# Introduction to ParaView

Raphael Münster
December 7, 2017

TU Dortmund

technische universität
dortmund

fakultät für
mathematik

# About ParaView

## ParaView Features

- Open-source and multi-platfrom visualization software
- Visualization backend provided by VTK library
- Huge number of visualization filters
- Extension is possible by programmable filters (Python) or
- User defined plugins
- Import and export of data to various formats used in CFD packages
- Easy access to numerical data for internal or external plotting etc.
- Tasks can be automated by using the PvPython interface
- Client/Server paradigm to view Big Data on remote clusters

# ParaView Software Ecosystem

**CMake**
- Script language to control the building process
- Generates a wide range of specific build files

**VTK**
- Visualization backend
- Uses OpenGL

**Qt (cute)**
- Provides widges and other GUI controls
- Support for modular plugins

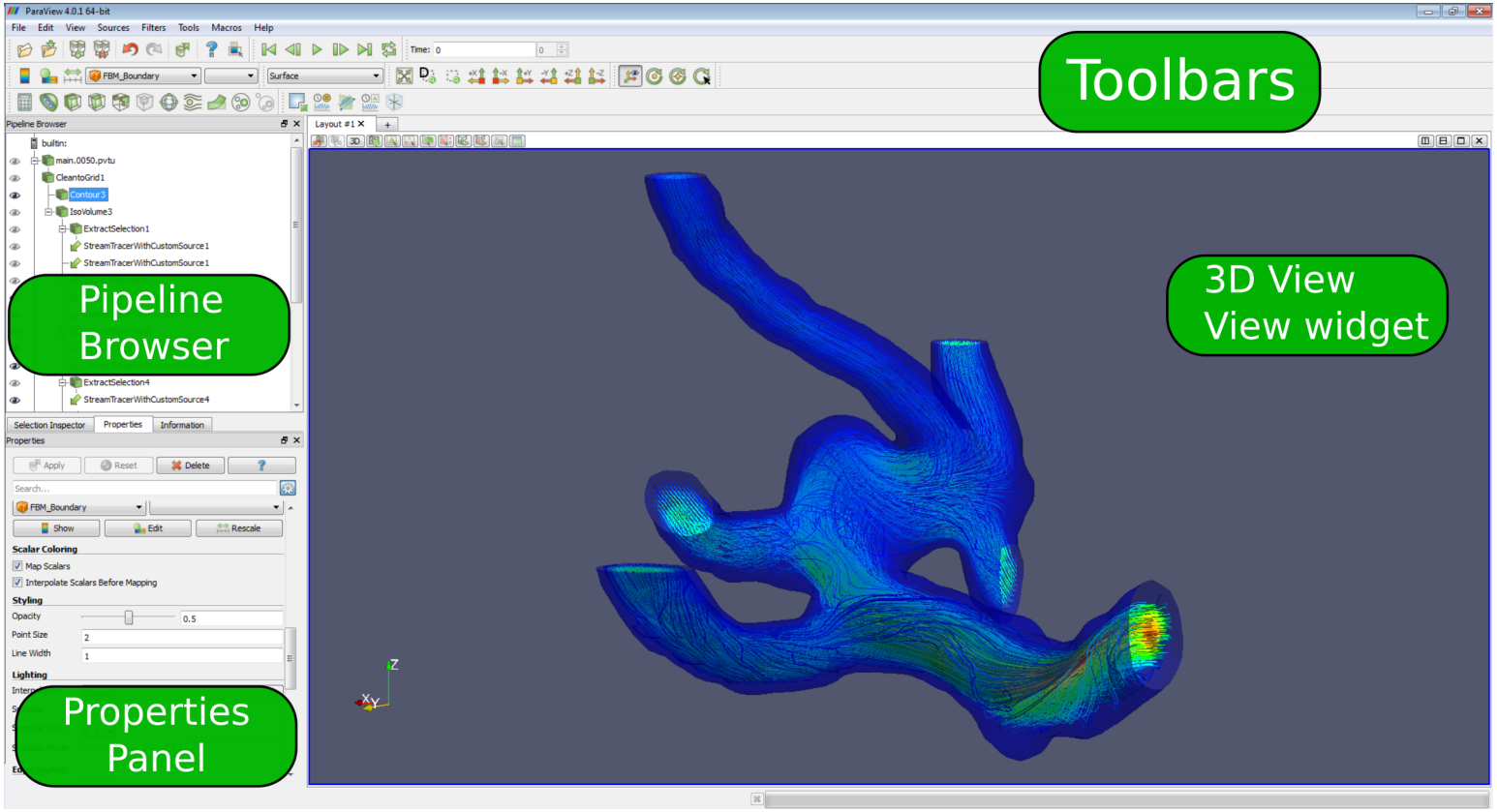
**ParaView**
- VTK visualization by GUI
- Scripting via Python
- Extension by plugins

