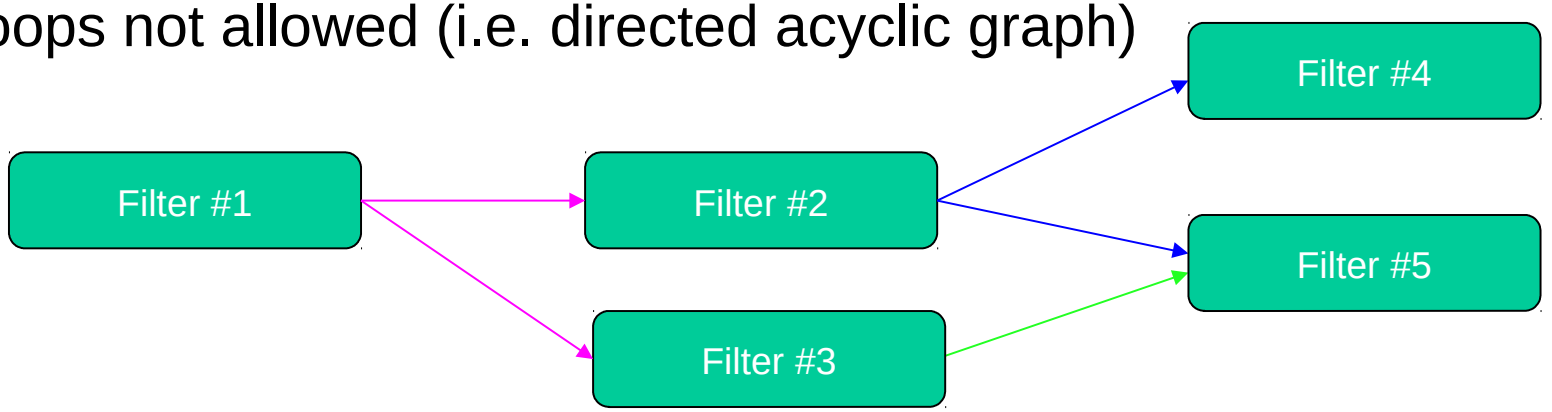
(after Haber, Robert B. & McNabb, David A., 1990, *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems*)

Example: extracting a contour from medical data



Pipeline creation

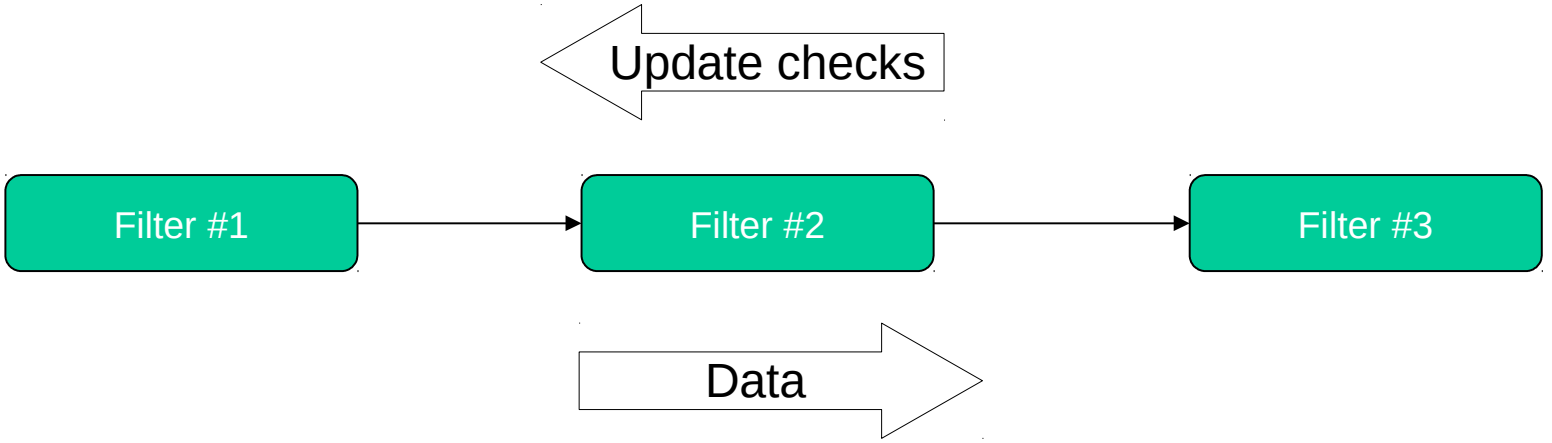
- Filters are connected together to form a "visualization pipeline" or "dataflow network"
  - Filters have inputs, outputs and parameters
- Restrictions:
  - Data types of connected input and output ports must match
  - Loops not allowed (i.e. directed acyclic graph)



