Scientific Visualisation

Dr. John Redford

Eurobios SCB, Cachan, France

Jan/Feb 2023

Content

Objectives

Master the principle techniques of representation, modelling, and visualisation of large groups of 3D data (structured and unstructured)

- Data : examples, meshing
- Surface data visualisation
- Volume data visualisation
- Visualisation of fluid mechanics data

Material

Installation on ParaView from **here**

Exercises from the **ParaView tutorial guide 5.11**

Download all ParaView tutorial data `ParaViewData-v5.11.0.zip` from **here** or the file for the first day's exercise **here**

Planning

Course dates/content:

1. Day 1 (CM/TD)
   - Fundamentals
   - Data filtering
2. Day 2 (CM/TD)
   - Time dependent data
   - File formats
3. Day 3 (TD/TD)
   - Exercises
   - Fluid mechanics (OpenFOAM) visualisation
4. Day 4 (CM/TD)
   - Python scripting
5. Day 5 (CM/TD)
   - Visualisation of large models
   - Evaluation exercise

CM (Cours Magistral/Lecture) – TD (Travaux Dirigés/Tutorial)

Introduction : Scientific Visualisation

Wiki page 'The purpose of scientific visualization is to graphically illustrate scientific data to enable scientists to understand, illustrate, and glean insight from their data.'

Topics of general interest

- Human Perception for Information Display
- Basic vision and displays
- Elementary interactive graphics programming
- Colour Basics Colour For Information Display
- Texture, pattern and motion for information display
- The visual thinking process when using visualizations as cognitive tools
- Interactive methods and cognitive efficiency

Rendering, Rasterization
Use of colour : **sciviscolor**, **K Moreland** anti-rainbow colourmap.

What is Scientific Visualisation?

- Transformation of data or information into pictures (visual outputs)
  - Does not necessarily imply the use of computers
  - Classical visualisation used hand-drawn figures and illustrations (2D means for visualisation)
- Modern visualisation is primarily 3D (digital images for 3D visualisation)
- In both cases, the ultimate goal is to understand important insights for the data through visual means
- Do not care how we visualise the picture – what picture we get is most important
- Technical means used to arrive at visual outputs are mainly dependent on computer graphics techniques

Why is visualisation useful and important?

Which is more useful or accessible: left or right, or both?

About 71% of the Earth's surface is water-covered, and the oceans hold about 96.5% of all Earth's water.

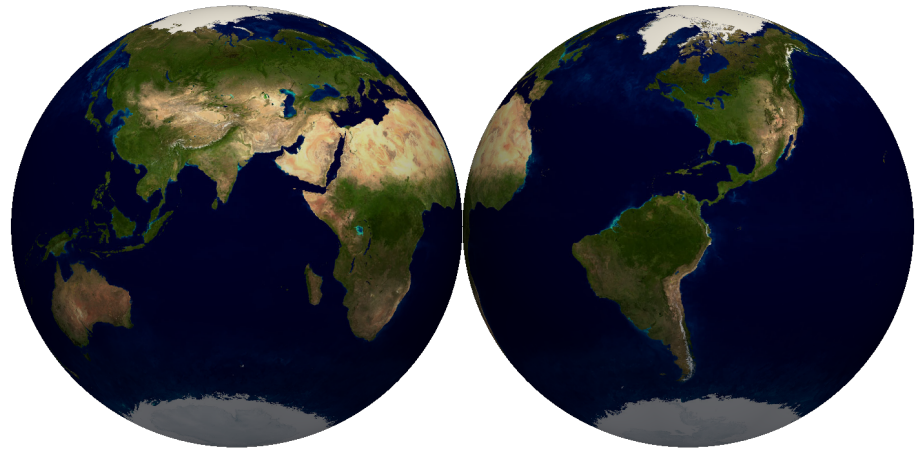


Figure 1: Earth

Visualisation terminology

- Different sub-fields of visualisation
- Scientific visualisation
  - discipline of computer science
  - visualisation of scientific and engineering data-sets
- Scientific visualisation touches on a number of areas:
  - data representations
  - data processing algorithms
  - visual representations
  - user interfaces

Visualisation Terminology

- *Data visualisation* – includes data from other sources, such as financial, marketing, business
- Sometimes involves statistical analysis and other analysis techniques not employed in scientific visualisation
- Can you think of an example of financial information we might want to visualise?
- So we might say that scientific visualisation is a type or subset of data visualisation

Figure 2: Cours du CAC 40 en points

Motivation

- Make sense of huge data-sets
  - Computational fluid dynamics produces very large data-sets
- Uncover insights hidden in the data
- Extract important features and meaningful knowledge of the data to assist in the decision-making process

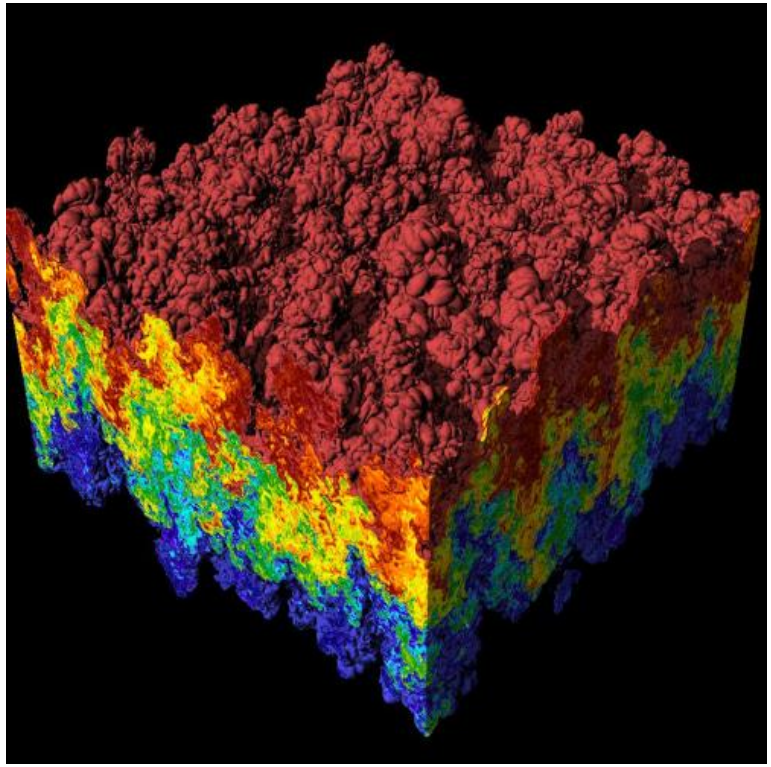


Figure 3: Rayleigh-Taylor instability simulation with up to 3072^3 cells

2D example: Flow map

- Show the movement of objects from one location to another
- Demonstrates route, size of army and return temperature