# Use Cases of Visualization Software

## Typical ParaView Use Cases

- Provide an intuitive illustration of raw simulation data
- Highlight key features of specific flow features
- Provide an intuitive understanding for non-expert viewers
- Help in the testing/debugging process of CFD software
- Assist during result validation by providing analysis tools
- Convert data in different formats in order to communicate with scientific partners

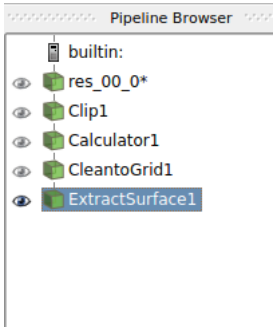
# ParaView Interface

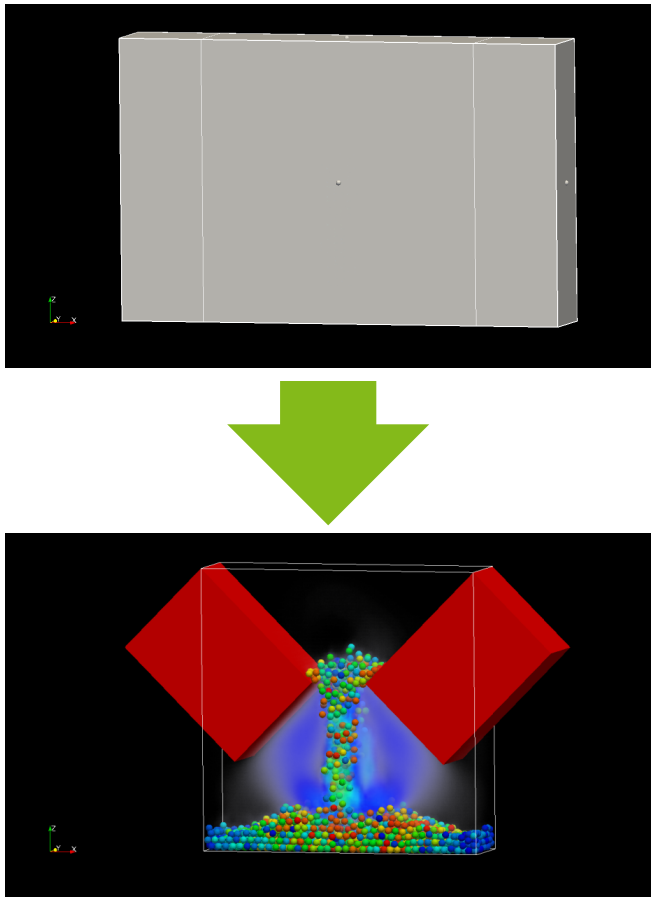


# Filter Pipeline Concept

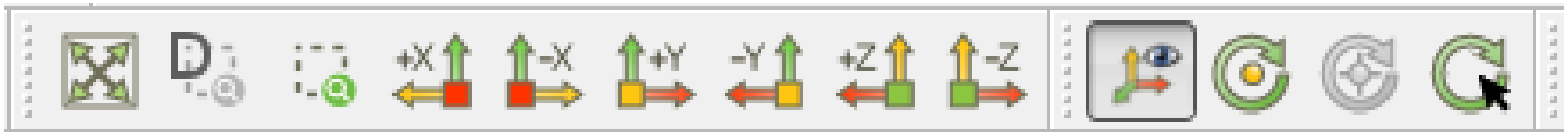
## Filter Pipeline

- A `filter` is an operation on an input data set
- Filters can be chained (pipelined)
- More complex visualizations require multiple filters