(Arrow color indicates output data type)

Pipeline behaviour

- Filters in a pipeline only execute when necessary
  - When a filter's input data has changed
  - When a filter's parameter(s) have changed
- Data flows downstream, update checks flow upstream
- → On-demand local execution

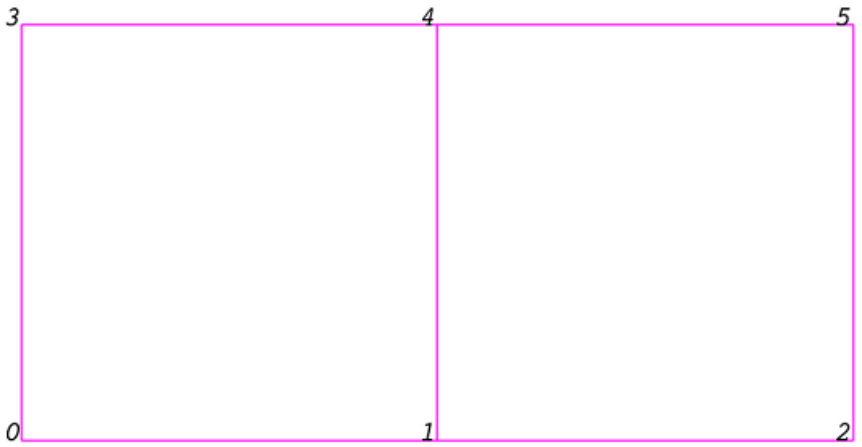


Data model

- Data sets are represented by a mesh and attributes

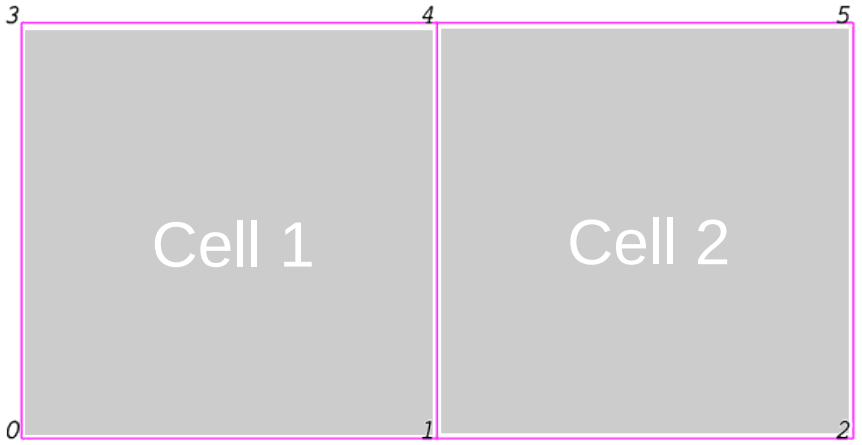
Data model

- Data sets are represented by a mesh and attributes
- A mesh consists of interconnected points in 2D/3D



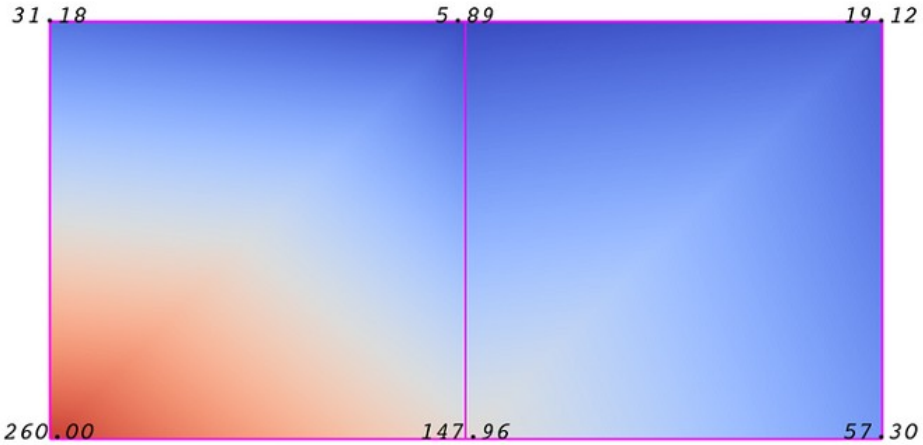
Data model

- Data sets are represented by a mesh and attributes
- A mesh consists of interconnected points in 2D/3D
- Collections of points form cells (regions, zones)