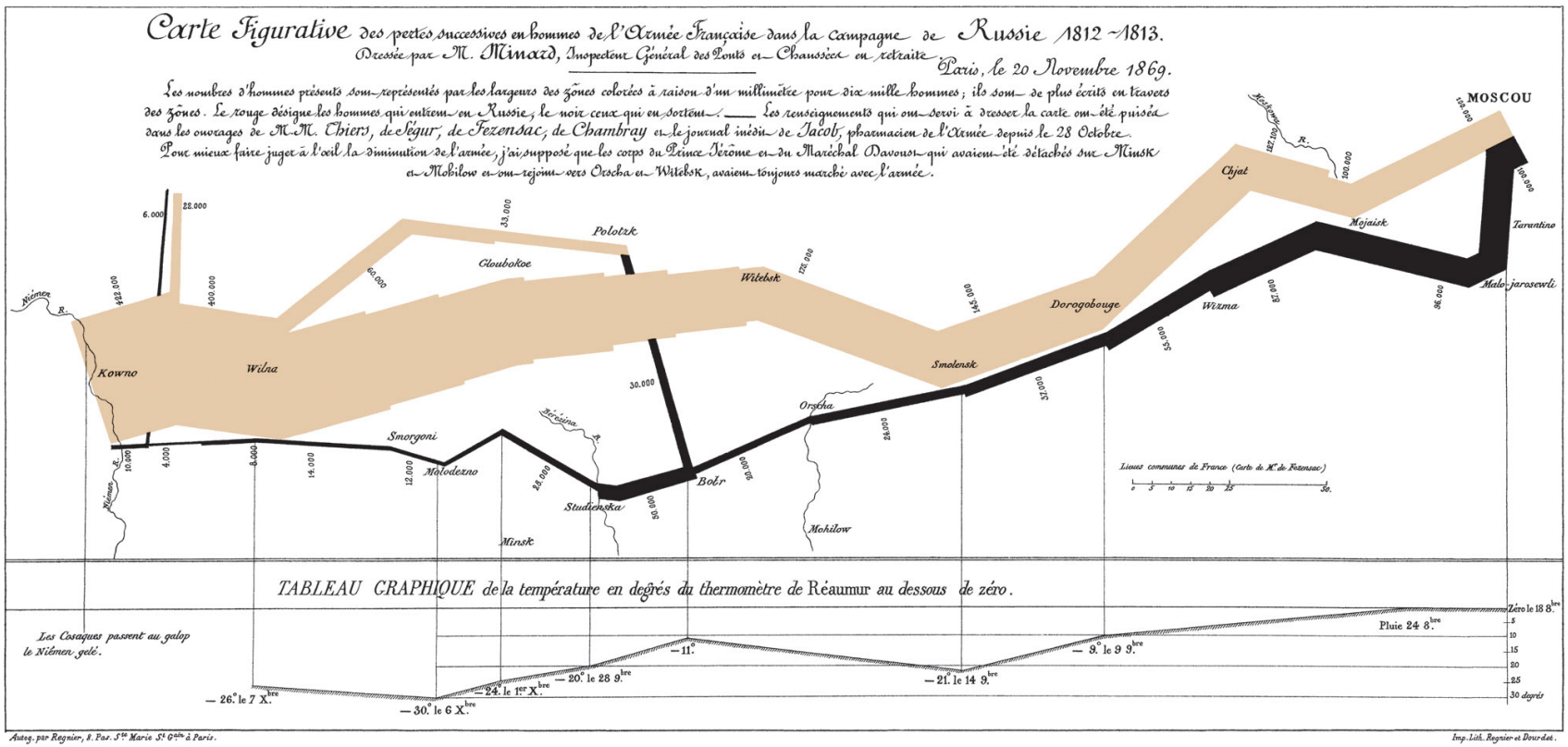


Figure 4: Charles Minard's flow map of Napoleon's March.

2D example: Dot style

Visualise disease (cholera) outbreak and find causes

- Cases clustered around pump in Broad (now Broadwick) street
- Water for the pump was polluted by sewage from nearby cesspit

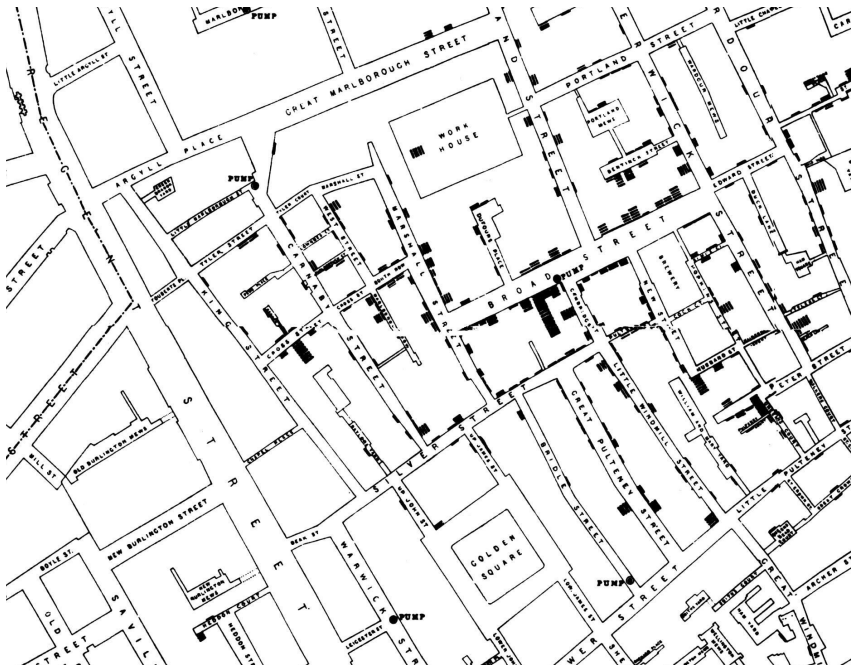


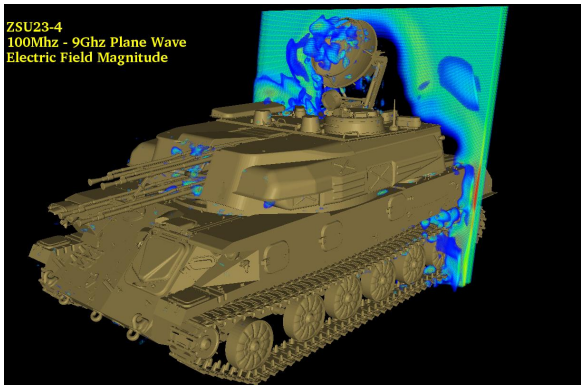
Figure 5: John Snow's Cholera map in dot style, 1854.

Introduction : ParaView

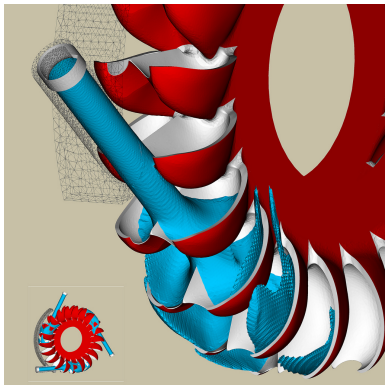
- Open-source, scalable, multi-platform visualisation application
- Data parallelism on shared-memory or distributed-memory multi-computers and clusters
- An open, flexible, and intuitive user interface
- An extensible, modular architecture based on open standards
- Large user community, both public and private sector
- A flexible BSD 3 Clause license

Downloaded roughly 100,000 times a year Won many awards

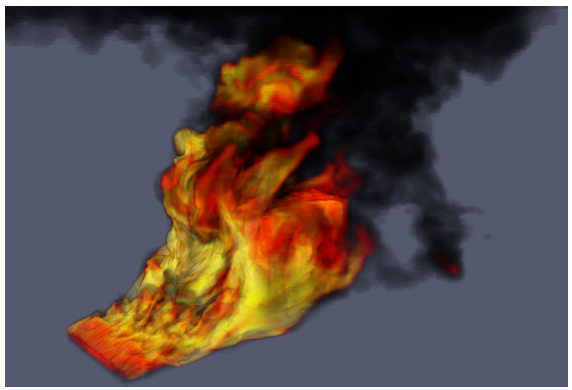
Introduction : ParaView examples



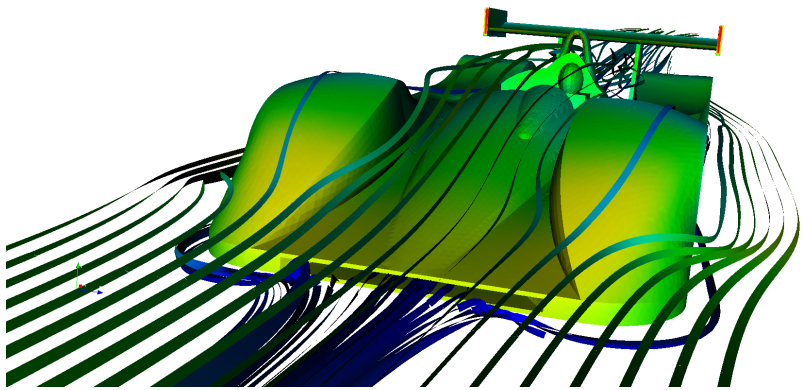
ZSU23-4 Russian Anti-Aircraft vehicle being hit by a planar wave. Image courtesy of Jerry Clarke, US Army Research Laboratory.