# Camera and Axes Toolbar



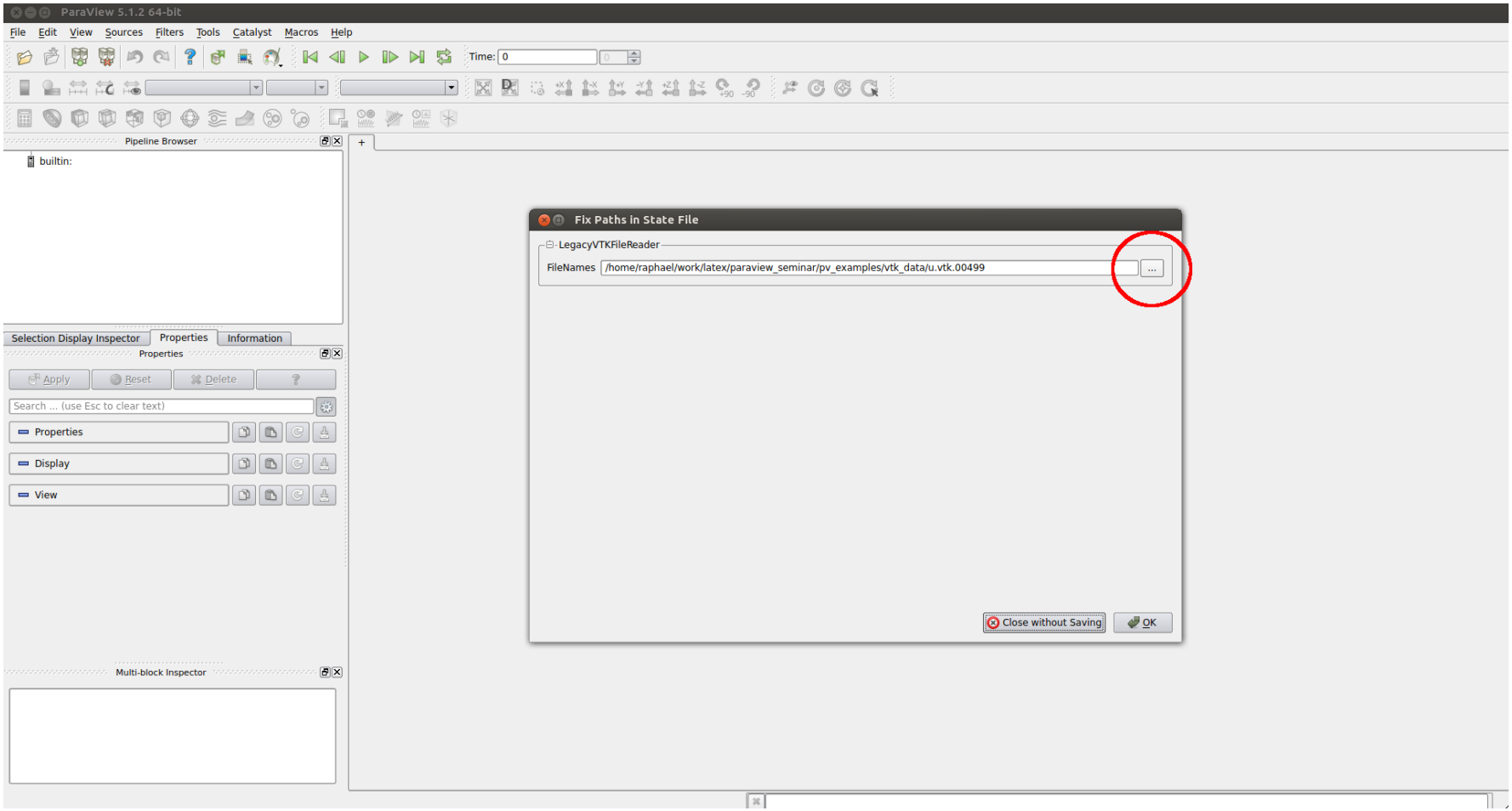

- Reset the camera to its default parameters
- Magnify a rectangular selection of the view area
- Choose a certain coordinate axes plane to look at
- Toggle rendering of orientation axes
- Toggle rendering of center of rotation
- Select a new center of rotation

# Basic Filters

- `Slice`: Extract a plane, box-, sphere- or cylinder surface out of a data set
- `Clip`: Extract a plane, box-, sphere- or cylinder volume out of a data set
- `Warp by Scalar`: Visualizes a scalar value by a height extrusion on a 2D data set
- `Contours`: Increases visibility of different solution contour levels
- `Surface LIC`: Streamlines on pixel basis, highlights small scale flow features
- `Glyphs`: Streamlines on pixel basis, highlights small scale flow features
- `Stream Traces`: Visualizes the pathline of a particle through a *stationary* vector field