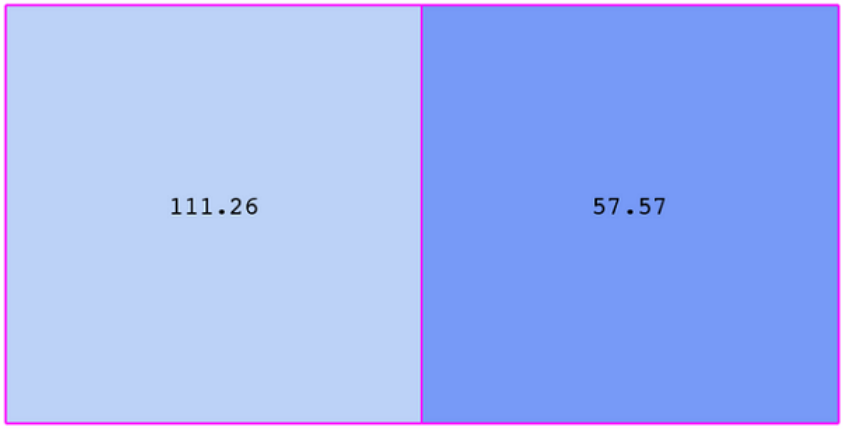
Data model

- Data sets are represented by a mesh and attributes
- A mesh consists of interconnected points in 2D/3D
- Collections of points form cells (regions, zones)
- Points can have attributes



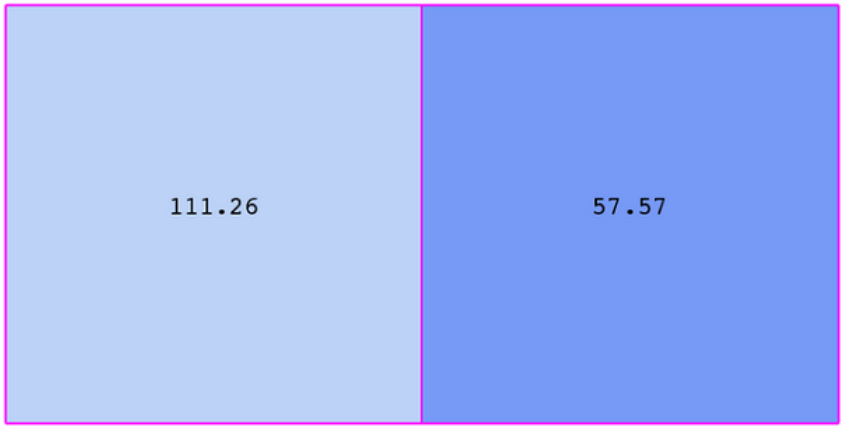
Data model

- Data sets are represented by a mesh and attributes
- A mesh consists of interconnected points in 2D/3D
- Collections of points form cells (regions, zones)
- Points can have attributes
- Cells can have attributes



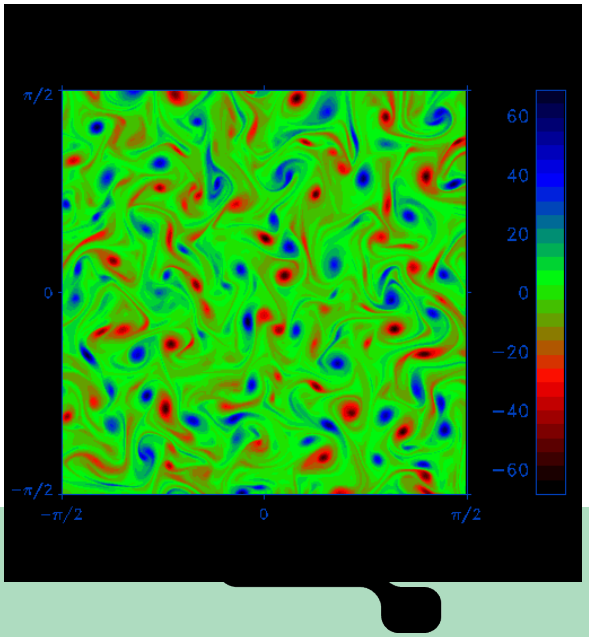
Data model

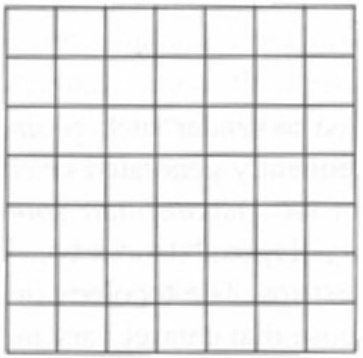
- Data sets are represented by a mesh and attributes
- A mesh consists of interconnected points in 2D/3D
- Collections of points form cells (regions, zones)
- Points can have attributes
- Cells can have attributes
- Points define geometry, cells define topology



