

# ParFlow Visualization with ParaView

Kitware, Inc.



# Overview

- Data types & filter pipelines in VTK & ParaView 20 min
- Visualization is composition 20 min
- Tools for ParFlow 45 min
- Going further with Python and NumPy 45 min
- Catalyst and web demonstrations 20 min



# What are VTK & ParaView

- The Visualization Tool Kit (VTK)<sup>1</sup> is a set of libraries.
- ParaView<sup>2</sup> is an application built using VTK.
- Both are freely licensed (modified BSD) and maintained by Kitware via government grants and some of our commercial customers.
- C++, Python, and JavaScript.
- ParaView Guide: <https://paraview.org/paraview-guide/>

1. <https://vtk.org/>

2. <https://paraview.org> ← If you are following along on your own computer, download ParaView from here now.

# Data Types & Filter Pipelines

- Data type refers to how it is stored:
  - The underlying primitive types (string, int, float, ...)
  - The structure of the values and the corresponding geometric interpretation.
- The basic unit of data in VTK is an **array**.
- The basic unit of structure in VTK is **field data**.

# Data Arrays

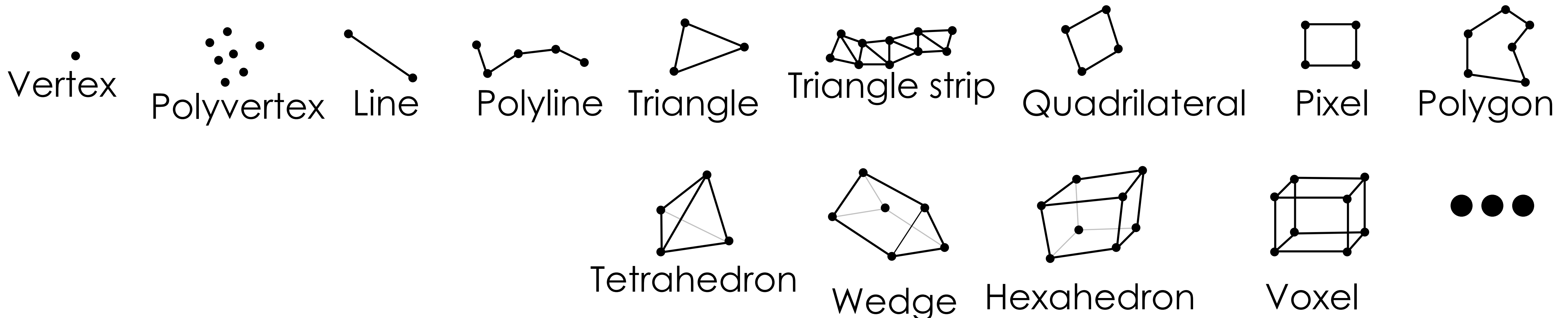
- An array in VTK is
  - an ordered collection of ***tuples***;
  - each *tuple* has the same number of ***components***;
  - each component is a number or string of the same primitive type.
- VTK arrays may also be interpreted as matrices.

# Data Types & Filter Pipelines

- Field data holds any number of arrays.
- A ***data object*** holds a single field data. It has no geometric interpretation; it is up to the application or user to add meaning.
- A ***data set*** has a concept of *points* and *cells* defined by those points. Each concept has associated field data (arrays corresponding to points and cells), plus the unassociated field data inherited from data object above.

# Points & Cells

- Cells are defined by points that serve as corners or shape controls.
- Multiple cells can reference the same point along shared boundaries.



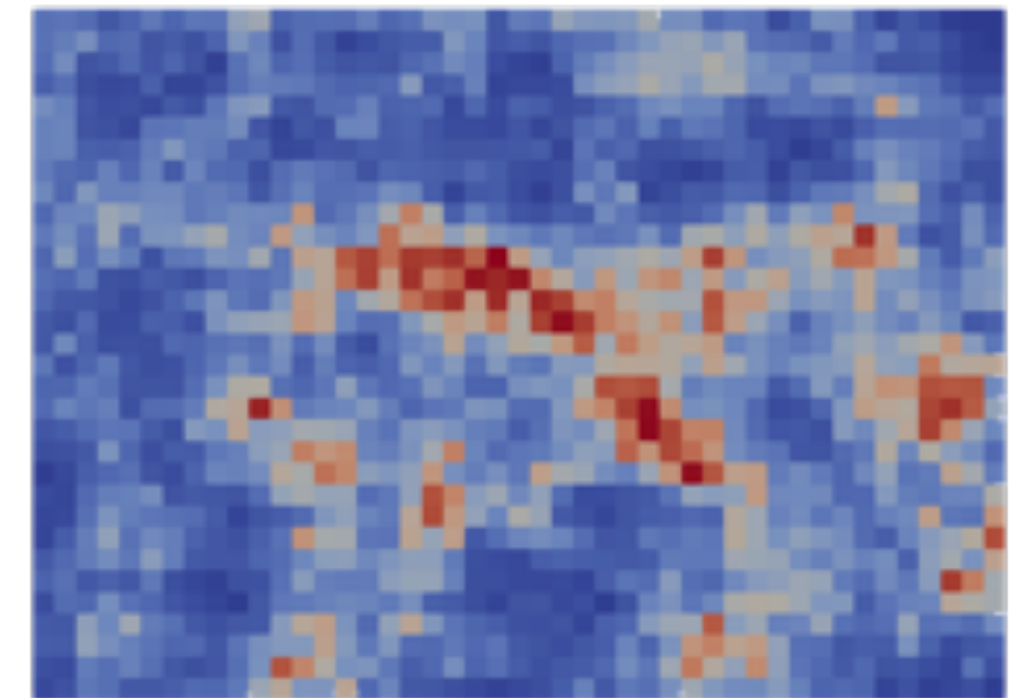
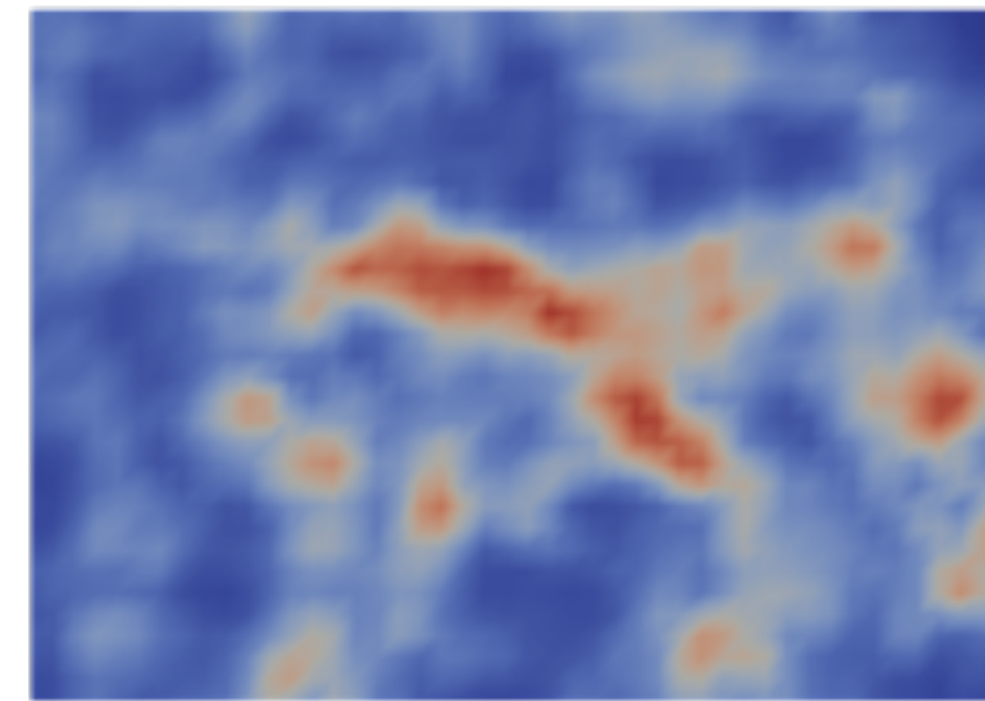
# Data Types & Filter Pipelines

- Points and cells may be implicitly or explicitly represented.
- Implicit points & cells: ***image data***.
- Explicit points, implicit cells: ***rectilinear data*** and ***structured (curvilinear) data***.
- Explicit points & cells: ***polydata*** and ***unstructured grids***.



# Data Types & Pipelines


- Field data defined on points and cells is treated differently:
  - point data is interpolated,
  - cell data is constant.
- Nearly all ParFlow data is cell-centered.



# Data Types & Filter Pipelines

- A ***filter*** constructs, transforms, or consumes data.
- Filters may be *chained* together: one filter's output can serve as another filter's input.
- Asking a filter for its output may cause the filter's upstream source filters to run, followed by the filter itself running.
- This lazy execution avoids unnecessary work.

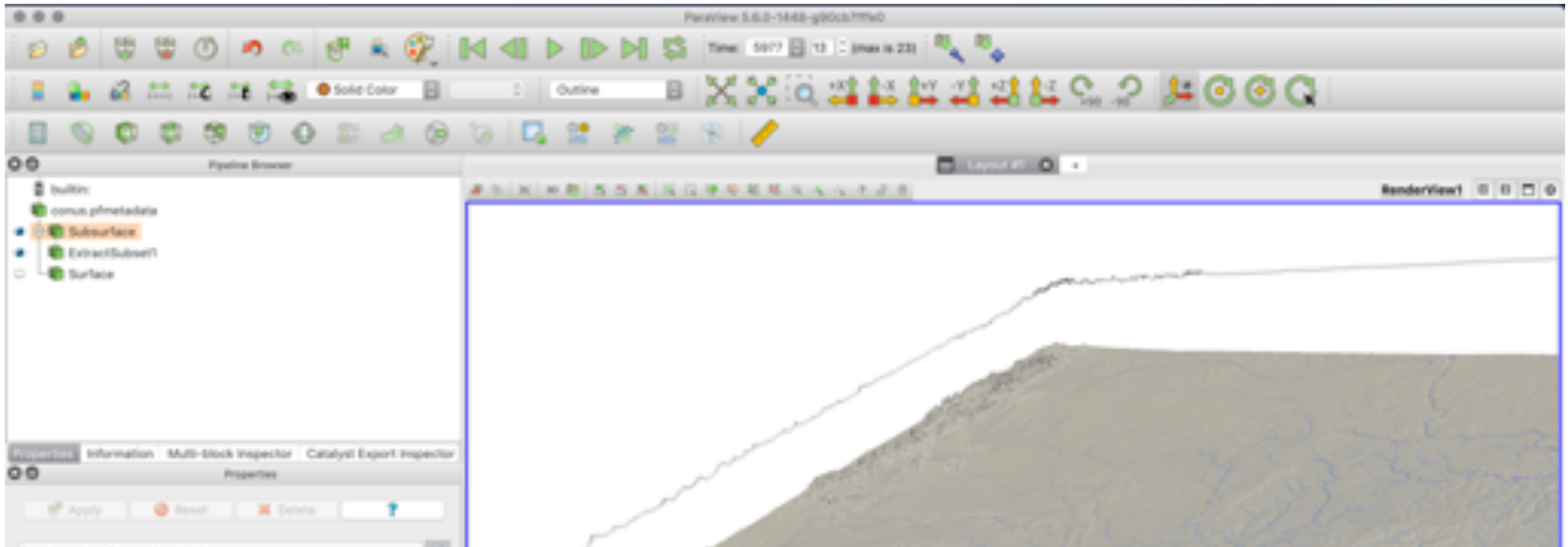
# Data Types & Filter Pipelines

- A **filter** constructs, transforms, or consumes data.  


source                      filter                      sink
- Filters may be *chained* together: one filter's output can serve as another filter's input.
- Asking a filter for its output may cause the filter's upstream source filters to run, followed by the filter itself running.
- This lazy execution avoids unnecessary work.