# State Files

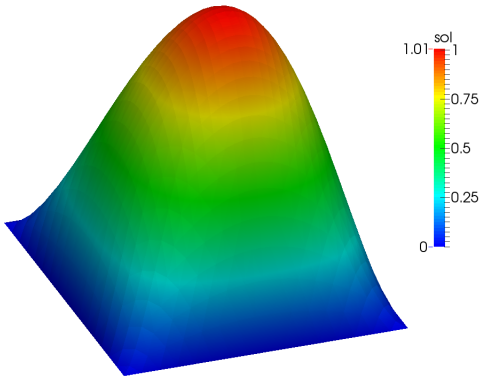
- `State file`: XML format based file with the ending .pvsm that is used to store the currently applied filters and a reference to the currently loaded data sets to a file
- Used to quickly restore a state or to recover from a crash
- The reference to the data set can be changed upon loading the state file in order to apply the filters to a different data set or if the location of the data on the hard drive has changed
- This way state files can be used to exchange a ParaView visualization with collaborators, they only need to set the location of their data set upon loading the state file
- State files can be exchanged between different version of ParaView
- Accessible from Menu: `File->Load State...`

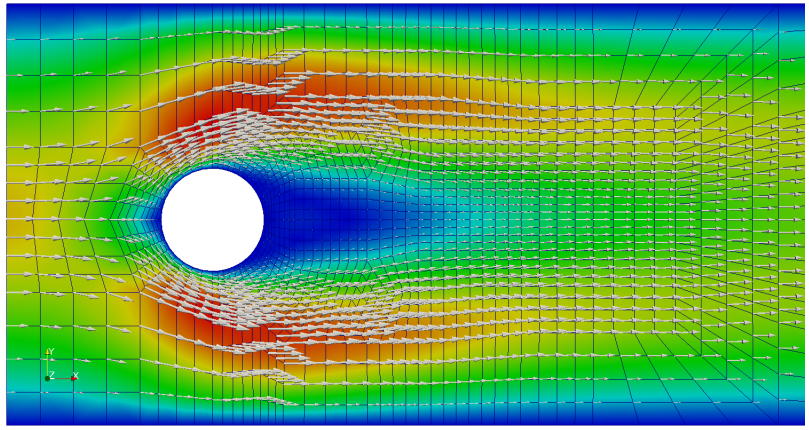

ParaView 5.1.2 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Time: 0
Pipeline Browser
builtin:
Selection Display Inspector
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Properties
Display
View
Multi-block Inspector
Fix Paths in State File
LegacyVTKFileReader
FileNames
/home/raphael/work/latex/paraview_seminar/pv_examples/vtk_data/u.vtk.00499
...
Close without Saving
OK

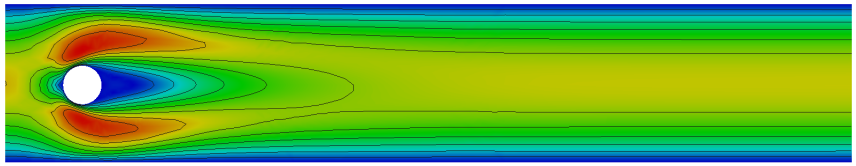
# Examples Files

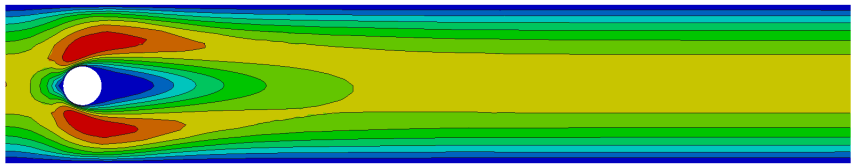
- ParaView examples are provided as state files
- Download archive of ParaView examples from:
- Extract archive to `/mypath/to/examples/`
  ```
  > tar xvfz pv_examples.tar.gz -C /mypath/to/examples/
  ```
- Basic filter examples are located in `/mypath/to/examples/pv_examples`
- Basic filter examples are located in `/mypath/to/examples/pv_examples/BasicFilters`
- To load a basic filter example, load the state file and set data path to: `/mypath/to/examples/pv_examples/vtk_data/u.vtk`