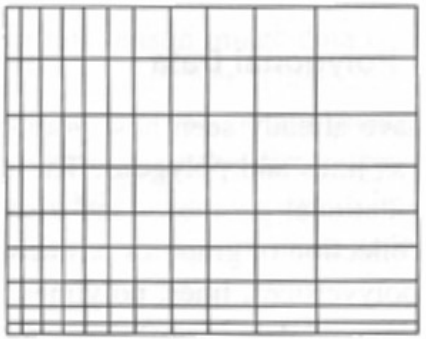
Rectilinear grid (a.k.a. “image data”)

- Regular 2D/3D grid
- Defined by
  - Origin
  - Spacing in X, Y and Z
  - Dimensions in X, Y and Z
- *Cells are always rectangular*
- Examples:
  - Images (2D)
  - Medical scans (3D)
  - Atmospheric/fluid simulations (2D/3D)



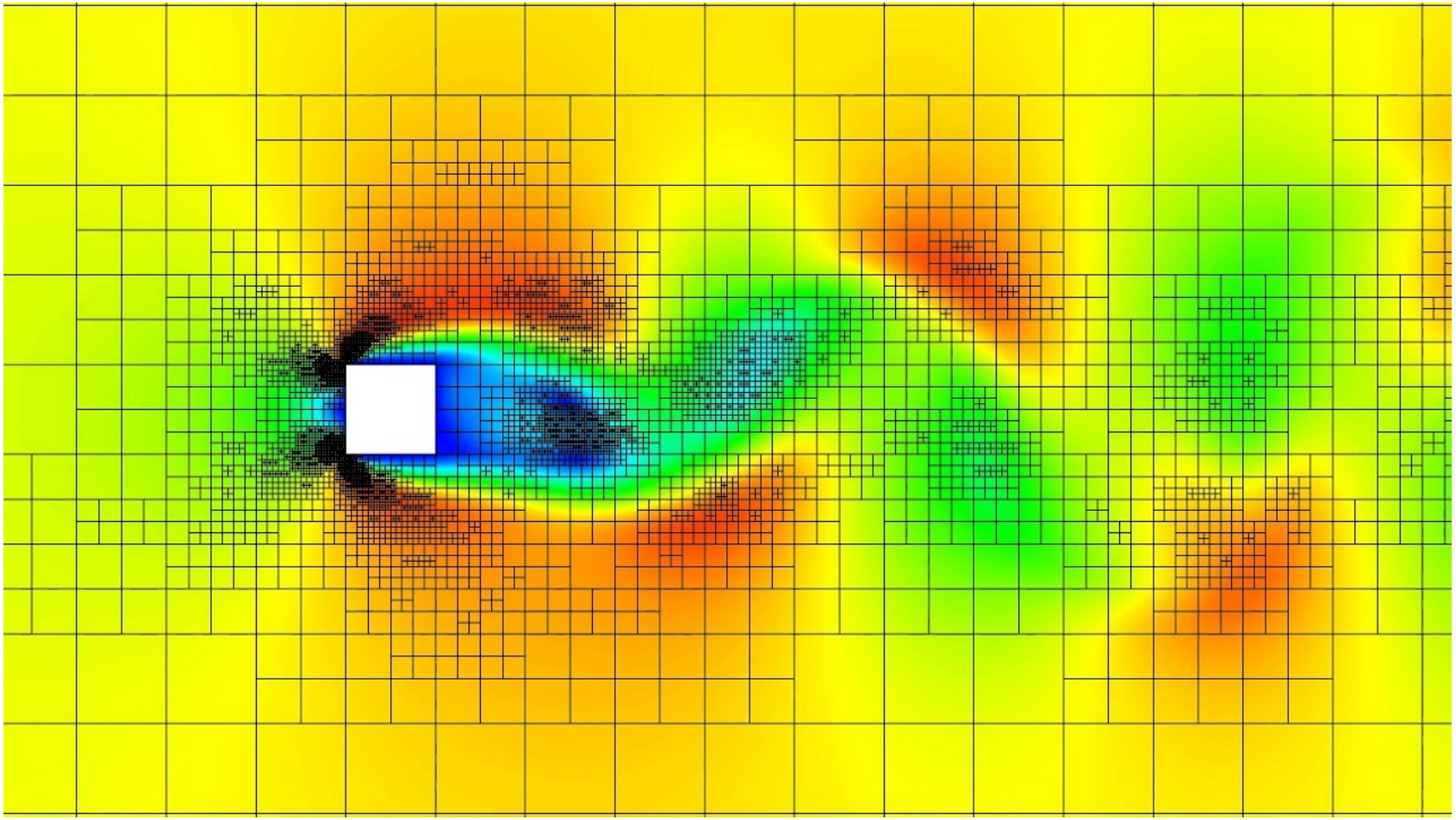


(a) Image Data