# ParaView

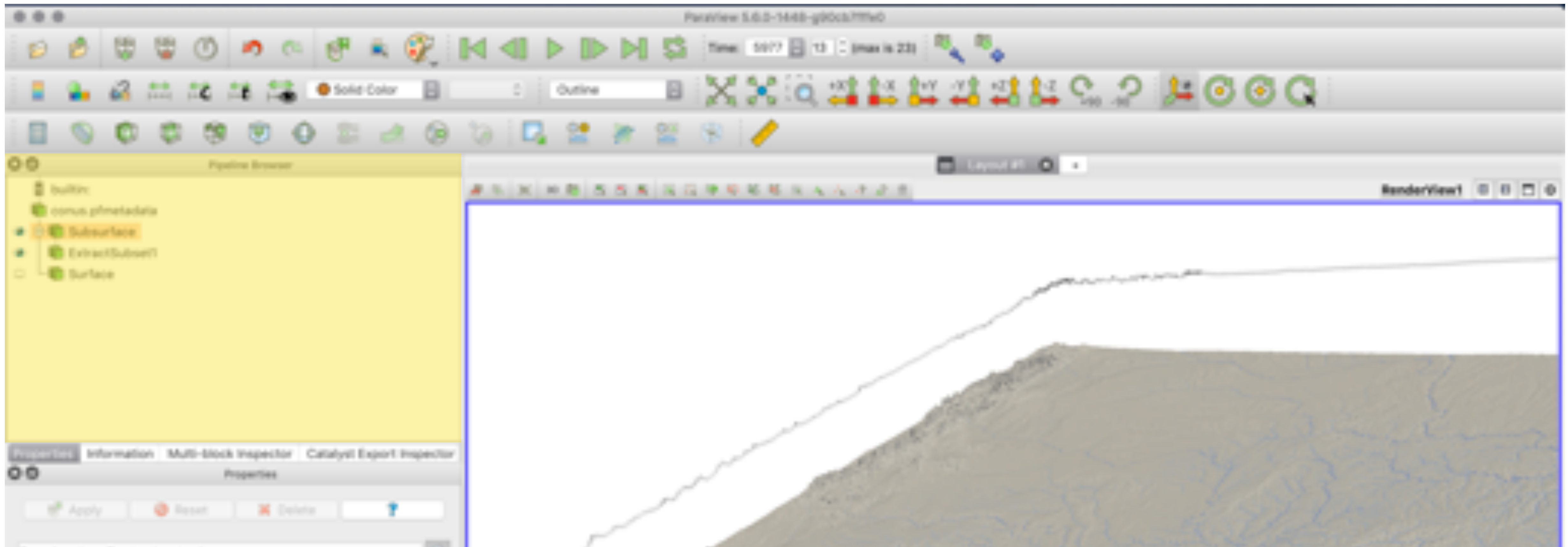
- ParaView shows you sources, filters, sinks, and their connections in the Pipeline Browser.





# ParaView

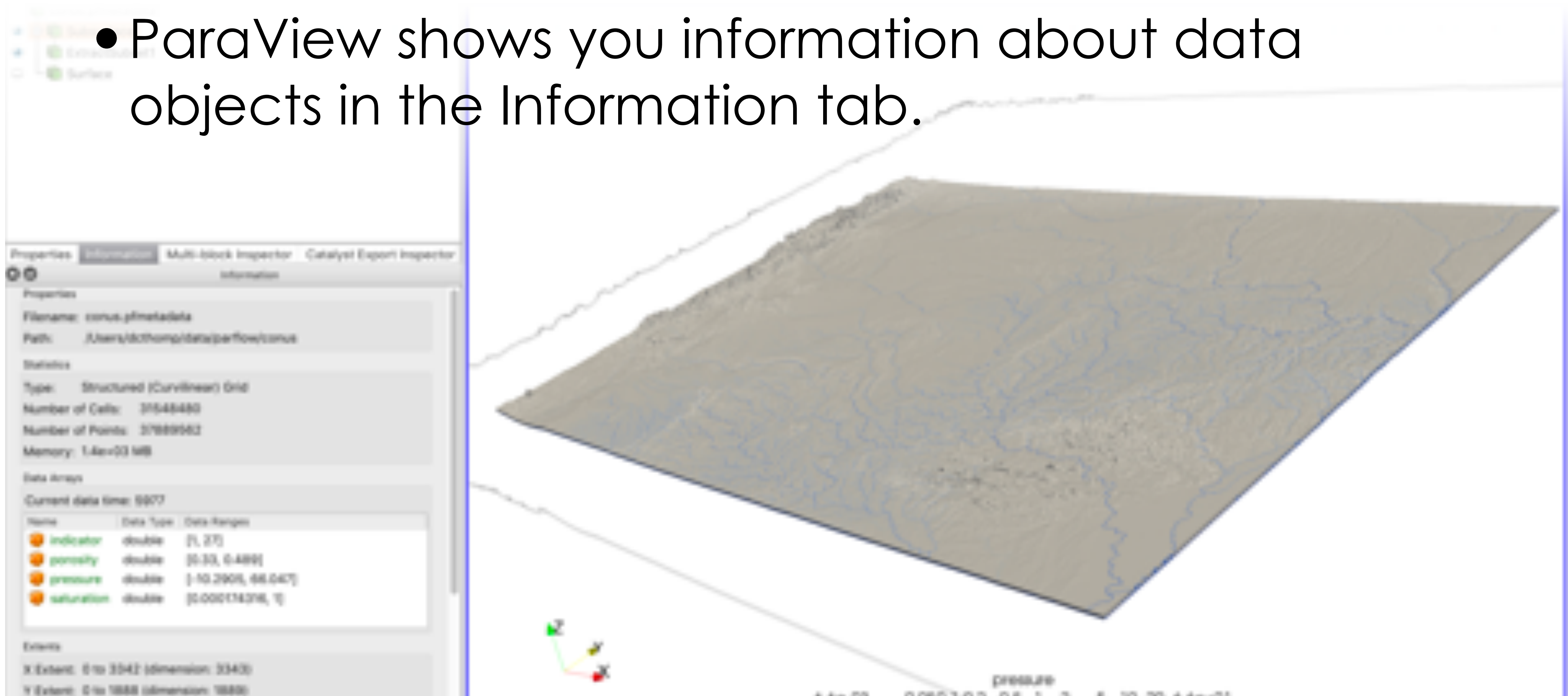
- ParaView shows you sources, filters, sinks, and their connections in the Pipeline Browser.





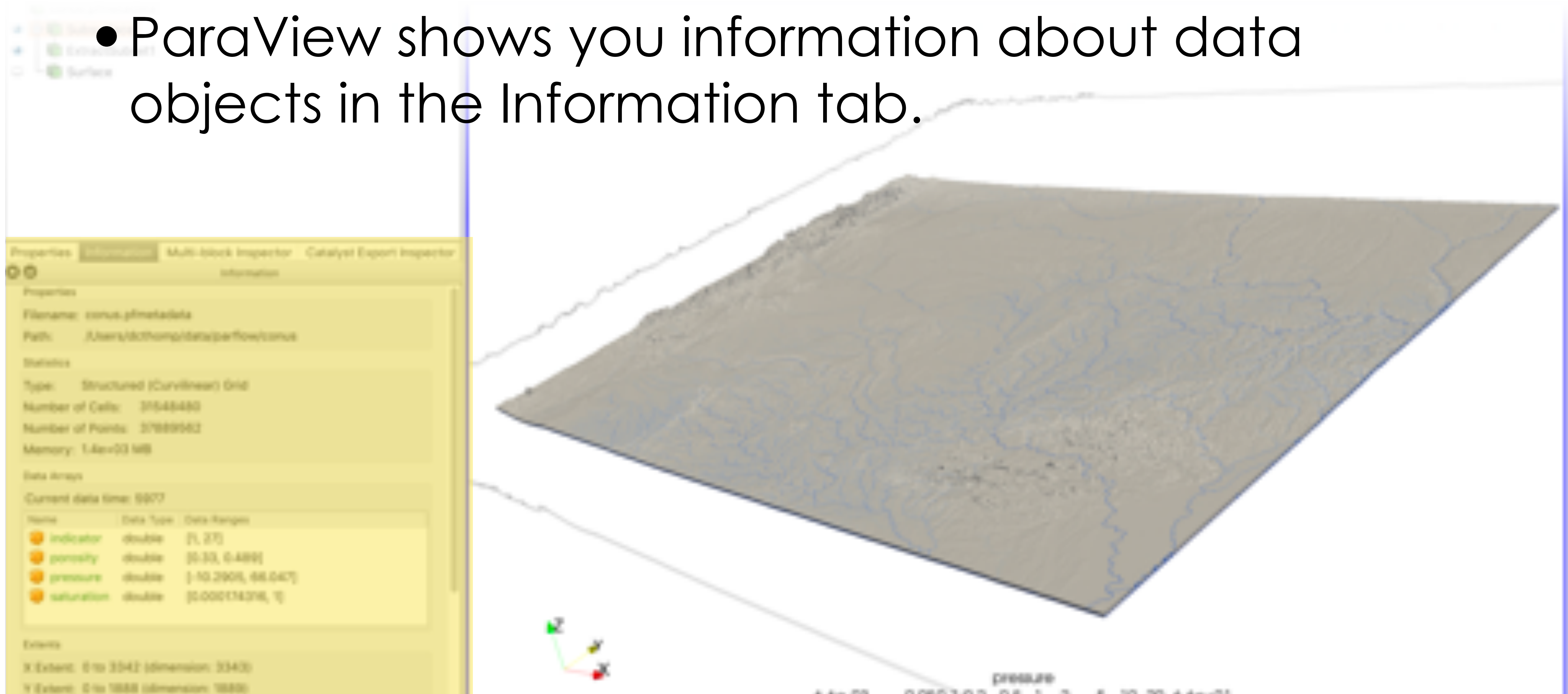
# ParaView

- ParaView shows you information about data objects in the Information tab.



# ParaView

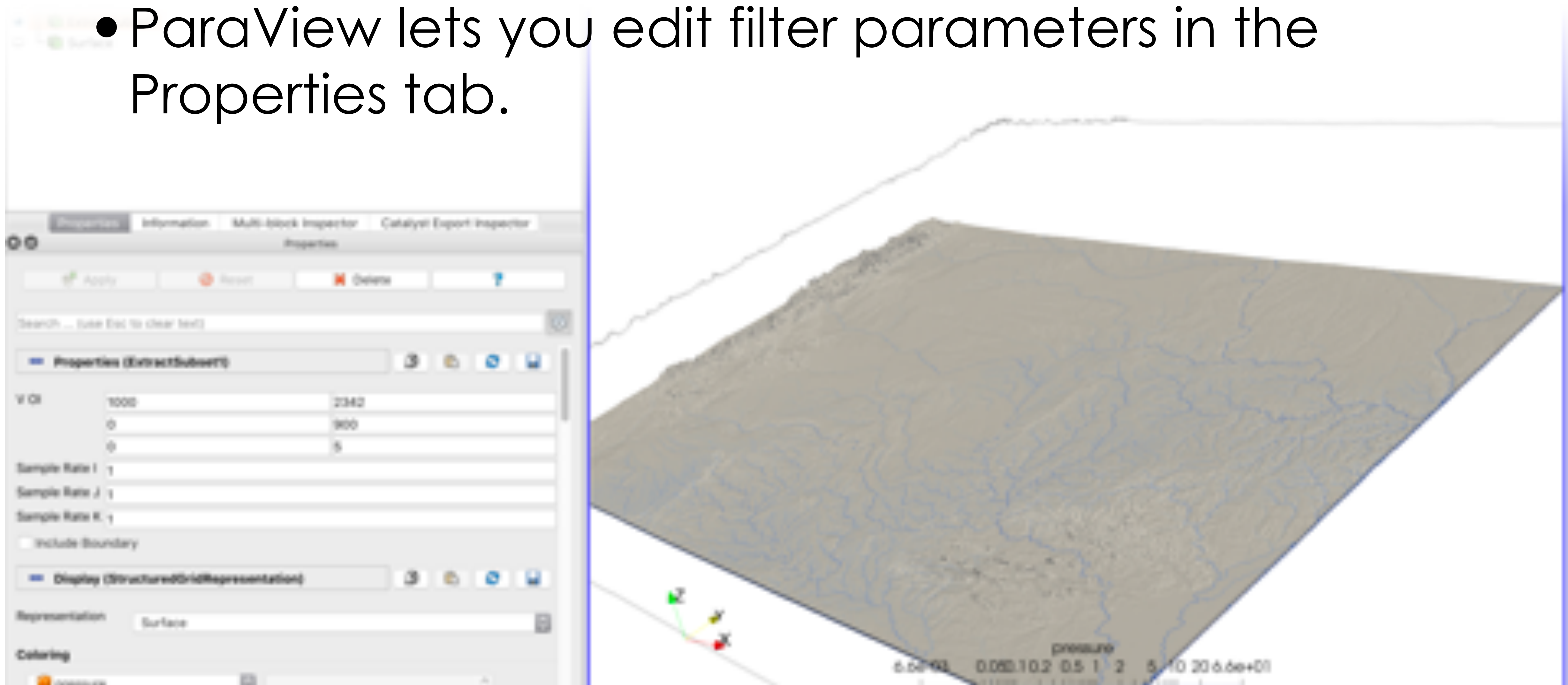
- ParaView shows you information about data objects in the Information tab.