Simulation of a Pelton turbine. Image courtesy of the Swiss National Supercomputing Centre

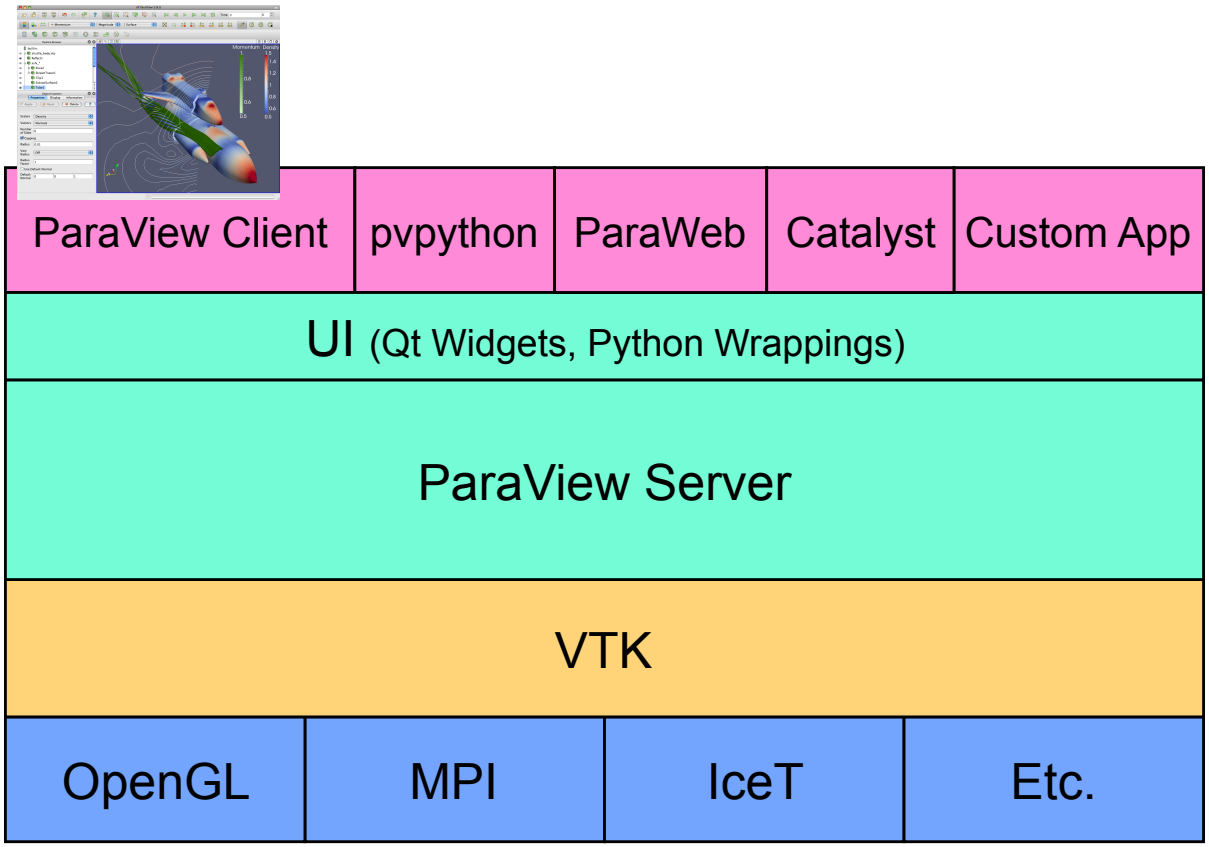


A loosely coupled SIERRA-Fuego-Syrinx-Calore simulation with 10 million unstructured hexahedra cells of objects-in-crosswind fire.



Airflow around a Le Mans Race car. Image courtesy of Renato N. Elias, NACAD/COPPE/UFRJ, Rio de Janerio, Brazil

Introduction : ParaView application Architecture



- ParaView is just a small client application built on top of a tall stack of libraries
- pvpython allows automation of post-processing

Introduction : VTK library

Visualization Toolkit (VTK) is an open-source, freely available software system for :

- 3D computer graphics, modelling, image processing, volume rendering, **scientific visualisation**, 2D plotting

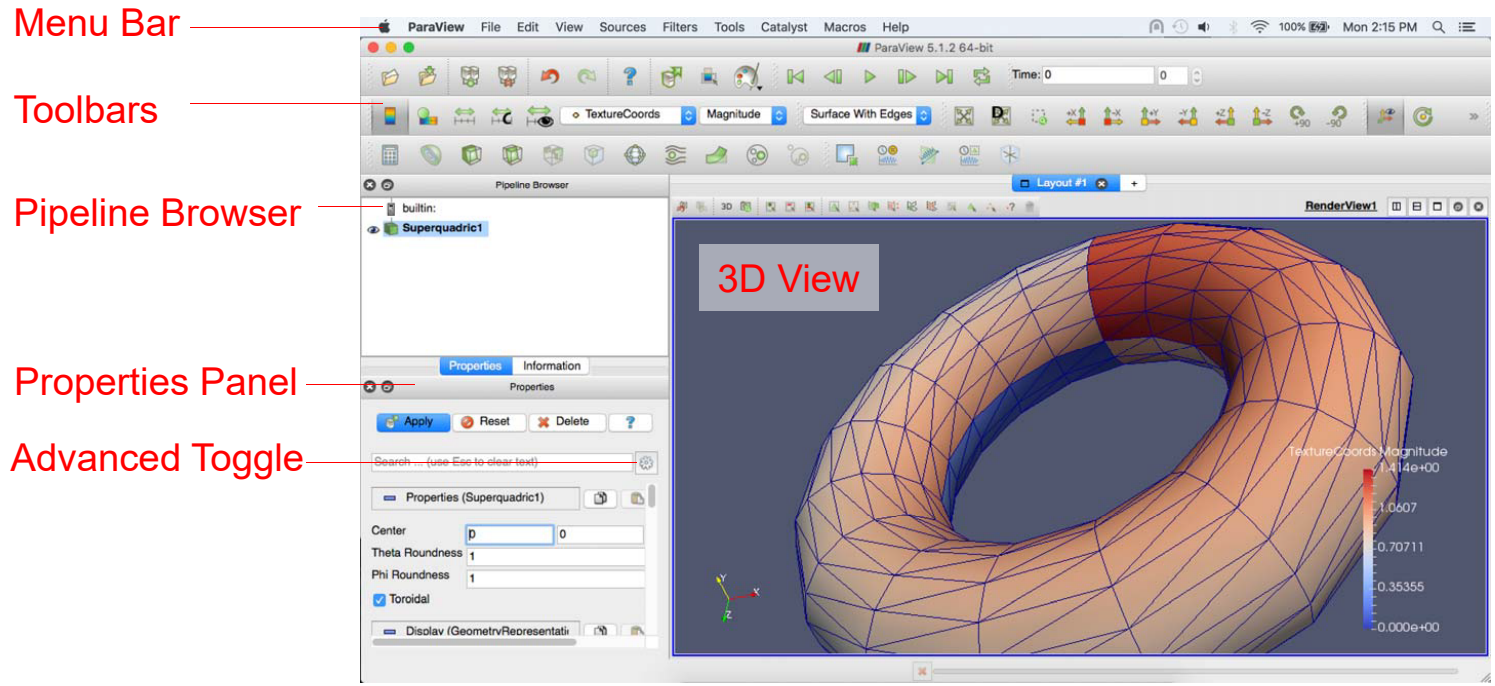
Filters :

- VTK applications manipulate data with filters
- inspect received data and produces derived data

Multiple applications, e.g. open source platforms include :

- The **Insight Segmentation and Registration Toolkit (ITK)**; algorithms for medical research
- **ParaView** works with data on the desktop, on the web, on supercomputers, in immersive environments and more
- **PyVista** 3D plotting and mesh analysis
- **3D Slicer** is an extensible platform for visualisation and medical image analysis

ParaView Interface



Menu Bar Allows access to majority of features

Toolbars Provides quick access to most commonly used features

Pipeline Browser ParaView manages the reading and filtering of data with a pipeline. Browser allows the pipeline structure and select pipeline objects to be viewed

Properties Panel Allows viewing and changing of parameters for current pipeline object

3D View Used to present data so that you may view, interact with, and explore data

Exercise 1: ParaView line plot

- Create `.csv` file (right)
- `File` → `Open`, `OK` & `Apply`
- `[ctrl] + [space]`, and search for `Plot Data`, `Apply`
- Change x-axis parameter

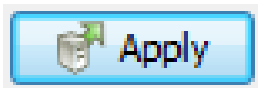
```
"x", "y"
0 , 0
1 , 1
2 , 4
3 , 9
4 , 16
5 , 25
6 , 36
7 , 49
```

Sources

Two ways to get data into ParaView:

- Read data from a file
- Generate data with a **source** object

Exercise; Open ParaView, then `Sources` $\rightarrow$ `Cylinder`, `Apply`



After, increase the cylinder resolution. Manipulate the cylinder in the viewing window. Continue by demonstrating basic features of ParaView interface (colours, axes, opacity, auto apply, colour palette).