(b) Rectilinear Grid

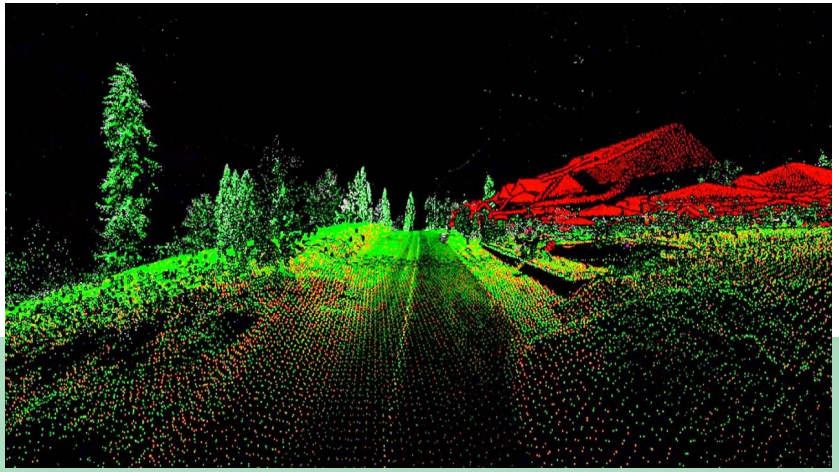
Adaptive mesh refinement



SURF SARA

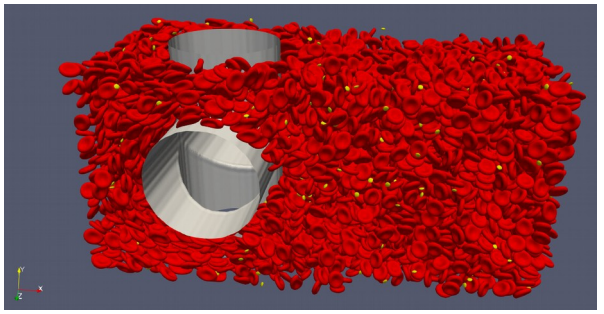
Point-based datasets

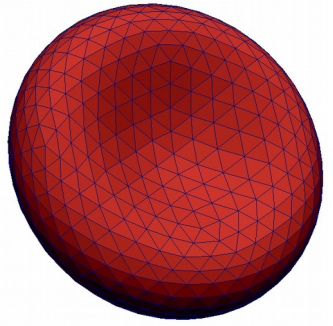
- No connectivity, only positions
- Per-point data
  - Velocity
  - Mass
  - Etc.
- *Cells are points*
- Examples:
  - N-body simulations
  - LiDAR data
  - Agent-based simulations