# Basic Filter Gallery I

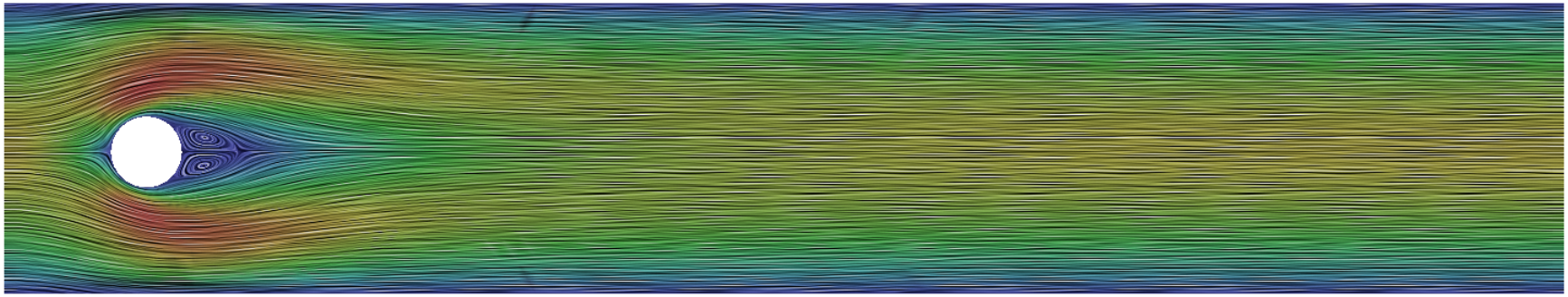


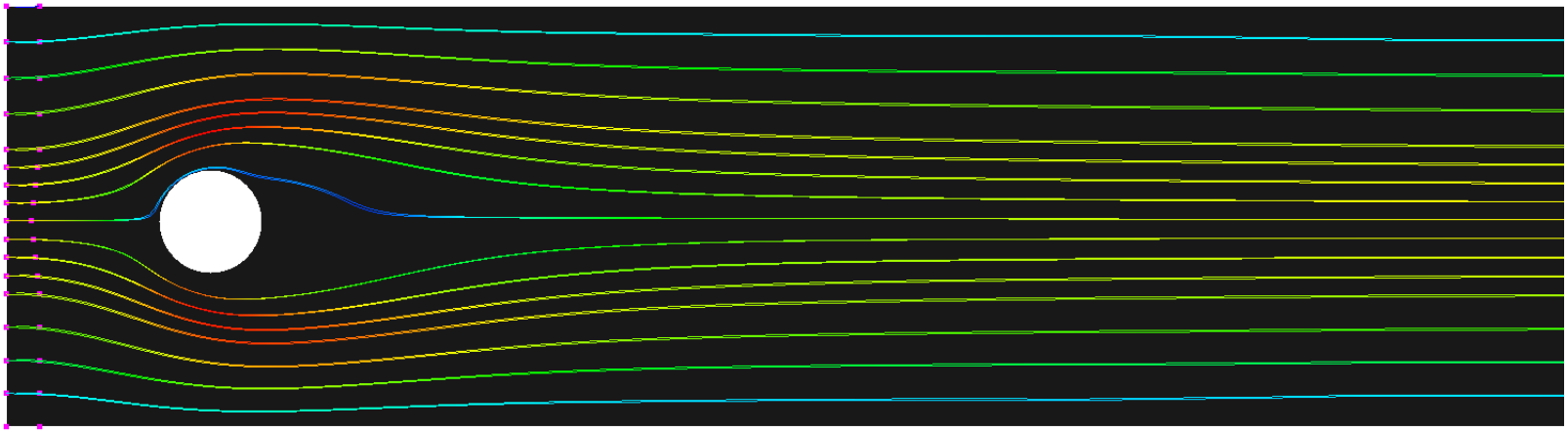







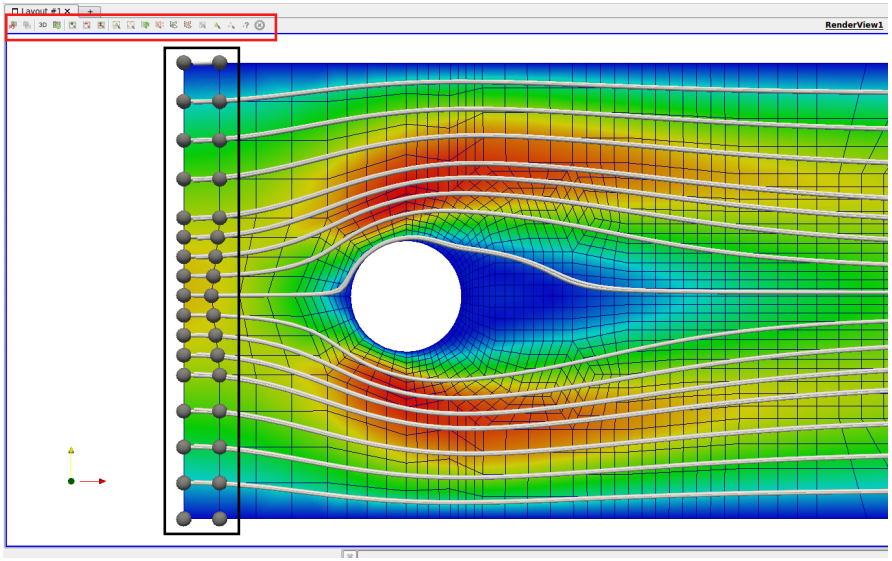
# Basic Filter Gallery II





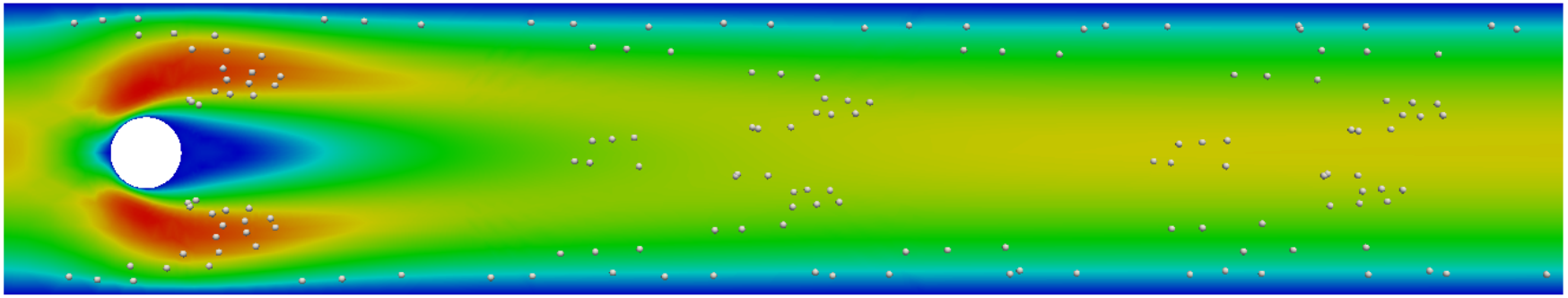
# Configuration of Tracer Filters

- Tracer type filters need an `input data set` and a `seed source`
- The input data set is the the flow field
- The seed source is a user-defined starting location for the particles inside the flow field



# Advanced Filters: Particle Tracer

- Used to visualize transient flow data by particles
- A particle path is produced by moving a particle along successive vector fields
- Particle movement can then be animated by the `Animation Control`
- Particle tracer example location:
  `/mypath/to/examples/pv_examples/AdvancedFilters/particle_tracer`



# List of Useful Filters

- `Extract Selection`: Extract a subset from a data set (tracer source, plotting, etc)
- `Extract Surface`: Generates a polygonal surface mesh (exporting, rendering, remeshing)
- `Clean to Grid`: Removes multiply defined simplices and converts to unstructured mesh
- `Integrate Variables`: Performs numerical integration of the fields defined in the data set on the mesh
- `Iso-Volume or Iso-Surface`: Constructs a volume or a surface from a function defined on the mesh
- `Plot Over Line`: Generates a plot of data fields along a line (inflow profiles, etc.)
- `Calculator`: Use a set of mathematical operations to compute a new data field from existing ones (often used together with integrate variables filter)

# Input Data Formats

- VTK data formats support polygonal data sets and structured or unstructured meshes
- `VTK Legacy format .vtk` is a simple format useable for unstructured meshes (not suited for distributed data)
- `VTK unstructured .vtu` is an xml based format for unstructured meshes
  - Can have the mesh data part encoded in a binary format
  - A `.pvtu` file can be used to reference the partial solutions of a distributed computation
  - A `.pvd` file can be used to add time information
- For further information on VTK file formats see:
  `http://www.vtk.org/VTK/img/file-formats.pdf`