# ParaView

- ParaView lets you edit filter parameters in the Properties tab.



# ParaView

- ParaView lets you edit filter parameters in the Properties tab.



# Overview

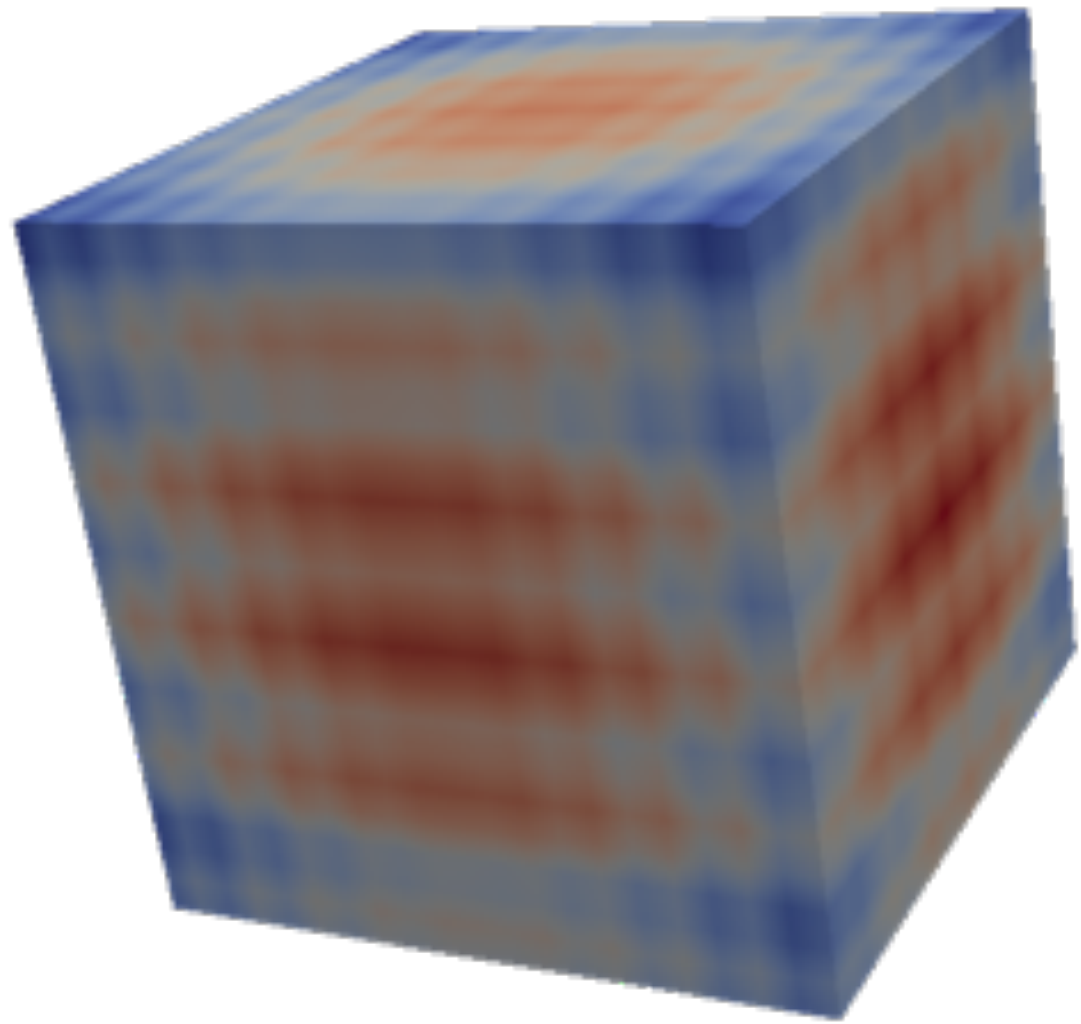
- Data types & filter pipelines in VTK & ParaView 20 min
- Visualization is composition 20 min
- Tools for ParFlow 45 min
- Going further with Python and NumPy 45 min
- Catalyst and web demonstrations 20 min



# Visualization is Composition

- There are relatively few fundamental visualization operations: color-by-value, cut, clip, threshold, isocontour, streamline/streakline, glyphs, volume rendering, deformation, legends/keys, parallel coordinates, focus+context, small multiples, ...
- Most compelling visualizations are compositions of these that convey information by using human perception while avoiding clutter.

# Color by value



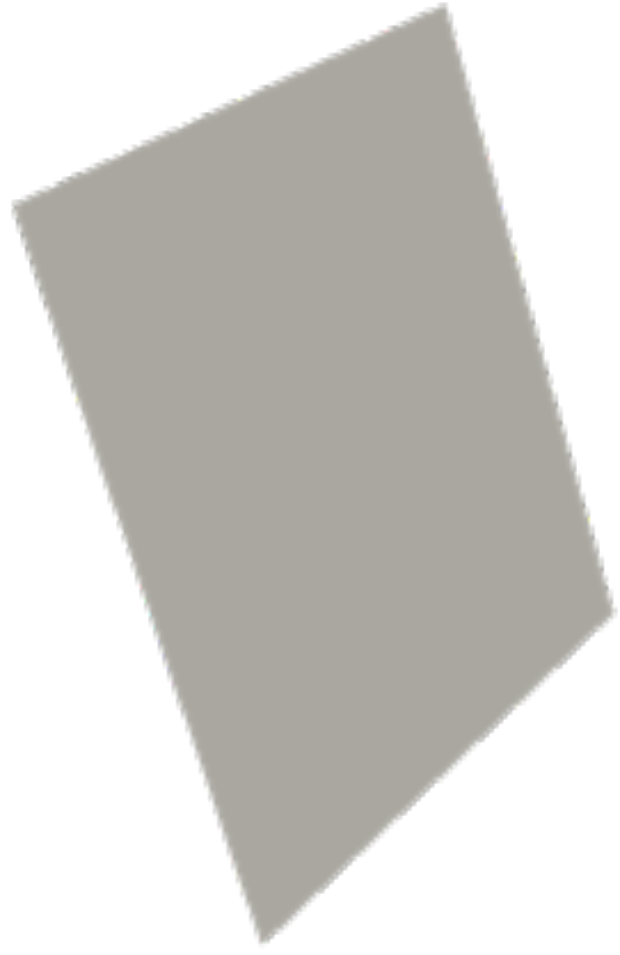
- Color each point by simulated value.
- Interpolate between points (or not, depending on simulation).
- Only shows surface values.
- Color scale and palette induce psychological bias.

# Isocontour



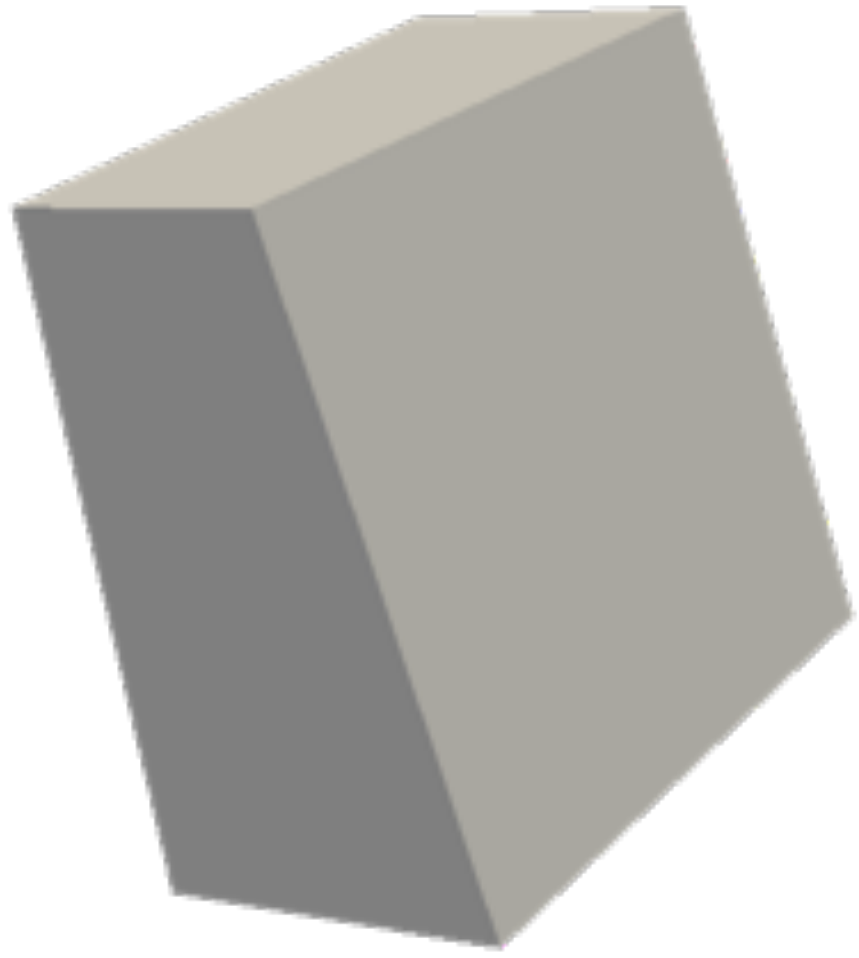
- Find locus of points with a given value at time  $t$ .
- Shows interesting shapes, but ...
- ... shapes may not be stable and may miss features.
- Multiple isocontours behave like onion layers.

# Cutting Surfaces



- Slice domain of simulation with a plane or other surface.
- Useful when domain changes shape over time.
- Only shows a small region of domain; features may be missed.

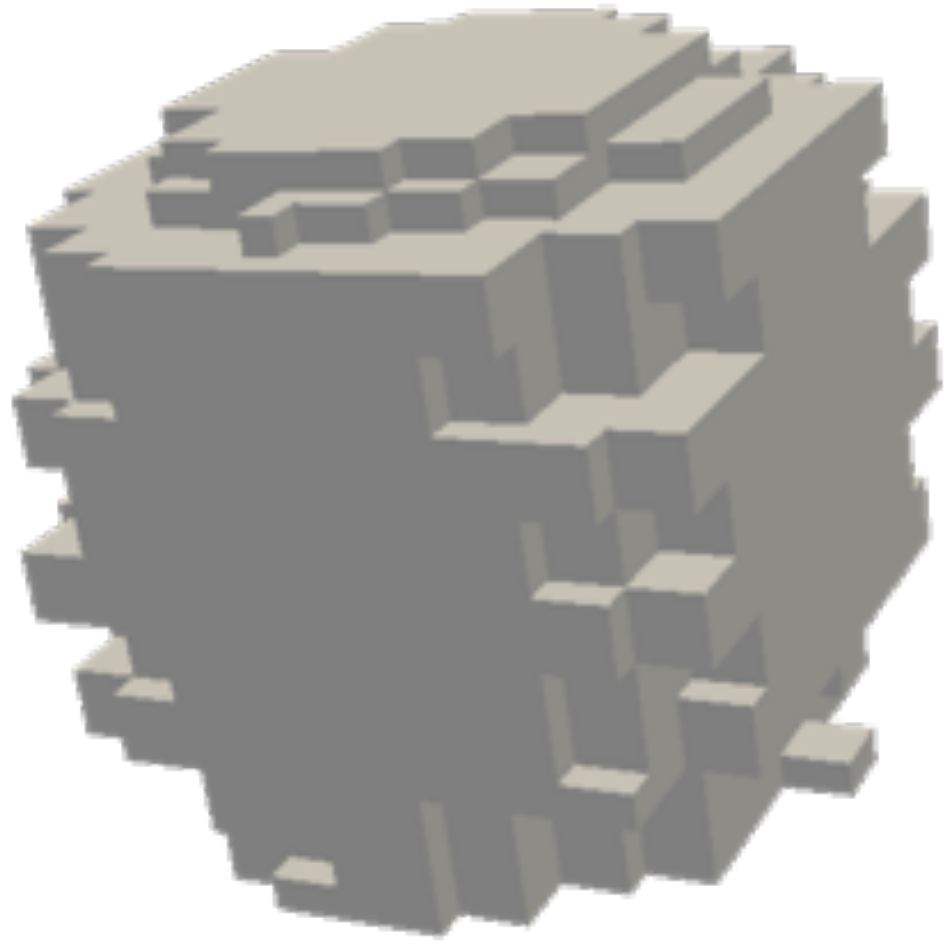
# Clipping



- Slice domain of simulation with a halfspace.
- Like a cutting plane, but provides more context.
- Clipped dataset often used as input for other visualizations.