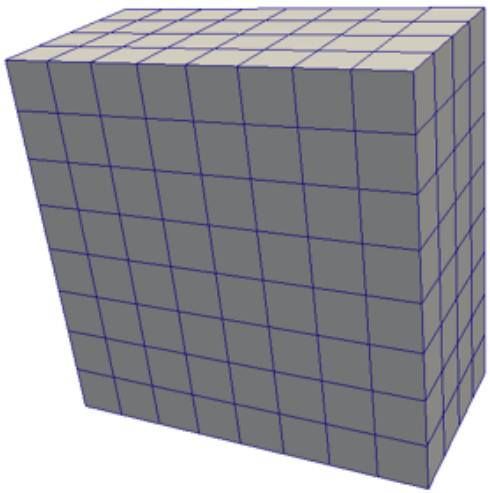
Earth visualisation

Familiarisation exercise using the earth example **here**. Prenons le **image**, puis :

1. Ouvrir ParaView
2. Cliquer sur le bouton `Open`, et ouvrir le fichier `Land_ocean_ice_2048.jpg`; `Apply`.
3. Faire bouger l'image avec la souris
4. Cliquer sur le bouton `Split vertical` (en haut à droite de la fenêtre de visualisation). Dans la fenêtre `Create View` qui apparaît, cliquer sur le bouton `Render View`
5. Sélectionner le menu `Sources` → `Sphere`. Cliquer sur Apply. Changer les valeurs Theta resollution et Phi resolution sur 16 et `Apply`
6. Sélectionner le menu `Filters` → `Alphabetical` → `Texture Map to Sphere`. `Apply`
7. Dans la boîte `Properties`, cliquer sur `Toggle advanced properties`  . Dans la zone Miscelaneous (tout en bas), cliquer sur le menu `Texture` et choisir l'option Load...
8. Dans la fenêtre `Open texture` qui apparaît, aller chercher le fichier `Land_ocean_ice_2048.jpg` et cliquer sur OK
9. Dans la boîte `Properties`, décocher l'option `Prevent` seam, `Apply`

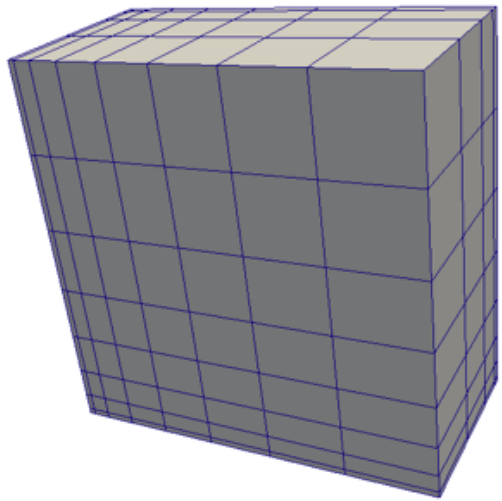
But, this example is not quite perfect...

Regular mesh types



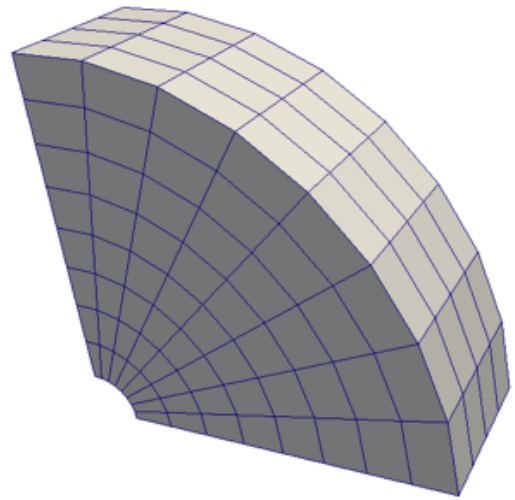
Uniform Rectilinear

A uniform rectilinear grid is a one- two- or three- dimensional array of data. The points are orthonormal to each other and are spaced regularly along each direction.



Non-uniform Rectilinear

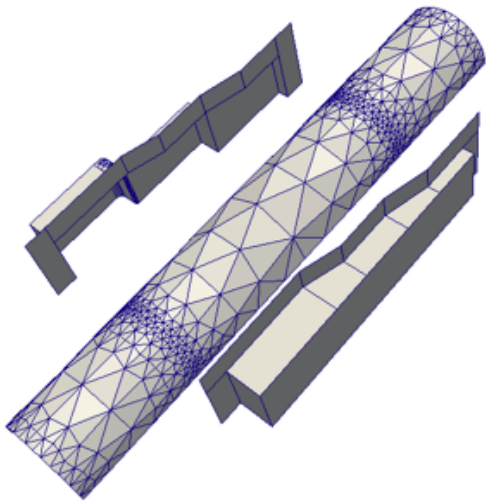
Similar to the uniform rectilinear grid except that the spacing between points may vary along each axis.



Curvilinear

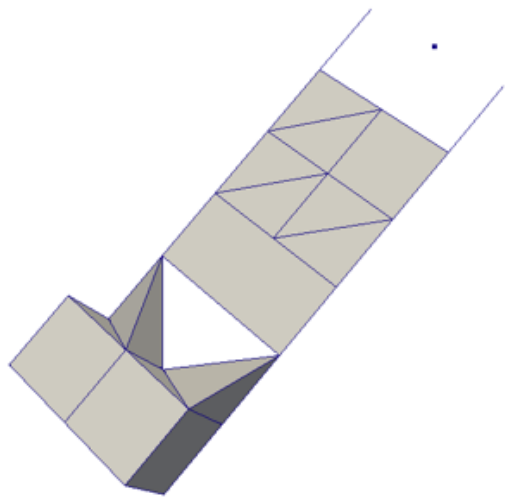
Curvilinear grids have the same topology as rectilinear grids. However, each point in a curvilinear grid can be placed at an arbitrary coordinate (provided that it does not result in cells that overlap or self intersect).

Poly data and unstructured mesh



Polygonal

Polygonal data sets are composed of points, lines, and 2D polygons. Connections between cells can be arbitrary or non-existent.



Unstructured

Unstructured data sets are composed of points, lines, 2D polygons, 3D tetrahedra, and nonlinear cells.

Also:

- multi-block data
- adaptive mesh refinement
- ...

Multiblock familiarisation exercise **here** for:

`Testing/Data/bake/bake.e`

`Testing/Data/can.ex2`

Data filtering

Data filtering is the process of choosing a smaller part of your data set and using that subset for viewing or analysis. Filtering is generally (but not always) temporary – the complete data set is kept, but only part of it is used for the calculation

For scientific visualisation, filtering may be used to

- Look at results for a particular period of time
- Calculate results for particular zone of interest
- Present results in an easily understandable manner
- Extract important and practically useful measurements

ParaView has 183 filters listed here

ParaView data import

There are two ways to get data into ParaView:

- read data from a file, or
- generate data with a source object.