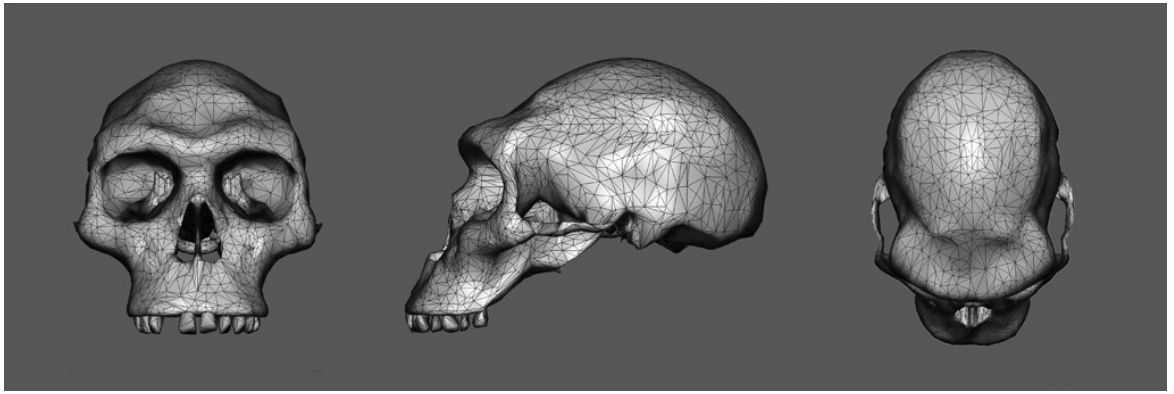


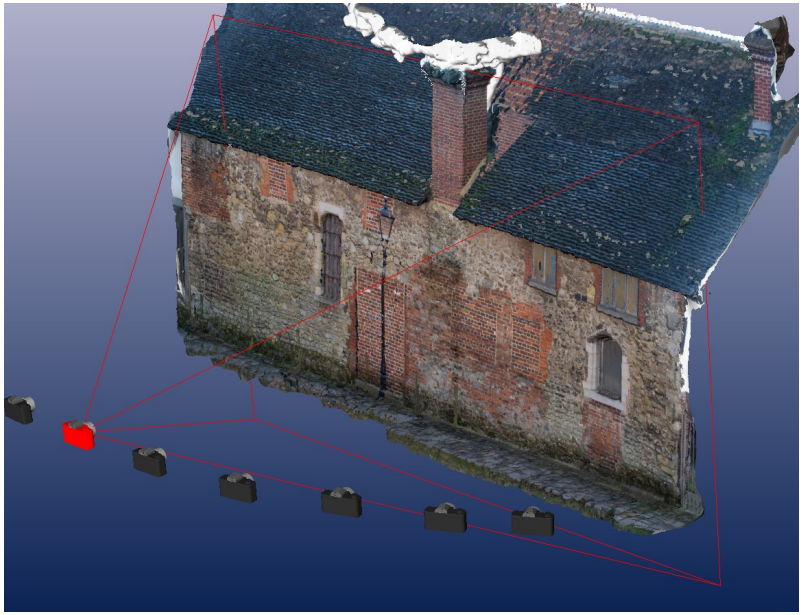
Polygonal data (surfaces)

- Thin surfaces
  - Possibly closed
  - Per-point and per-cell values
  - *Cells are triangles, quads, lines, points, …*
- Examples:
  - Cell-based biological simulations
  - Isosurfaces from medical scans
  - Photogrammetry output



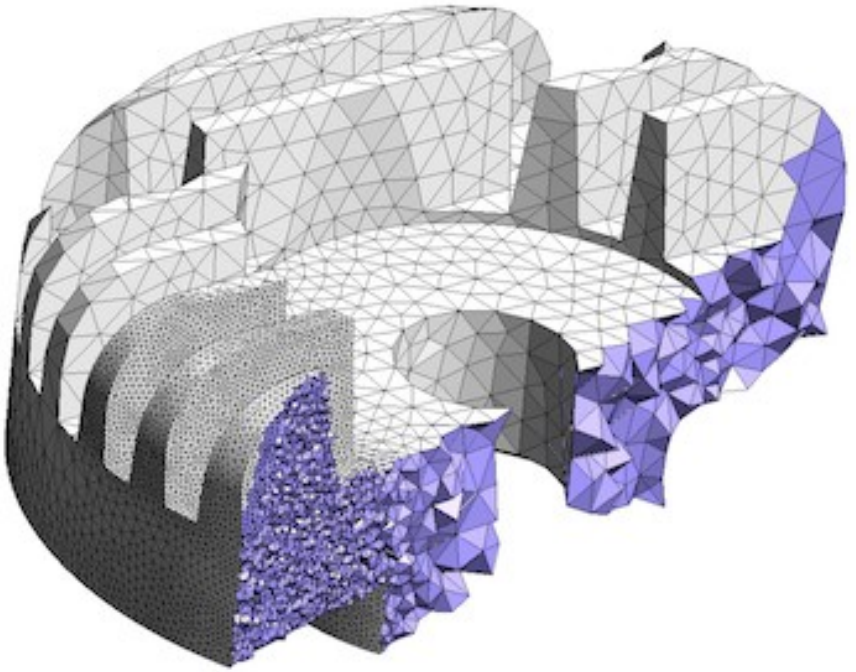






Unstructured grid

- Collection of different cells types
  - “Bag of cells”
- No regular structure
- Last resort when previous dataset types are not applicable
- Examples
  - CFD meshes (tetrahedron cells)
  - Polydata (previous slide) is a form of unstructured grid
- *Some ParaView filters produce unstructured grids*



ParaView GUI
and
basic functionality
(demo)

Hands-on preparation

- Step 1: Install ParaView 5.4 on your laptop
  - We have a USB stick with Paraview + data files
  - OR download from http://bit.ly/2tn868Q (will be faster than downloading from www.paraview.org)
- Step 2: Download the data files
  - Download .zip file from http://bit.ly/2rvX5kj (only 8MB)
  - Unpack somewhere on your laptop
- (Optional) Step 3: Troubleshoot GPU rendering
- Or download through URLs listed in hand-out
- Ask us for help if needed

Hands-on!