# Threshold



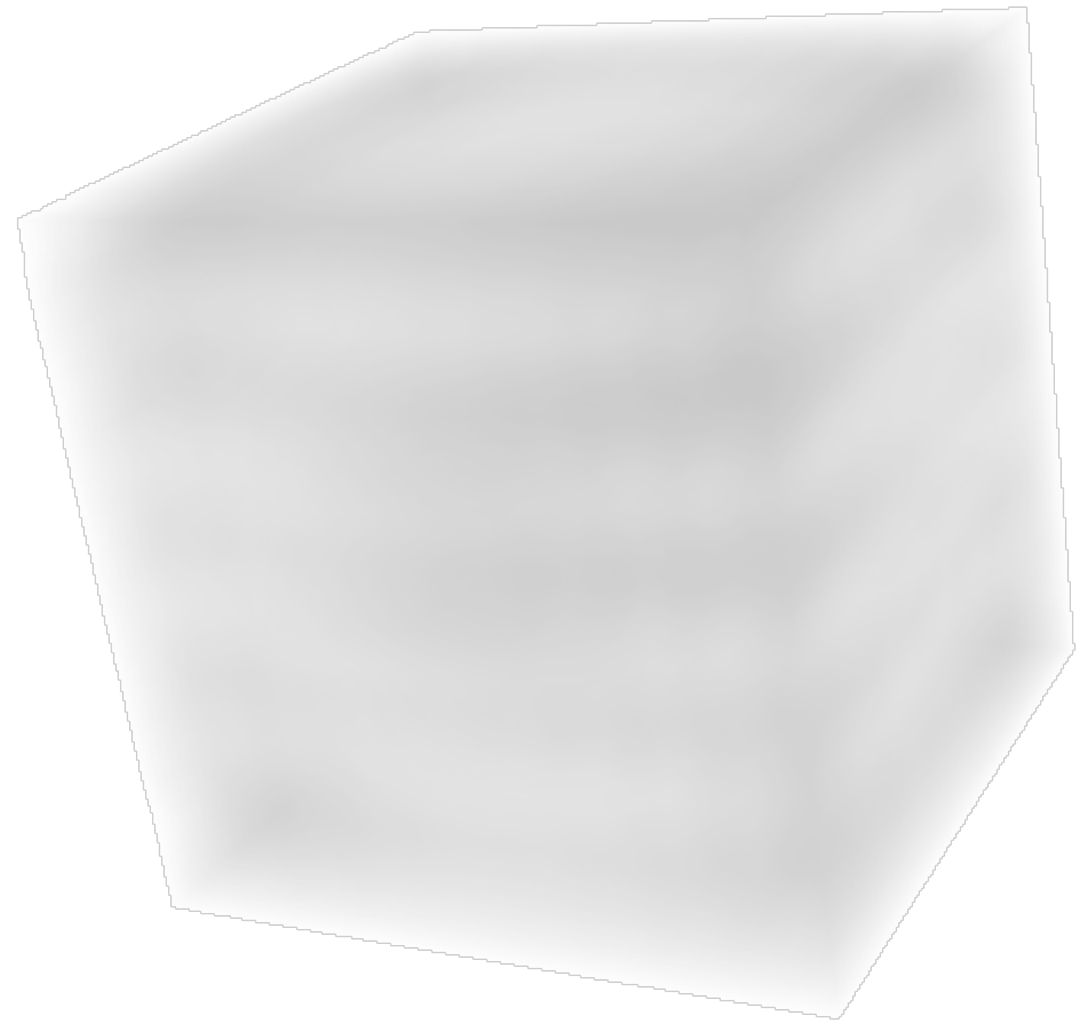
- Subset of grid cells whose values are in a given range.
- Like clipping plane, but preserves shape of grid cells.
- Good for showing effect of discretization of domain.

# Streamlines/Streaklines



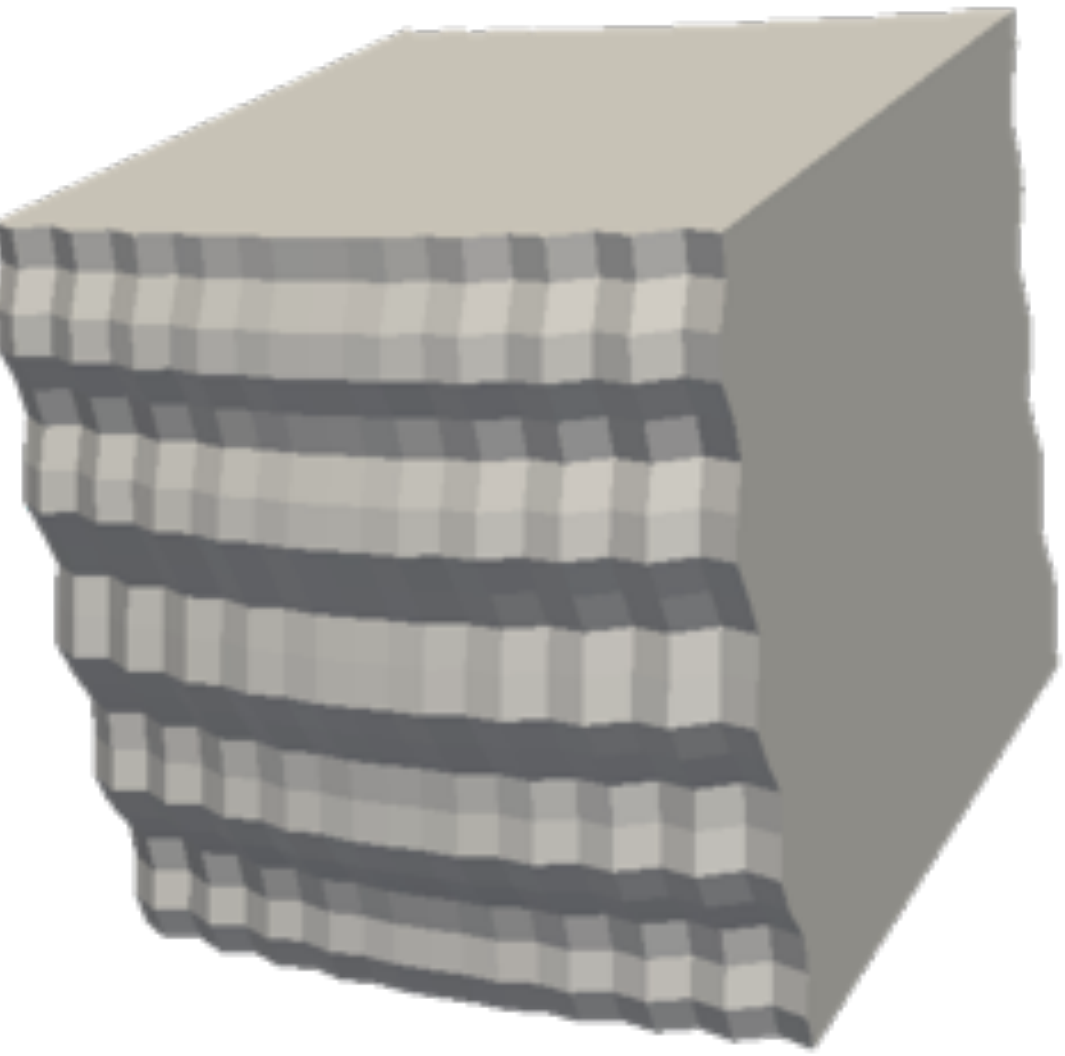
- Slice domain of simulation with a halfspace.
- Like a clipping plane, but provides more context.
- Clipped dataset often used as input for other visualizations.

# Volume Rendering



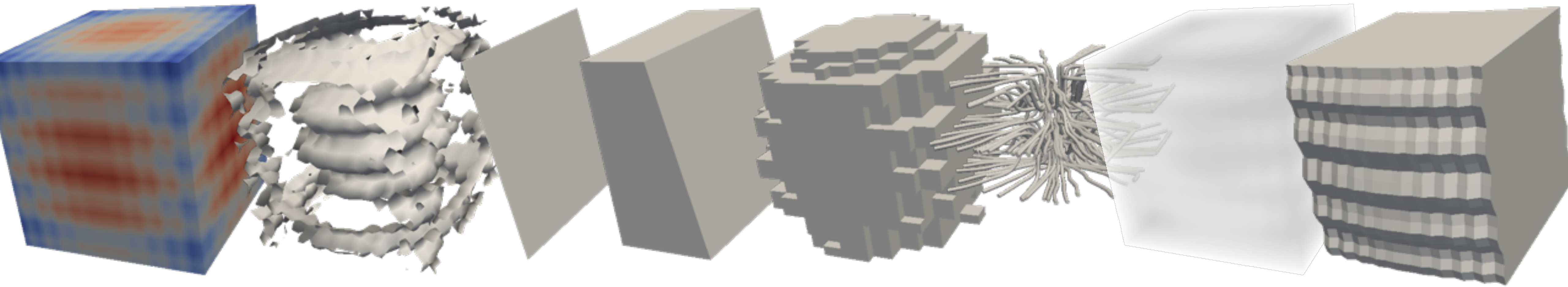
- Values are semitransparent according to a transfer function.
- See features inside volume.
- Often fuzzy, so it can be hard to discern feature shape.
- Occlusion can still be a problem.

# Deformation



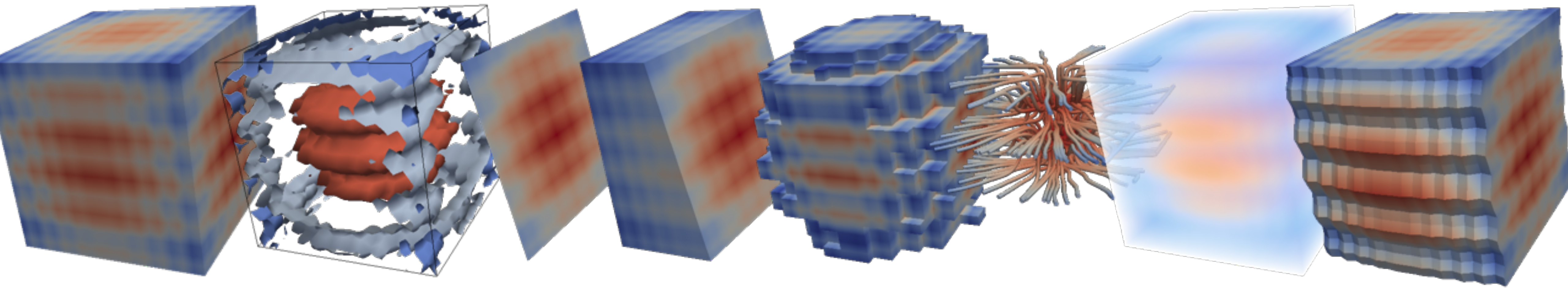
- Displace each point by a vector to change shape of grid.
- Often used in solid mechanics.
- Displacement can be exaggerated to make features clear.
- Occlusion can be a problem.