ParaView currently supports about 220 distinct file formats

Exercises: Day 1

For Exercise 2.7, load `disk_out_ref.ex2` from `ParaView-v5.8.0-RC1/Testing/Data/`

Day 2: Understanding data

Chapter 3 of ParaViewGuide-5.8.0.pdf

VTK data model is used by ParaView

Most fundamental data structure in VTK is a data object

These are either:

- scientific datasets, such as rectilinear grids or finite elements meshes
- more abstract data structures, such as graphs or trees

Mesh

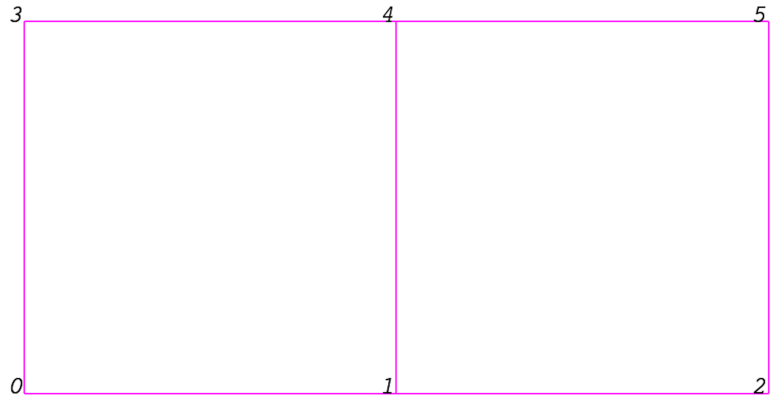
Mesh consists of

- Verticies (points)
- Cells (elements, zones)

Cells are used to discretize a region and can be tetrahedra, hexahedra, etc

Mapping from cells to vertices is called connectivity

Faces & edges not represented explicitly by VTK (only needed for arbitrary polyhedron)



Example of a mesh

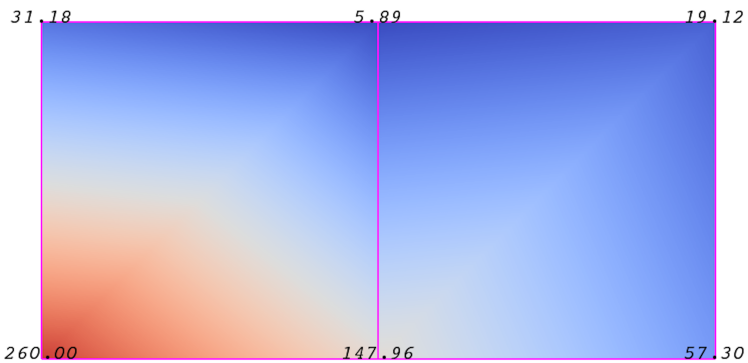
- Cell 1; 0, 1, 3, 4
- Cell 2; 1, 2, 4, 5

Share points 1 4

Attributes (fields, arrays)

Data attributes stored at different mesh locations

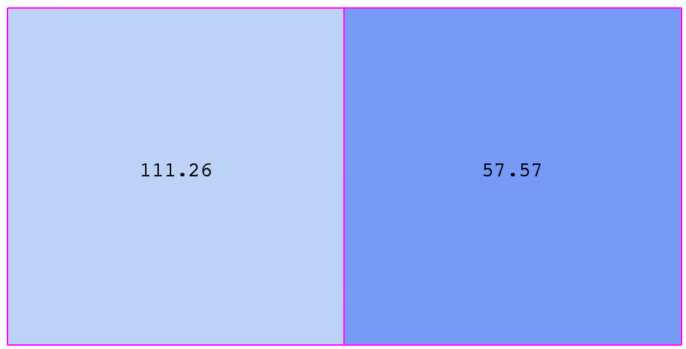
- May depend on calculation, e.g. velocity at vertices and cell centred pressure



Point-centered attribute in a data array or field

Attribute only defined at vertices

- Interpolation used to obtain values everywhere else



Cell-centred attribute

Cell-centered attributes are assumed to be constant over each cell

- Filter may not apply; requires `Cell Data to Point Data` filter

VTK Data file formats

VTK supports five different data-set formats:

- structured points, structured grid, rectilinear grid, unstructured grid, polygonal data

Exercise; Create `file1.vtk`:

```
# vtk DataFile Version 2.0
Structured example
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 3 4 5
ORIGIN 0 0 0
SPACING 1 2 3
```

Now open with ParaView (beware! '_' character doesn't copy-paste)

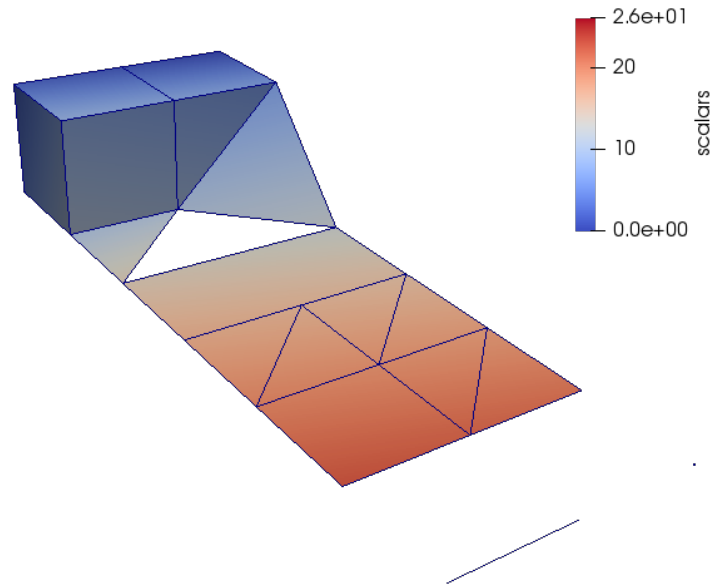
Then add lines:

```
POINT_DATA 60
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 5 10 15 20 25 25 20 15 10 5 0
0 0 0 0 0 0 0 0 0 0 0 0
```

Open in ParaView and check data

Unstructured grid example

```
# vtk DataFile Version 2.0
Unstructured Ex
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 27 float
0 0 0 1 0 0 2 0 0 0 1 0 1 1 0 2 1 0 0 0 1
1 0 1 2 0 1 0 1 1 1 1 1 2 1 1 0 1 2 1 1 2
2 1 2 0 1 3 1 1 3 2 1 3 0 1 4 1 1 4 2 1 4
0 1 5 1 1 5 2 1 5 0 1 6 1 1 6 2 1 6
CELLS 11 60
8 0 1 4 3 6 7 10 9
8 1 2 5 4 7 8 11 10
4 6 10 9 12
4 5 11 10 14
6 15 16 17 14 13 12
6 18 15 19 16 20 17
4 22 23 20 19
3 21 22 18
3 22 19 18
2 26 25
1 24
CELL_TYPES 11
12 12 10 10 7 6 9 5 5 3 1
POINT_DATA 27
SCALARS scalars float 1
LOOKUP_TABLE default
0.0 1.0 2.0 3.0 4.0 5.0 6.0
7.0 8.0 9.0 10.0 11.0 12.0
13.0 14.0 15.0 16.0 17.0 18.0
19.0 20.0 21.0 22.0 23.0 24.0 25.0 26.0
VECTORS vectors float
1 0 0 1 1 0 0 2 0 1 0 0 1 1 0 0 2 0 1 0 0
1 1 0 0 2 0 1 0 0 1 1 0 0 2 0 0 0 1 0 0 1
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
```



Mesh given by `Unstructured Ex` (left)

Details on `CELL_TYPES` given **here** on pp. 9 & 10

Time dependent data

ParaView automatically detects several file naming patterns that indicate a file series, including:

`fooN.vtk` `foo_N.vtk` `fooN.vtk` `foo.N.vtk`
`Nfoo.vtk` `N.foo.vtk` `foo.vtk.N` `foo.vtksN`

where *foo* is any filename, *N* is a numeral sequence, and *vtk* could be any extension

File sequences can be made that will be grouped by ParaView; e.g. `file1.vtk`, `file2.vtk`, ...