# Scripted Postprocessing in ParaView

- Python interface to access ParaView functionality by scripts
- Can be done interactively via a python shell: `Tools->Python Shell`
- In a 'batch' style by passing a script to the executables `pvpython` or `pvbatch`
- Documentation of the ParaView Python interface is under construction at: `https://www.paraview.org/ParaView3/Doc/Nightly/www/py-doc/`
- A trace mechanism is available to generate a PvPython script from a sequence of actions
- Beginners should use the trace mechanism (`Tools->Start Trace`) with the settings  and
- Upon `Tools->Stop Trace` a file is generated showing the PvPython script equivalent of the user's GUI actions

# PvPython Simple Example

- PvPython simple example location:
  `/mypath/to/examples/pv_examples/PvPython/paraview_python`
- Navigate to the folder and execute the PvPython script by:
  `pvbatch ./python_test.py $(pwd)` or for versions higher than 5.1
  `pvbatch --use-offscreen-rendering ./python_test.py $(pwd)`
- The script will write an image `res.png` to the directory where you executed it
- `Exercise:` Try to recreate the python script using the trace mechanism, compare the output images if they are the same.
- `Hint:` Prefer `pvbatch` over `pvpython` as pvpython tries to open an X windows which may fail on some computers

# When to use PvPython

- Repeated application of filters to a lot of different data sets (parameterize script w.r.t. data set)
- Repeated application of operations that cannot be parametrized by ParaView GUI
- Perform an operation that cannot easily be done by ParaView filters
- Quickly and repeatedly generate an ouput of a running simulation
- Generate outputs on remote clusters
- `ParaView Programmable Filter` or `Python Calculator` may serve as an alternative

# Plotting with ParaView

- PV plotting example:
  `/mypath/to/examples/pv_examples/Plotting/plotting.pvsm`
- Common ParaView plotting filters:
  - `Plot Data`
  - `Plot Over Line`
  - `Plot Selection Over Time`
- Data can be exported to .csv to use in your favorite plot generator `File->Save Data`
- ParaView will by default export ALL data fields to the .csv file (even those that you do not want or need for the plot)
- `Solution 1:` use <awk> to select the data columns you want:
  `awk -F ',' '{print $1 " " $4}'` (extract first and fourth column)
- `Solution 2:` Remove unneccessary fields before export:
  `/mypath/to/examples/pv_examples/Plotting/plotting2.pvsm`

# Client-Server Mode

- In default mode ParaView is both the client and the server
- When client/server are different rendering and data processing can be handled by different computers
- Simple X forwarding works adequately only if the network speed is fast
- Client-Server is preferable to access data on remote (non-local) clusters
- Client-Server steps: `port forwarding`, `starting the remote server`, `connecting the client to the server`

## Port Forwarding

- Establish an ssh tunnel to forward the local port to the remote server:

```
> ssh lidong1.itmc.tu-dortmund.de \
-L 11111:lidong1.itmc.tu-dortmund.de:11111
```

## Start the remote Server

- Start a ParaView data server on the remote machine

```
> pvserver --server-port=11111 --use-offscreen-rendering
```

## Connect to the remote Server

- Start a ParaView client locally
- Press the <Connect> button on the toolbar
- Manually configure the Server dialog:
  - Name: myname
  - Server Type: Client/Server
  - Host: localhost
  - Port: 11111

Pitfalls:

- You **have to** make use the same ParaView version of the client and the server
- Check that the port is not occupied, otherwise use a different port:

```
> lsof -i:11111
```

# Additional ParaView Resources

- ParaView documentation:
  `https://www.paraview.org/documentation/`
- ParaView Wiki:
  `https://www.paraview.org/Wiki/ParaView`
- ParaView Tutorial:
  `https://www.paraview.org/Wiki/The_ParaView_Tutorial`
- ParaView Mailing List:
  `https://public.kitware.com/mailman/listinfo/paraview`
- ParaView Catalyst:
  `https://www.paraview.org/Wiki/ParaView/Catalyst/Overview`
- ParaView Web JavaScript:
  `www.paraview.org/Wiki/ParaViewWeb_JavaScript_Introduction`

# Happy ParaViewing