# Visualizations Are Compositions!





# Visualizations Are Compositions!



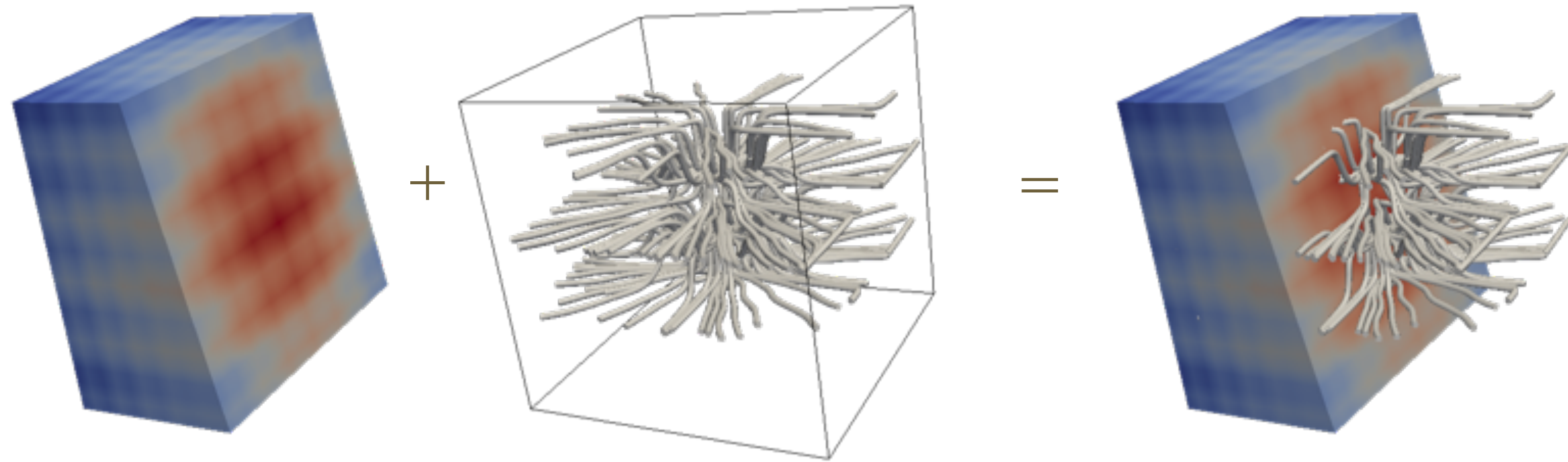
# Visualization

- Traditionally, visualization performed after a simulation (hence *post*-processing); until recently, adjusting the composition has been simple.
- Heterogenous nature of computers and communication provides an opportunity for visualization/analysis with higher fidelity, but at the expense of some flexibility in composition.

# Visualization Strategies

- Perform analysis during simulation to determine when to save results and which results to save.
- Generate visualizations using a script prepared with results of smaller runs where traditional post-processing tools can be used.
- Generate visualization artifacts which allow post-simulation composition.

# Visualization Strategies



# Visualization is Composition

- For further study:
  - Edward Tufte, *The Visual Display of Quantitative Information*. Graphics Press, 2001.
  - Colin Ware, *Information Visualization*. Morgan Kaufmann, 2012.



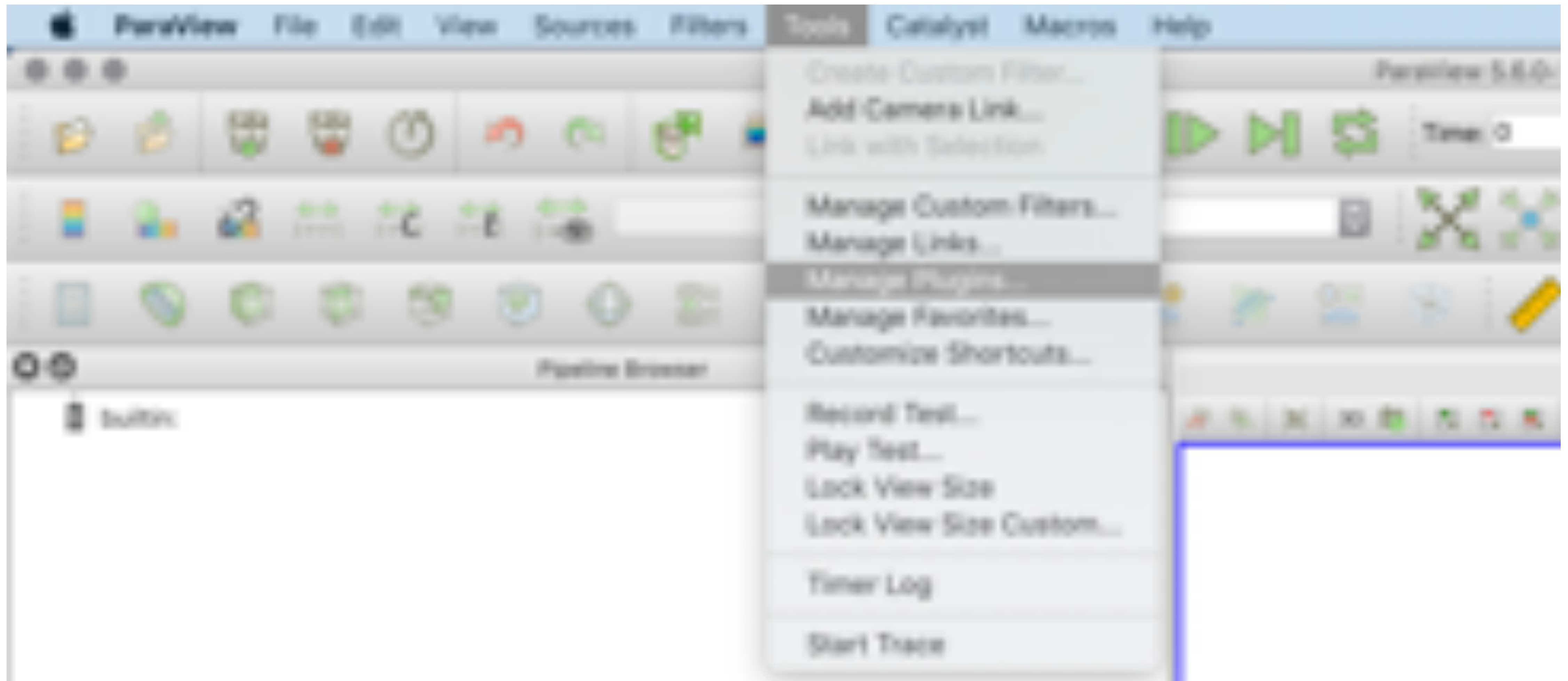
# Overview

- Data types & filter pipelines in VTK & ParaView 20 min
- Visualization is composition 20 min
- Tools for ParFlow 45 min
- Going further with Python and NumPy 45 min
- Catalyst and web demonstrations 20 min

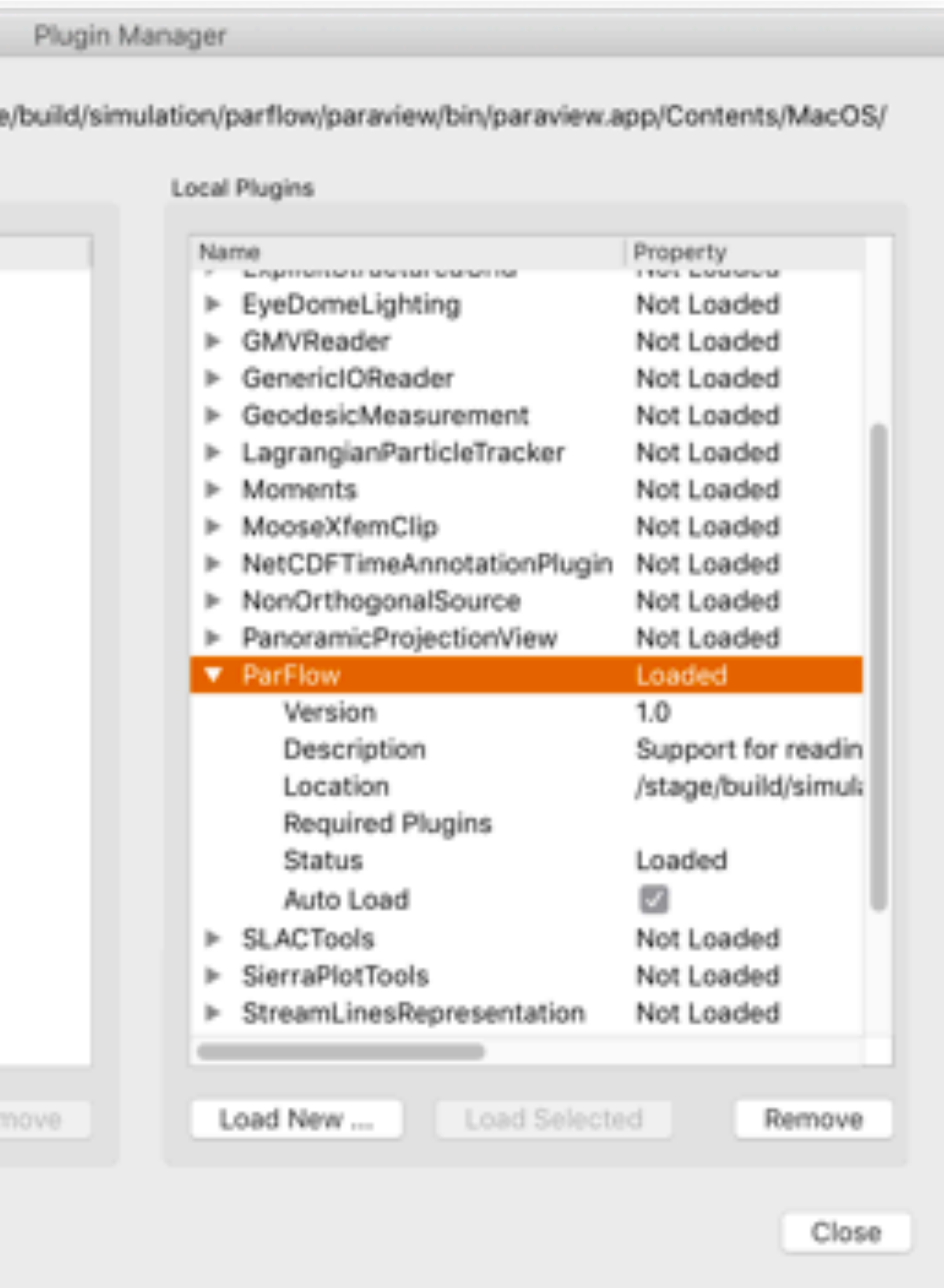
# Tools for ParFlow

- Loading plugins
- Reading and animating data
- Plotting selections over time
- Colormaps for stream flow
- Simple calculations

# Tools for ParFlow



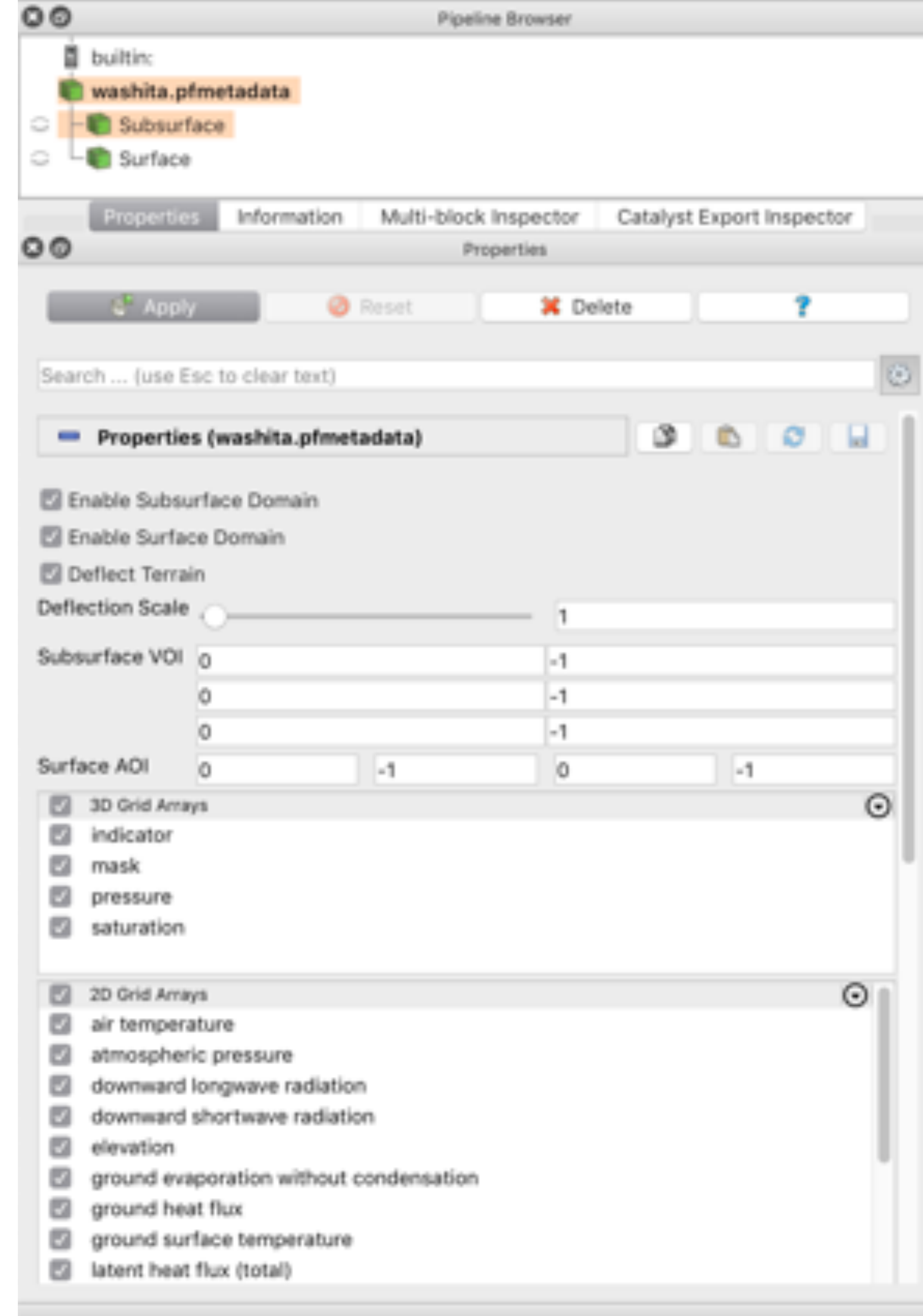
# Tools for ParFlow



- Expand "ParFlow" plugin
- Click "Auto Load"
- Quit ParaView and restart (this saves the autoload preference)
- Now .pfb, .C.pfb, .pfmetadata are recognized file formats.