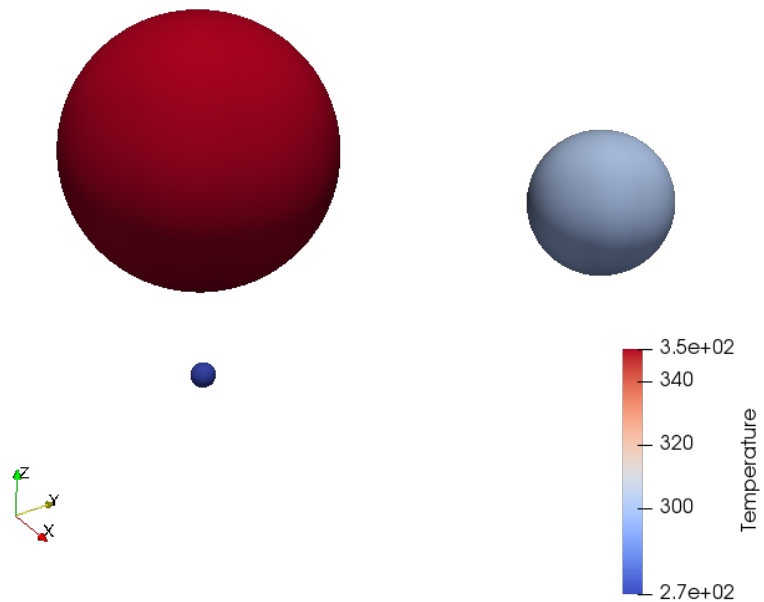
Particle file 1

Create file named `part0.vtu` with:

```
<VTKFile type="UnstructuredGrid" version="2.0" byte_order="LittleEndian">
<UnstructuredGrid>
<Piece NumberOfPoints="3" NumberOfCells="0">
<Points>
<DataArray name="Position" type="Float32" NumberOfComponents="3" format="ascii">
0 0 0 1 1 1 0 0 1
</DataArray>
</Points>
<PointData Vectors="vector">
<DataArray type="Float32" Name="Velocity" NumberOfComponents="3" format="ascii">
4 4 4 4 0 0 2 2 -2
</DataArray>
<DataArray type="Float32" Name="Diameter" format="ascii">
0.1 0.5 1
</DataArray>
<DataArray type="Float32" Name="Temperature" format="ascii">
273 300 350
</DataArray>
</PointData>
<Cells>
<DataArray type="Int32" Name="connectivity" format="ascii">
</DataArray>
<DataArray type="Int32" Name="offsets" format="ascii">
</DataArray>
<DataArray type="UInt8" Name="types" format="ascii">
</DataArray>
</Cells>
</Piece>
</UnstructuredGrid>
</VTKFile>
```

Particle file 2

- Open file in ParaView
- Add glyphs with diameter specified in file (change Properties→Glyph Mode to All Points)
- Colour particles according to temperature
- Copy `part0.vtu` to create `part1.vtu` with different particle coordinates/temperature/diameter



Particles

Other file formats

STL ("stereolithography") file, e.g. **Utah teapot**

- Describe only the surface geometry of a three-dimensional object
- Unstructured triangulated surface

Exodus Element Block Specification (e.g. `.ex2`)

- **CUBIT** Geometry and Mesh Generation Toolkit

Binary files

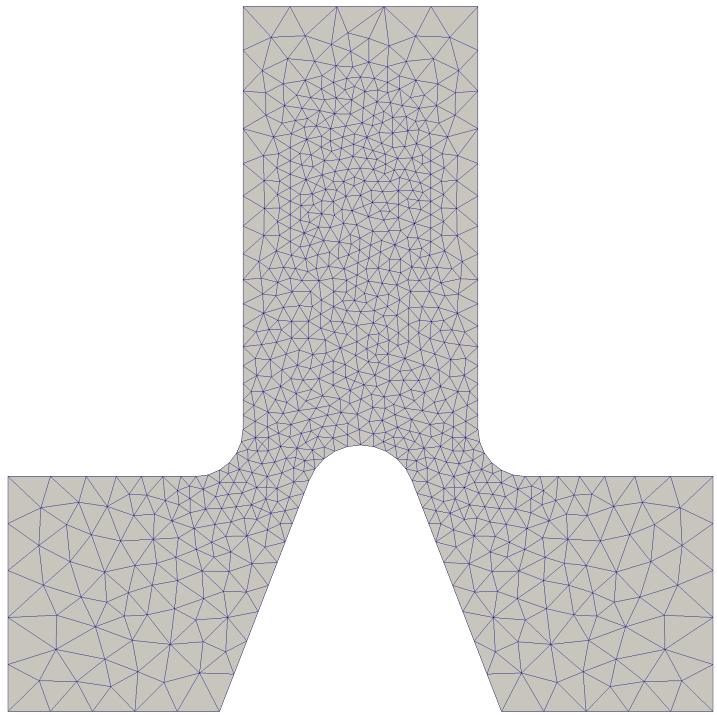
- More compact and quicker to read
- Binary versions of most file formats

Grid generation: Gmsh

Normally, a mesh is generated by software

For example, **Gmsh** open-source finite-element mesh generator. After installing Gmsh:

1. Download script **t4.geo**
2. run command `$ gmsh -2 -format stl t4.geo`
3. open `t4.stl` in ParaView



Mesh generated with `Gmsh`

Computer simulation: OpenFOAM

OpenFOAM is free, open source software for CFD

Can be obtained on Ubuntu using :

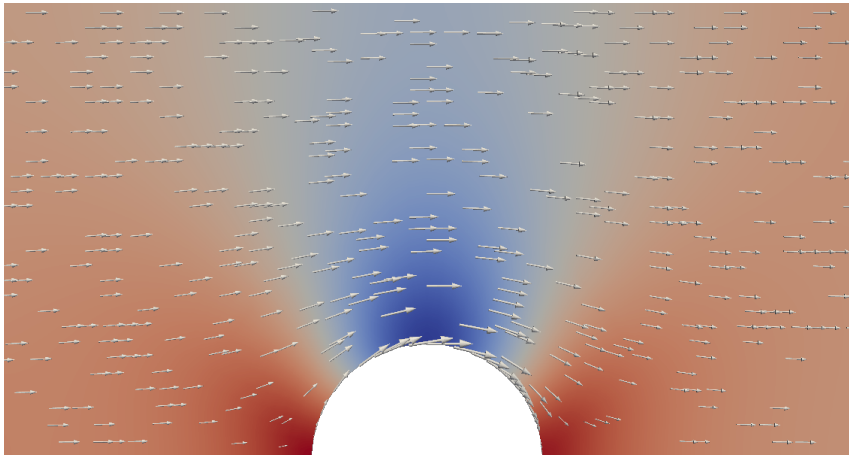
```
$ sudo apt-get -y install openfoam7
```

Tutorial examples including cylinder potential flow (right)

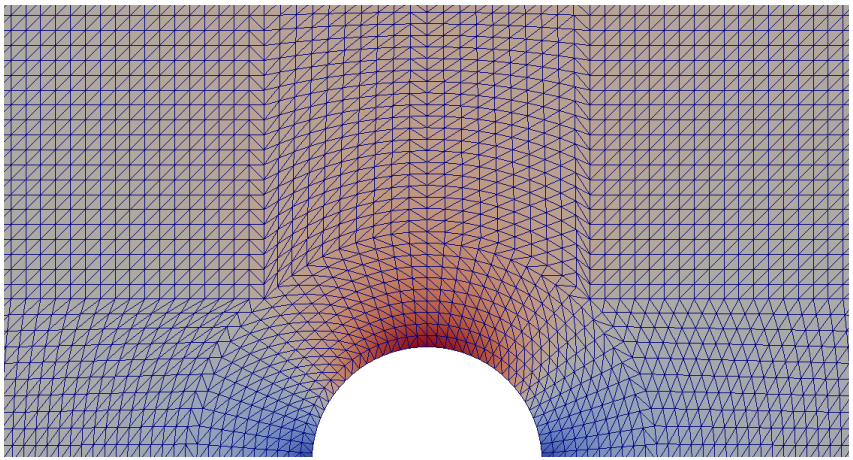
```
tutorials/basic/potentialFoam/cylinder
```