- Until 11:30
- ParaView exercises
  - Exercise 1: CT-scan of a head
    - Data inspection, slices, volume rendering, contouring
  - Exercise 2: Tornado simulation
    - CSV file reading, streamlines, glyphs, coloring
  - Exercise 3: Coral growth
    - Time-varying datasets, camera orbiting
  - Bonus exercise: A stationary fluid mixer
- Ask us for help if needed!

Topics not covered

- Making movies (animation rendering)
- Python scripting
  - Save/restore sessions
  - Write your own filter
  - Integration with NumPy and matplotlib
- And much more...

- User's Guide is freely available as PDF (239 pages)!
  - It is included when downloading binaries
  - Or see Kitware blog: http://www.kitware.com/blog
- See Kitware ParaView Tutorial
  http://www.paraview.org/Wiki/The_ParaView_Tutorial

Getting data into ParaView?

- Lots of formats already supported by ParaView
  - NetCDF, OpenFOAM, PLY, HDF5, ExodusII, ...
    - See http://www.paraview.org/Wiki/ParaView/Users_Guide/List_of_readers
  - .csv, .txt (Delimited text) – Loads as a table, need to do extra manual steps
  - Binary data without header, select “Raw (binary) file” type. Very limited
- When you're going to write the data yourself
  - ParaView/VTK native formats
    - Legacy” VTK file format or VTK/ParaView XML file format
    - See http://www.vtk.org/VTK/img/file-formats.pdf
    - Writing Legacy and XML files possible using VTK library instead of doing it “by hand”
  - HDF5 + XDMF
    - See http://www.xdmf.org

The end...

Legacy VTK versus XML-based

```
# vtk DataFile Version 2.0
Volume example
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 3 4 6
ASPECT_RATIO 1 1 1
ORIGIN 0 0 0
POINT_DATA 72
SCALARS volume_scalars char 1
LOOKUP_TABLE default
0 0 0 0 0 0 0 0 0 0 0 0
0 5 10 15 20 25 25 20 15 10 5 0
0 10 20 30 40 50 50 40 30 20 10 0
0 10 20 30 40 50 50 40 30 20 10 0
0 5 10 15 20 25 25 20 15 10 5 0
0 0 0 0 0 0 0 0 0 0 0 0
....
```

```
<VTKFile type=”StructuredGrid” ...>
  <StructuredGrid WholeExtent=”x1 x2 y1 y2 z1 z2”>
    <Piece Extent=”x1 x2 y1 y2 z1 z2”>
      <PointData>...</PointData>
      <CellData>...</CellData>
      <Points>...</Points>
    </Piece>
  </StructuredGrid>
</VTKFile>
```

SURF SARA

Comparison

Legacy

- Pros
  - Easy to write from your own software
  - ASCII format is very easy to write
  - Binary format is space-efficient
- Cons
  - Not developed anymore
  - ASCII format is space-inefficient
  - Binary format slightly harder to write
  - No support for parallel reading/rendering

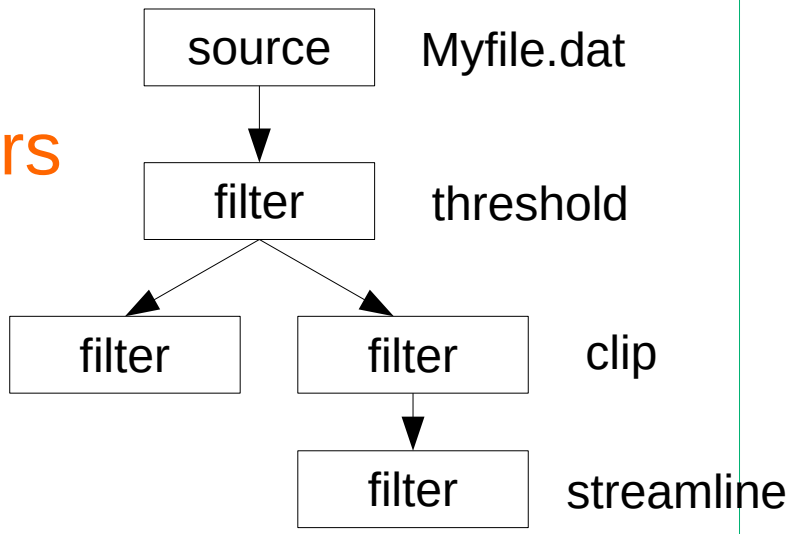
XML

- Pros
  - Probably more future-proof than legacy format
  - Optional data compression
  - Possibility of parallel reading/rendering for large datasets
- Cons
  - Need to write XML-compliant files and understand more complex file structure
  - Not as compact as pure binary, due base64 encoding of data