# Tools for ParFlow

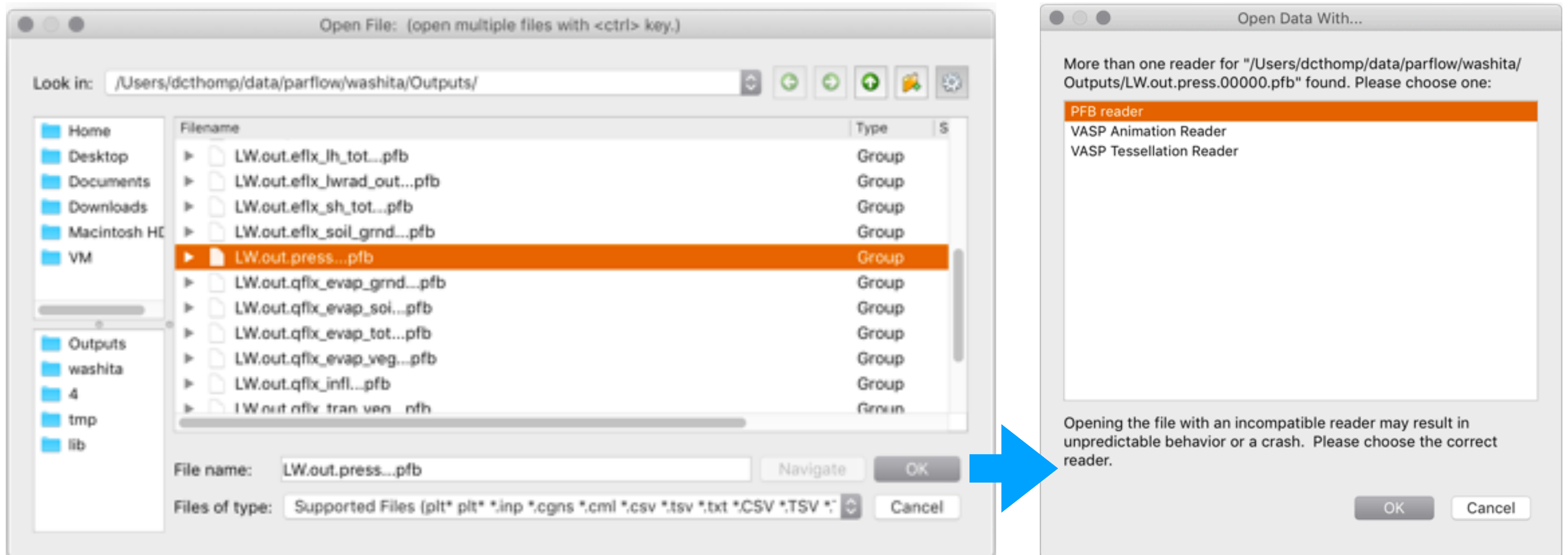
- Open a .pfmetadata file.
- Nothing is actually loaded until you click *Apply* since the data may be large; this can load many PFB files at once.
- VOI/AOI use "-1" to indicate the full extent should be used.





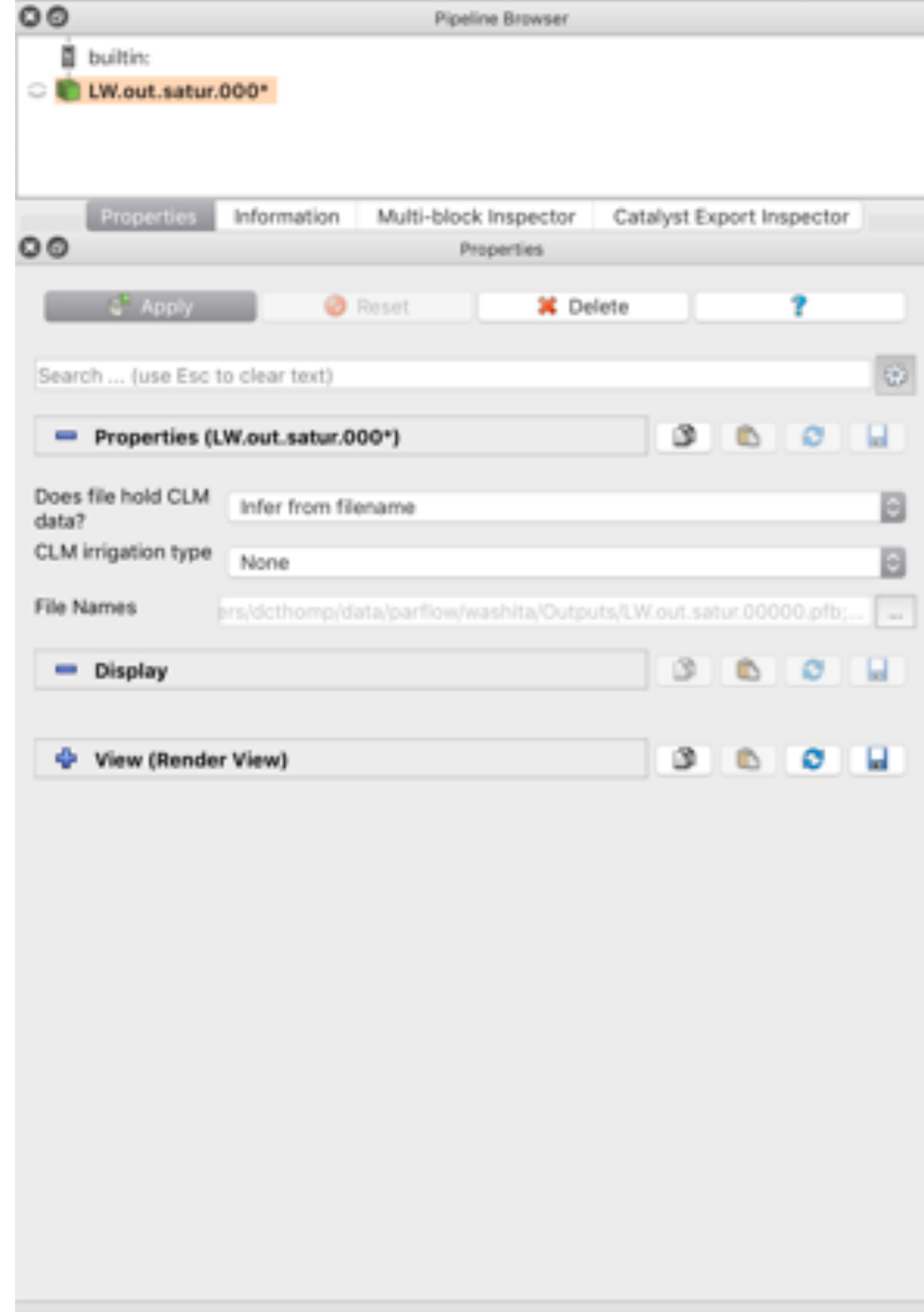
# Tools for ParFlow

When opening .pfb, .C.pfb files, you need to tell ParaView the files are ParFlow data, not VASP data.



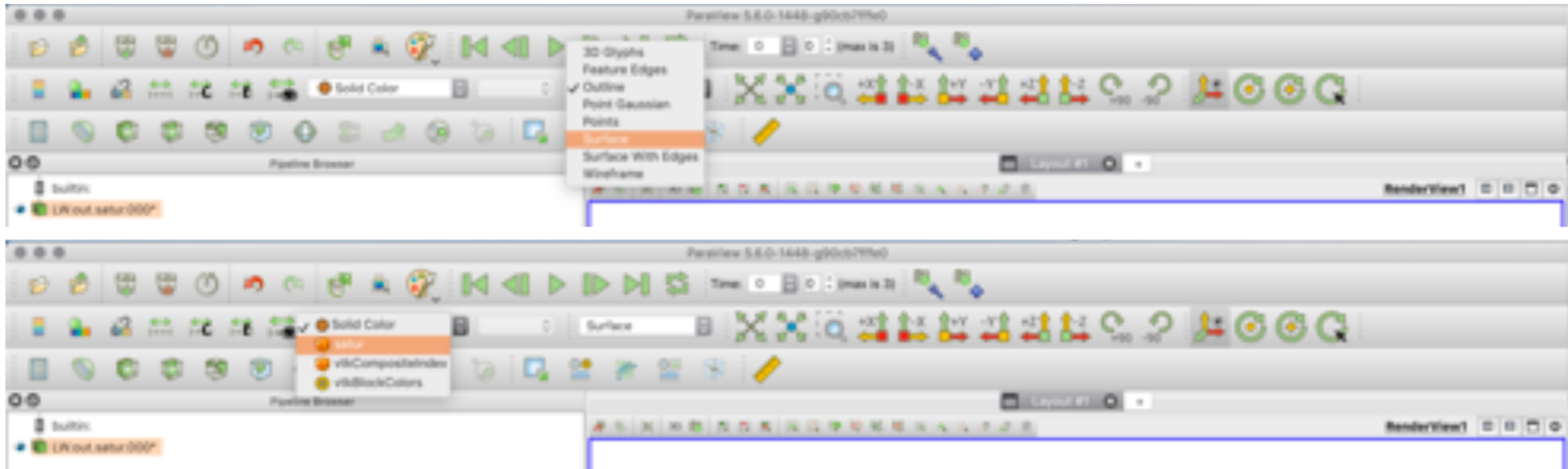
# Tools for ParFlow

- Open a .pfb file.
- Nothing is actually loaded until you click *Apply*.
- If loading CLM PFB files (.C.pfb), you need to specify the irrigate type since the file does not indicate it.