Version of ParaView launched from command line:

```
$ paraFoam
```



Pressure and glyphs



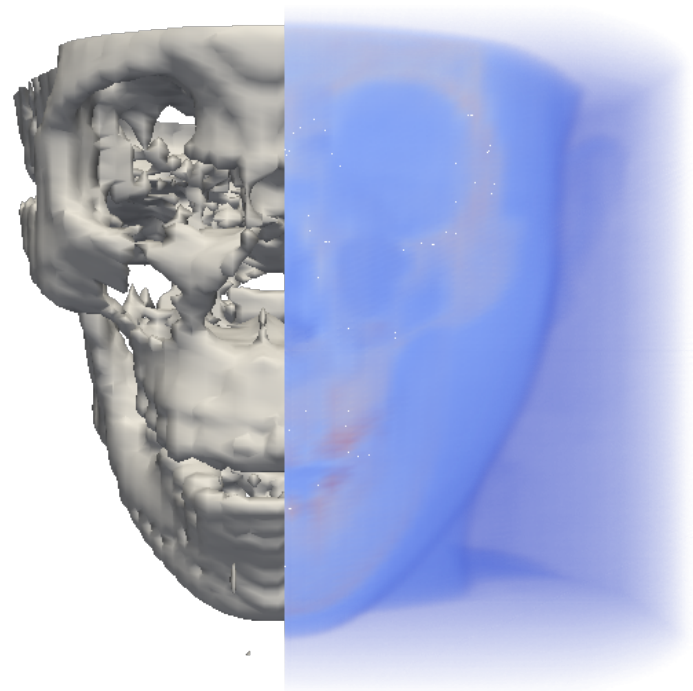
Velocity and computational grid

Cylinder potential flow

Medical data: Head scan

3D image file. Download
headsq.vti

- VTKFile with XML formats
- Binary image data



Data from head scan

Exercises: Day 2

For Exercise 2.19, load `can.ex2` from `ParaView-v5.8.0-RC1/Testing/Data/`

Exercises: Day 3

To prepare for the evaluation exercise:

- Use data (ParaView examples, internet, simulation software, e.g. OpenFOAM tutorials,...)
- Create images, videos, etc then put in document (good formatting, e.g. correct font sizes, etc)
- Create plots / analysis to show interesting behaviour or phenomena

Work on several data-sets throughout the day. Discuss the results in groups and show me what you have found. Ideas for data sources:

- (morning) ParaView example data
  - AMR `amr/spcth.0`
  - Aerofoil `EnSight/naca.bin.case`
  - Iron protein(?) `Iron_Xdmf/Iron_Protein.ImageData.Collection.xmf`
  - **3GQP** protein `/3GQP.pdb`
- (afternoon) OpenFoam tutorials **here**, other 3D simulation software
- Other possibilities; create sources to demonstrate animations, mathematical geometric data, ...

OpenFOAM data

1. **Cavity flow tutorial;** download `cavityVTK.zip` (with documentation **here**) and open in ParaView, then apply filters (glyph, streamline, plot velocity profile along centre line). After output images and use in a document to make descriptions of the flow
2. **Dam break tutorial;** download `damBreakVTK.zip` and open in paraView. Then create an animation and use to describe what is happening in the simulation, then find the maximum pressure on the right hand wall : **damBreak**

Batch scripting: Day 4

Python scripting can be leveraged in two ways within ParaView:

1. Automate the setup and execution of visualisations by performing the same actions as a user at the GUI
2. Run inside pipeline objects, thereby performing parallel visualisation algorithms

Good way to automate mundane and repetitive tasks

Critical component when using ParaView in situations where the GUI is undesired or unavailable

Python scripting to establish *in situ* computation within simulation code, i.e. script included in simulation code

Scripting allows access to much wider range of capabilities

Python

From **Python.org**: *"Python is a programming language that lets you work quickly and integrate systems more effectively"*

Open-source and freely available

Good for beginners and those experienced with other languages

Both Python's standard library and the community-contributed modules allow for endless possibilities. Libraries include:

- **NumPy** is the fundamental package for scientific computing
- **matplotlib** is a comprehensive library for creating static, animated, and interactive visualisations

Runs from command line and can be scripted

- Basic demonstration

Scripting approaches

Python Shell Find in `Tools` → `Python Shell`

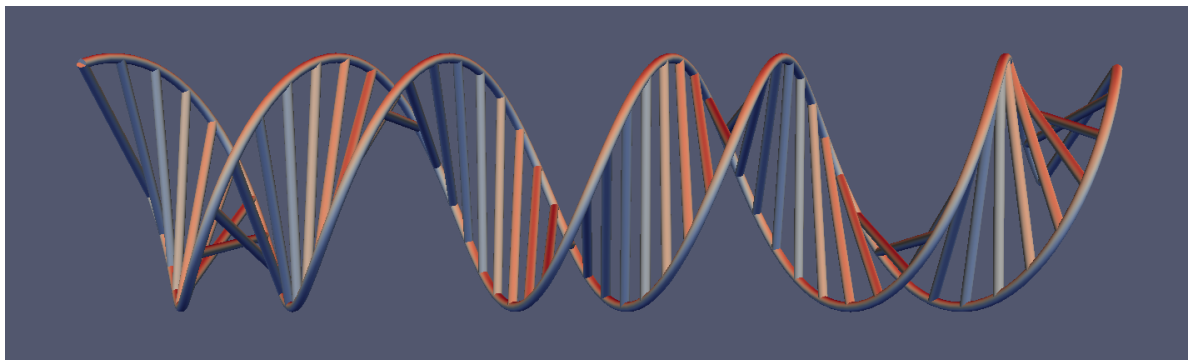
pvpython Runs on command line for interactive scripts

pvbatch Runs on the command line for batch processing. Can be run in parallel

pvserver For remote visualization, useful on HPC resources

There are also programmable sources and filters

- Try examples given **here**, adjust the code to see what changes



Double helix created with programmable source

Tracing actions for scripting

For tracing actions in the UI as a Python script

- `Tools` → `Start Trace`
- Execute a series of actions in the UI
- `Tools` → `Stop Trace`

Produces a python script that corresponds to performed actions

Save script and use for batch processing

Good way to discover ParaView python commands and syntax

Exercises: Day 4

Exercises in chapter 3 of the ParaView tutorial on scripting

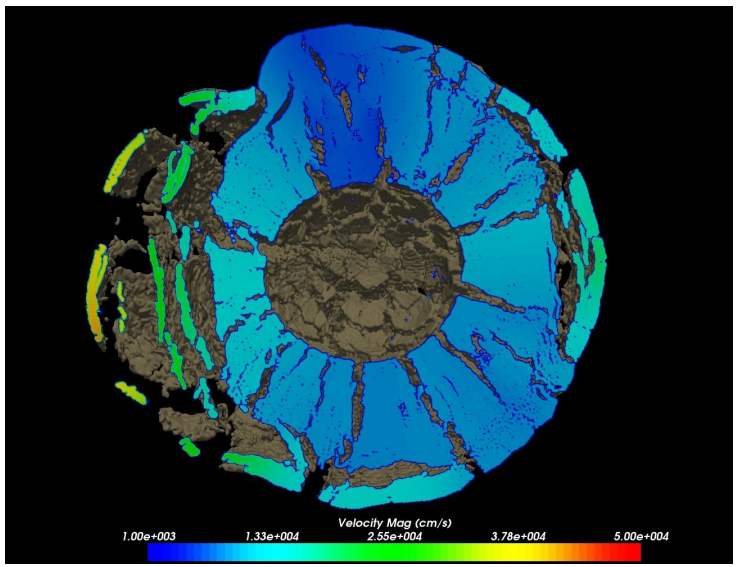
Create a script that outputs an annotated visualisation including filters, and a plot over line, from day 4 OpenFoam excercises, e.g. cavity flow tutorial. After, run this in a terminal using $ pvpython script.py (note, may be necessary to put Interact() at the end of the script)