http://www.vtk.org/VTK/img/file-formats.pdf

ParaView pipeline

- Filter
  - Operates on (one or more) input datasets
  - Produces an output dataset
  - Usually has a set of parameters
  - Example operations:
    - Clip, threshold, streamline
- Source
  - Conceptually a filter with no inputs
  - A loaded file (or set of files) becomes a source



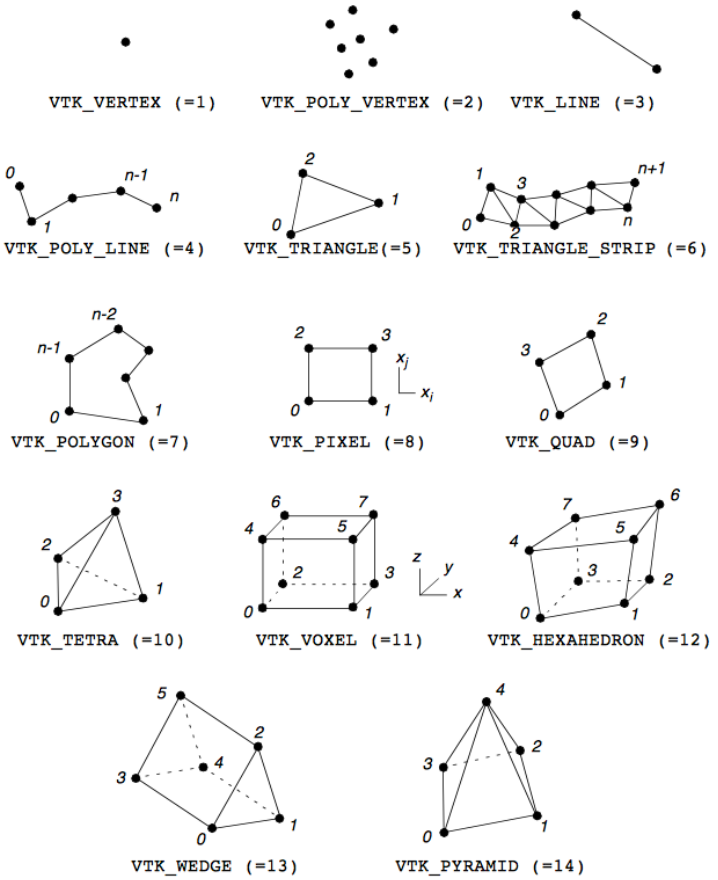
ParaView/VTK data model

- Points
  - 1D/2D/3D coordinates
  - Describe the geometry of the data (i.e. spatial locations)
- Cells
  - Cells refer to points
  - Describe the topology of the data (i.e. connectedness)
  - Cells are the things that ParaView visualizes
- Data can be associated with both points and cells
  - E.g. temperature, pressure, flow direction and/or velocity
  - Types of values:
    - Scalar values (integer, floating-point)
    - Vectors
    - Strings
    - (3x3 sym. tensors, 3D normals, texture coordinates, field data)
  -

Why ParaView (and why for this course)?

- Advantages
  - Free & open-source
  - Actively developed and supported by Kitware (a US company)
  - Lots file formats and data operations (filters) supported
  - Allows parallel visualization of large datasets
  - For this course?
    - GUI and workflow is pretty good, especially for *interactively* building a visualization pipeline
    - Concepts and operations in ParaView transfer easily to other packages like VisIt or MayaVi, so provides good introduction to scivis methods
- Caveats
  - It does have bugs in some areas, so...
    - It might crash unexpectedly
    - Annoying warning messages sometimes pop up
  - Does not handle out-of-memory situations very well

Cell types



(Linear cell types)

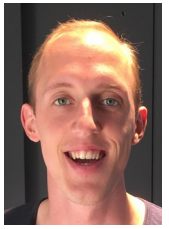
Cells are the things that ParaView visualizes!

SURFsara visualization group

- Hands on time!
  - Time to start working on the exercises
  - If you have questions, let us know
  - Data:
  - Software:

Paul Melis

Tijs de Kler



Casper
van Leeuwen

SURF SARA