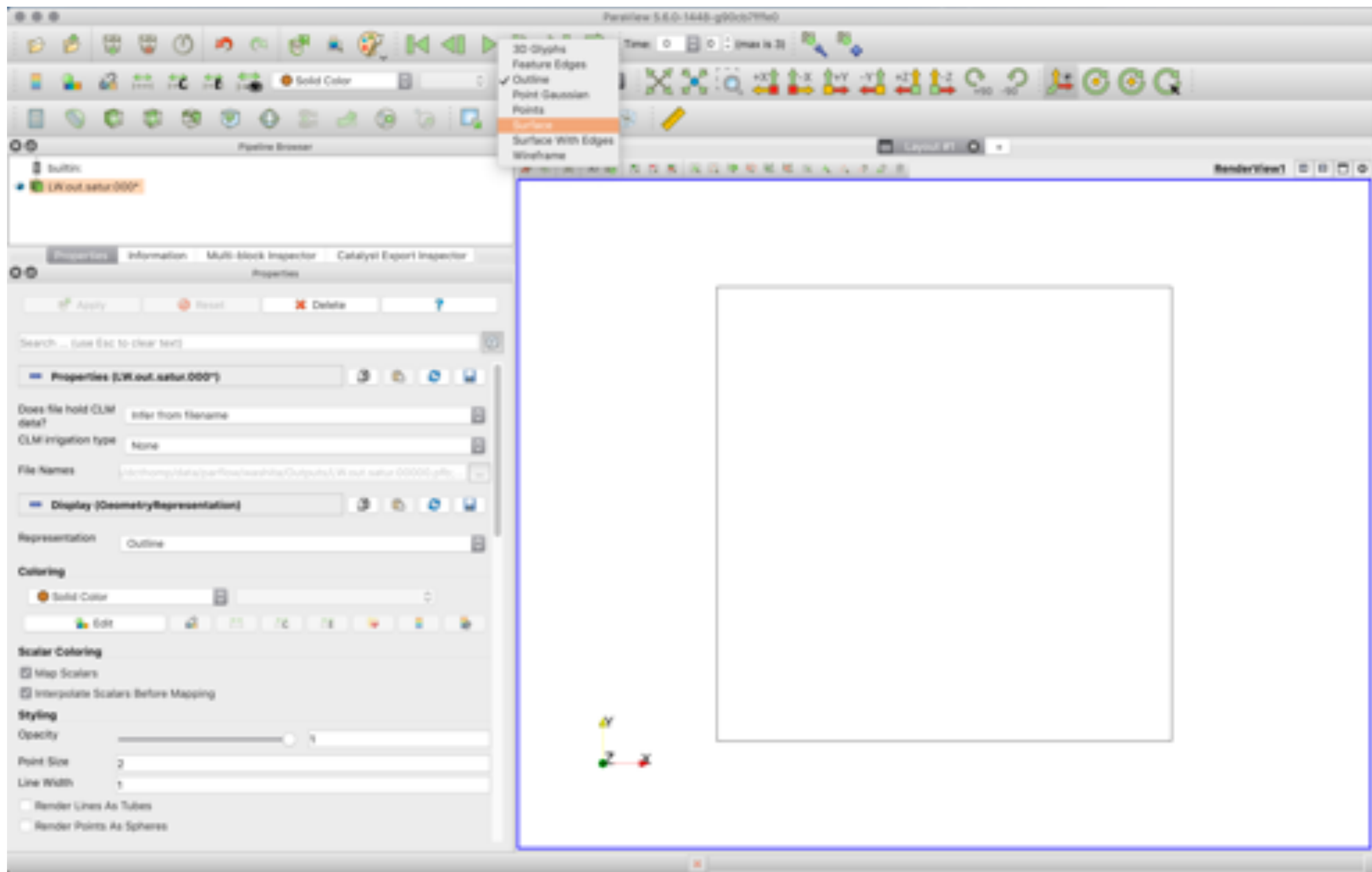


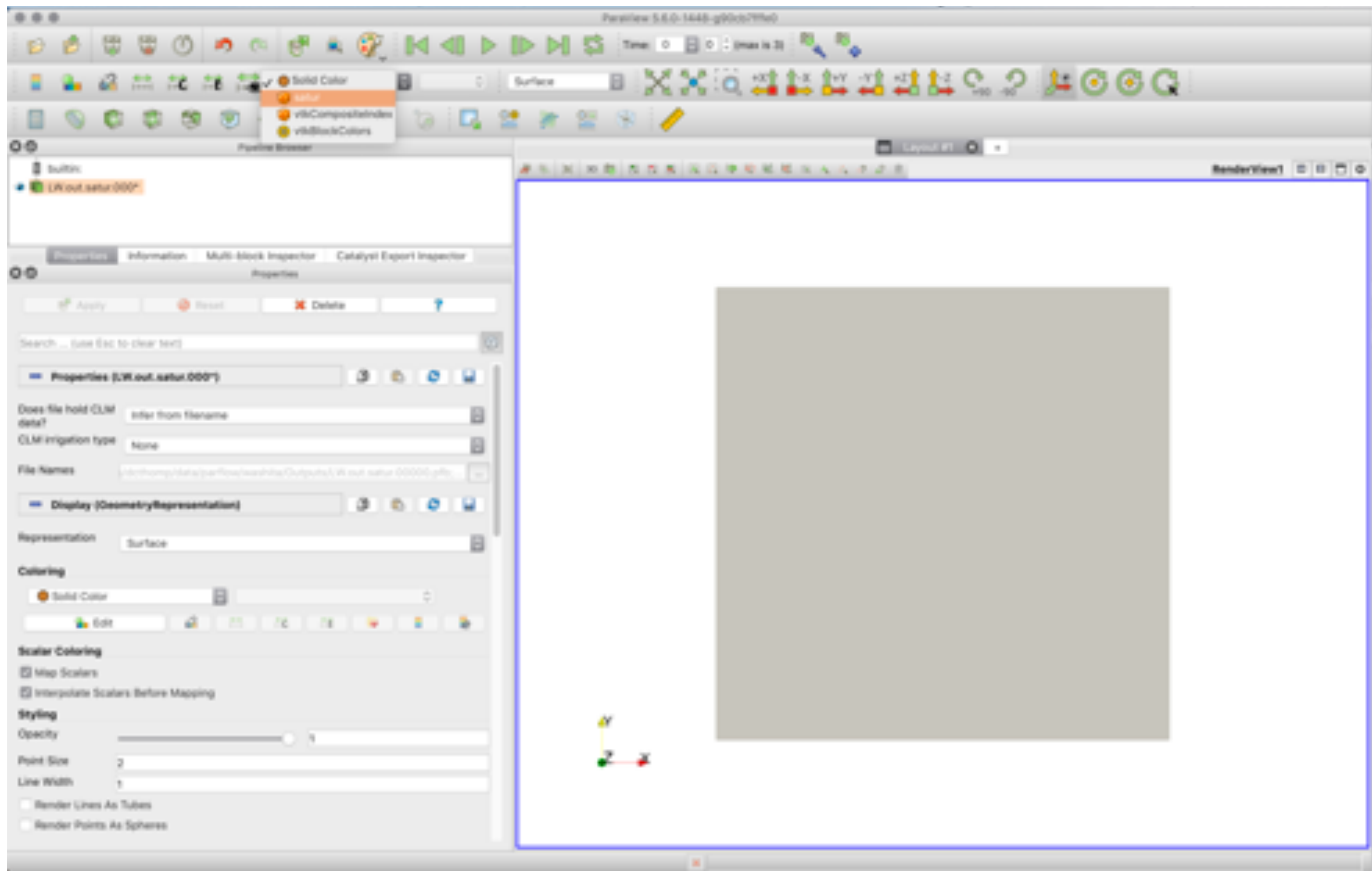
# ParaView Tip: Representations

- The toolbar controls the active filter's visual properties (how it is drawn and colored).