Other scripting possibilities

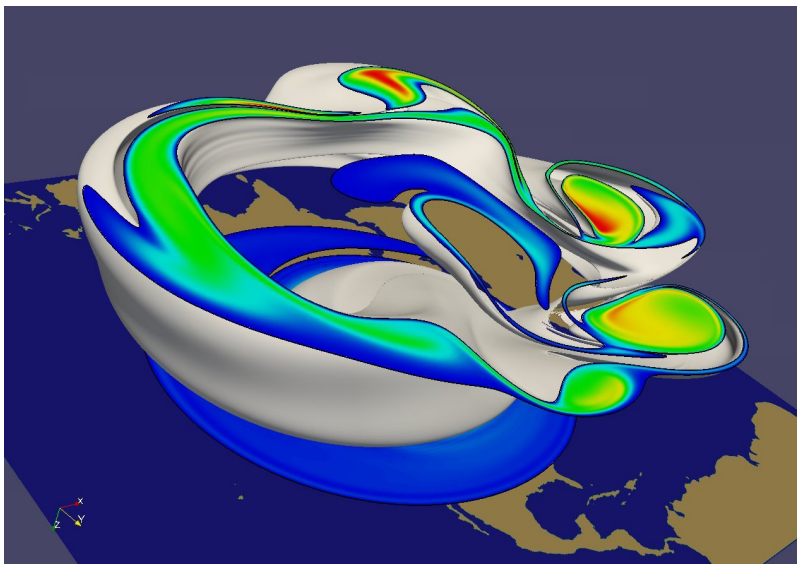
- PyVista 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)
  - No GUI
  - Lighter than ParaView
  - Some filters that are not included in ParaView, e.g. $\text{delaunay}_2 d$ to construct mesh from data points
- VTK in python
  - No GUI, hence necessary to develop script from online examples and python commands
- VTK in c++
  - No GUI, need for make/cmake files
  - Allows access to whole VTK library
  - Useful for full integration into simulation code, e.g. Liggghts

Visualising large models

ParaView used at institutions around the world to visualise data from large-scale simulations run on the world's largest supercomputers

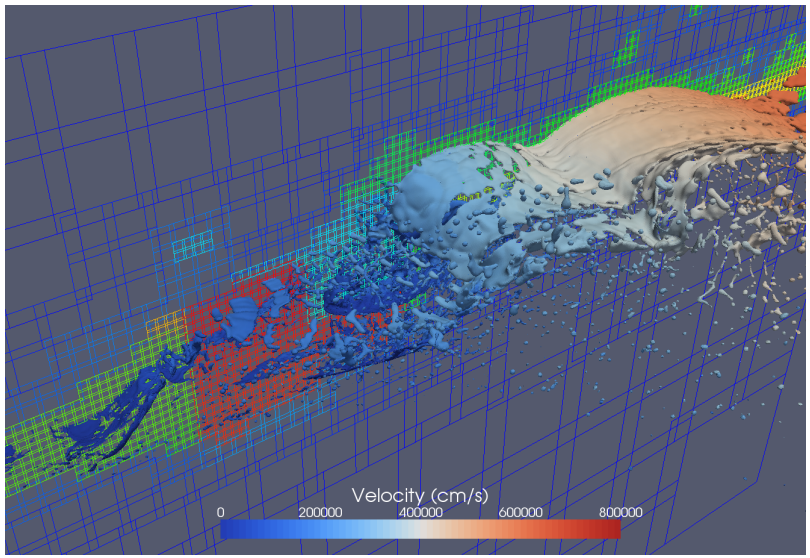


CTH shock physics simulation with over 1 billion cells of a 10 megaton explosion detonated at the centre of the Golevka asteroid

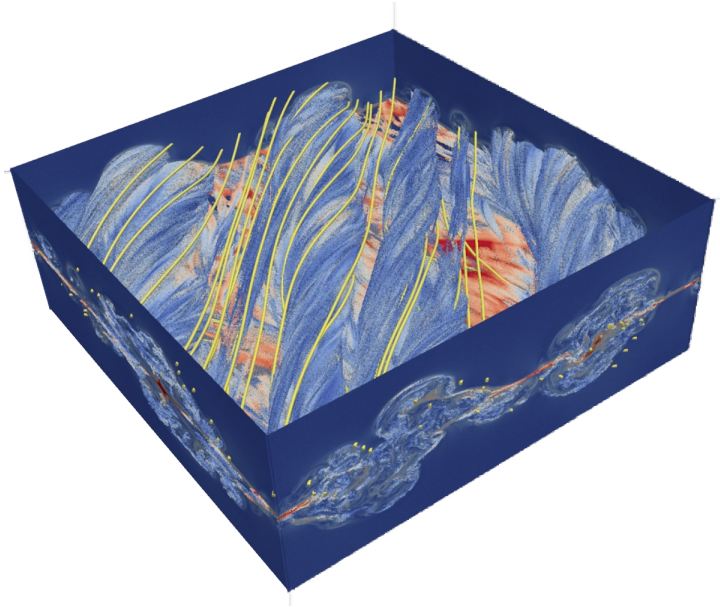


SEAM Climate Modeling simulation with 1 billion cells modeling the break-down of the polar vortex, a circum-polar jet that traps polar air at high latitudes

Visualising large models

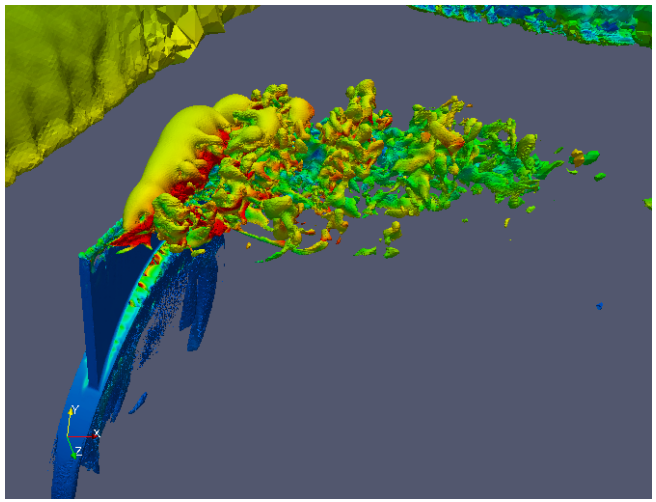


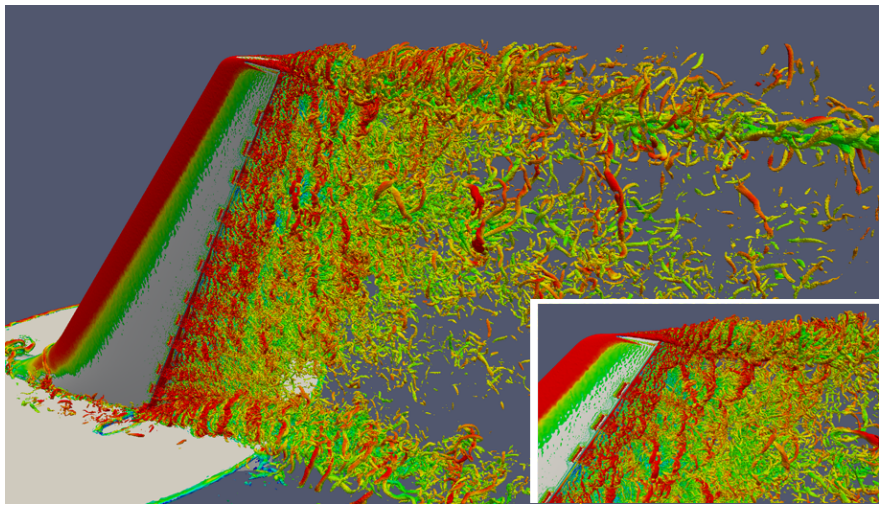
A CTH simulation that generates AMR data. ParaView has been used to visualize CTH simulation AMR data comprising billions of cells, 100's of thousands of blocks, and eleven levels of hierarchy (not shown)



A VPIC simulation of magnetic reconnection with 3.3 billion structured cells

Visualising large models





ParaView visualizations run in situ with large scale **PHASTA** simulations

Left 3.3 billion tetrahedral mesh simulating the flow over a full wing where a synthetic jet issues an unsteady crossflow jet (run on 160 thousand MPI processes)

Right is a 1.3 billion element mesh simulating the wake of a deflected wing flap (run on 256 thousand MPI processes)

Exercise

Read parallel visualization algorithms and parallel rendering in Chapter 4 of the tutorials

Paraview tutorial exercise 4.1

Evaluation

Evaluation exercise using scientific data:

- Choose from OpenFOAM tutorials given here. All students must use different data
- Create images, videos, etc then put in document (good formatting, e.g. correct font sizes, etc)

Marks awarded for:

3-D	visualisations using filters
Line plots	extracted from 3-D data
Animation	showing instructive behaviour
Description	of interesting behaviour
Formatting	i.e. layout, colour choices, font sizes, etc

Evaluation hints

Add axis labels to line charts.

Add annotation to movie; e.g. time, inlet parameters, etc.

Best marks for comparisons involving superimposed statistics