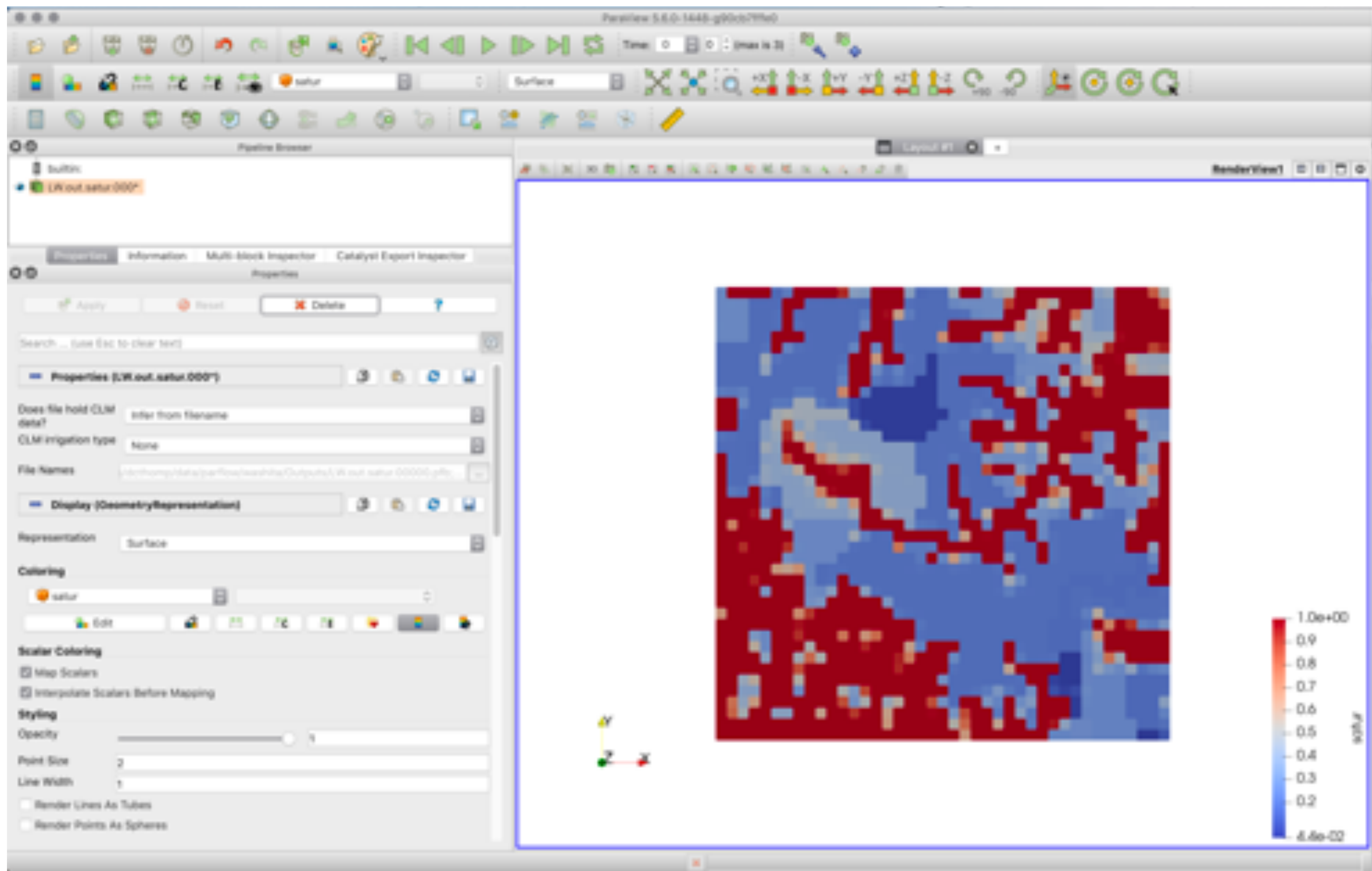












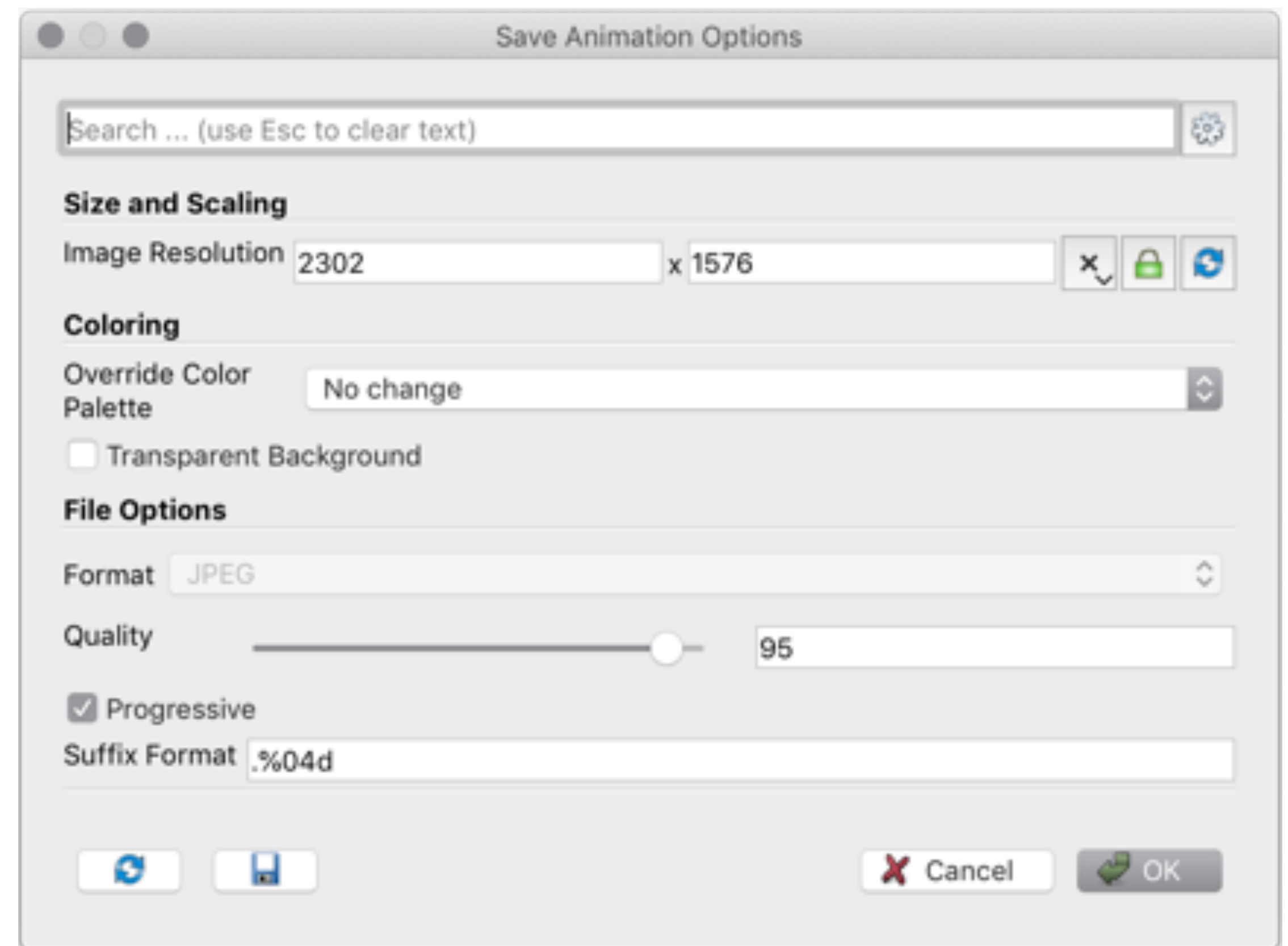
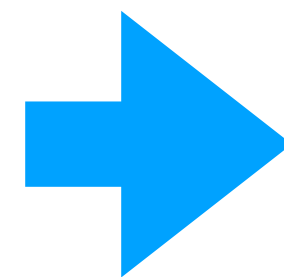
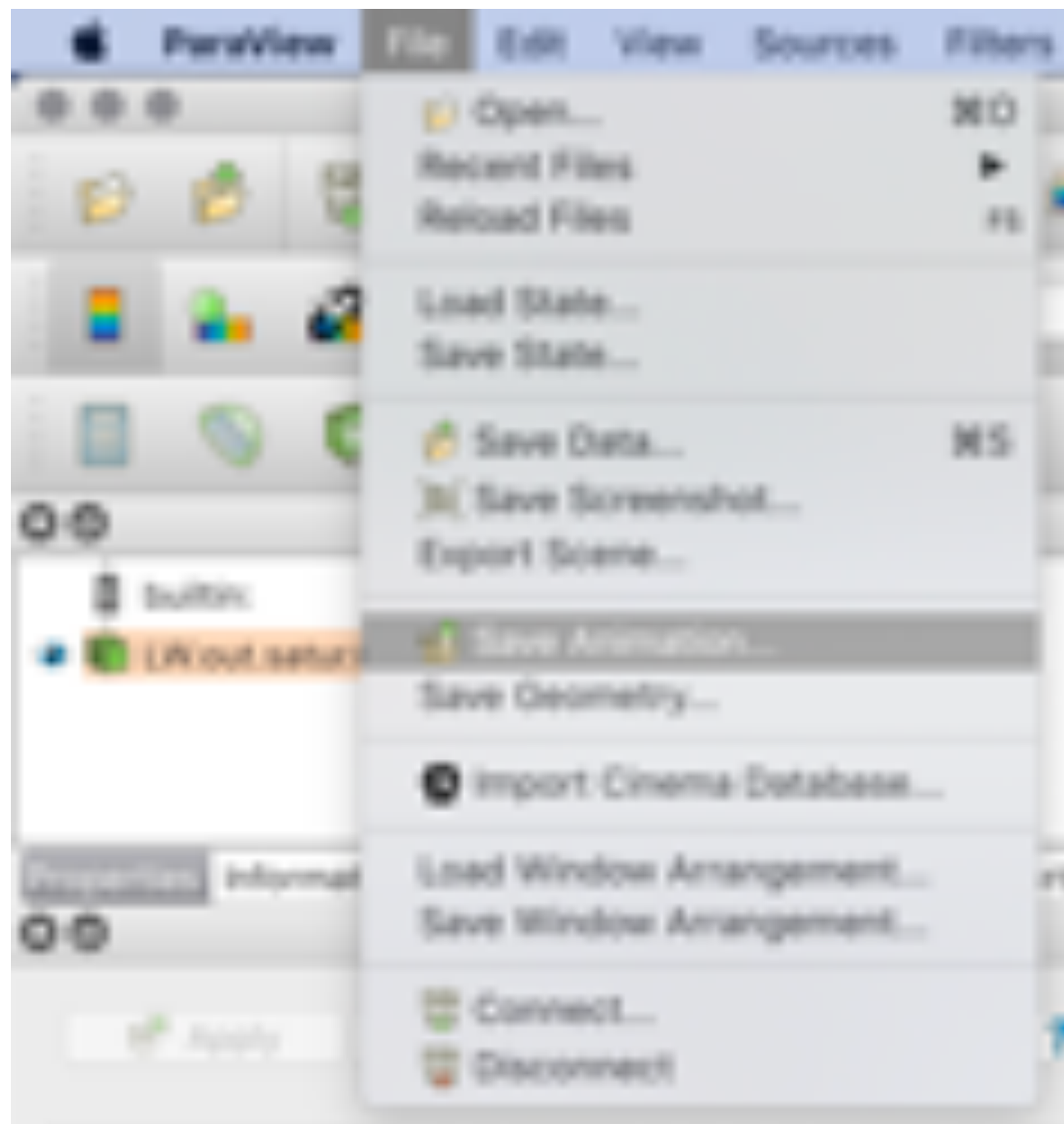
# Tools for ParFlow

- Both the .pfb and .pfmetadata readers can read timeseries data. Neither provide the actual time value yet, only an integer timestep.
- This toolbar controls the current timestep being displayed in ParaView.



# Tools for ParFlow

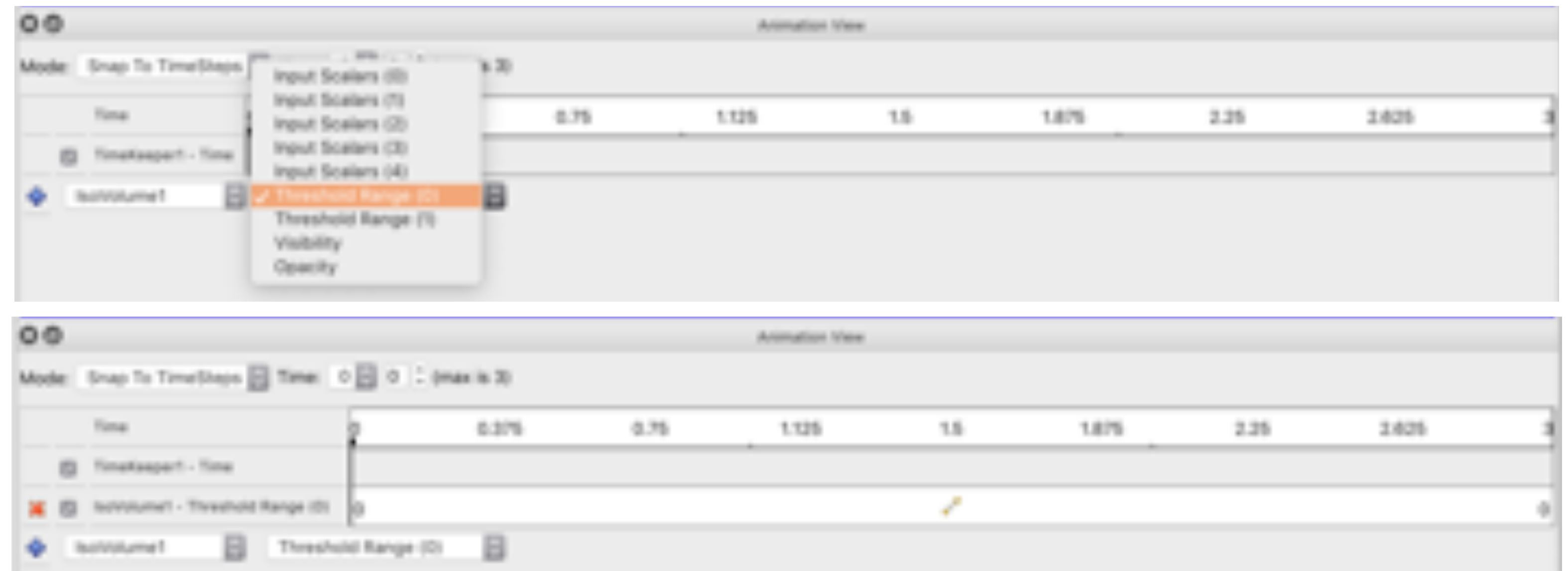
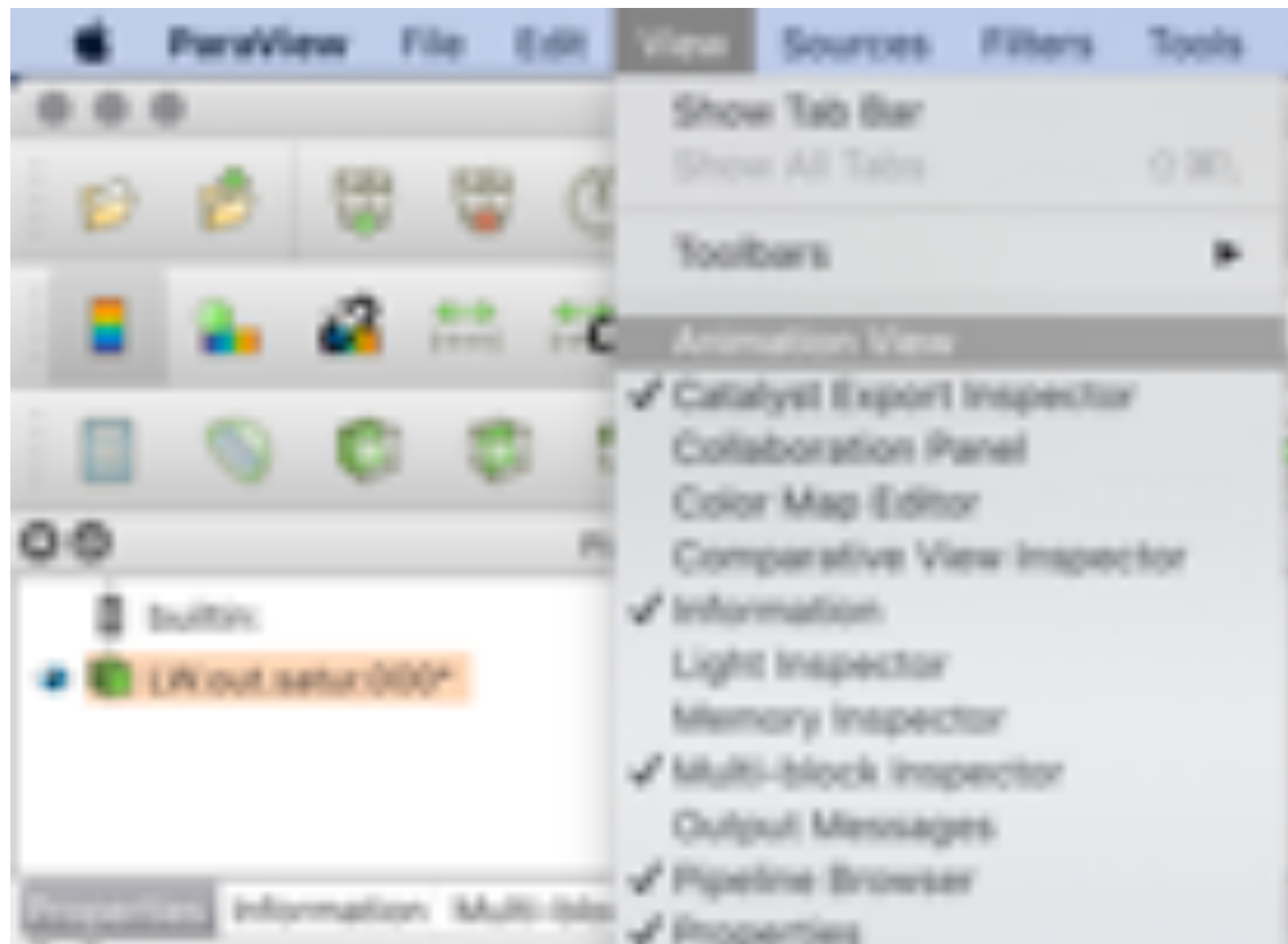
ParaView can also create image sequences (and movies directly on some platforms) for animation.





# Tools for ParFlow

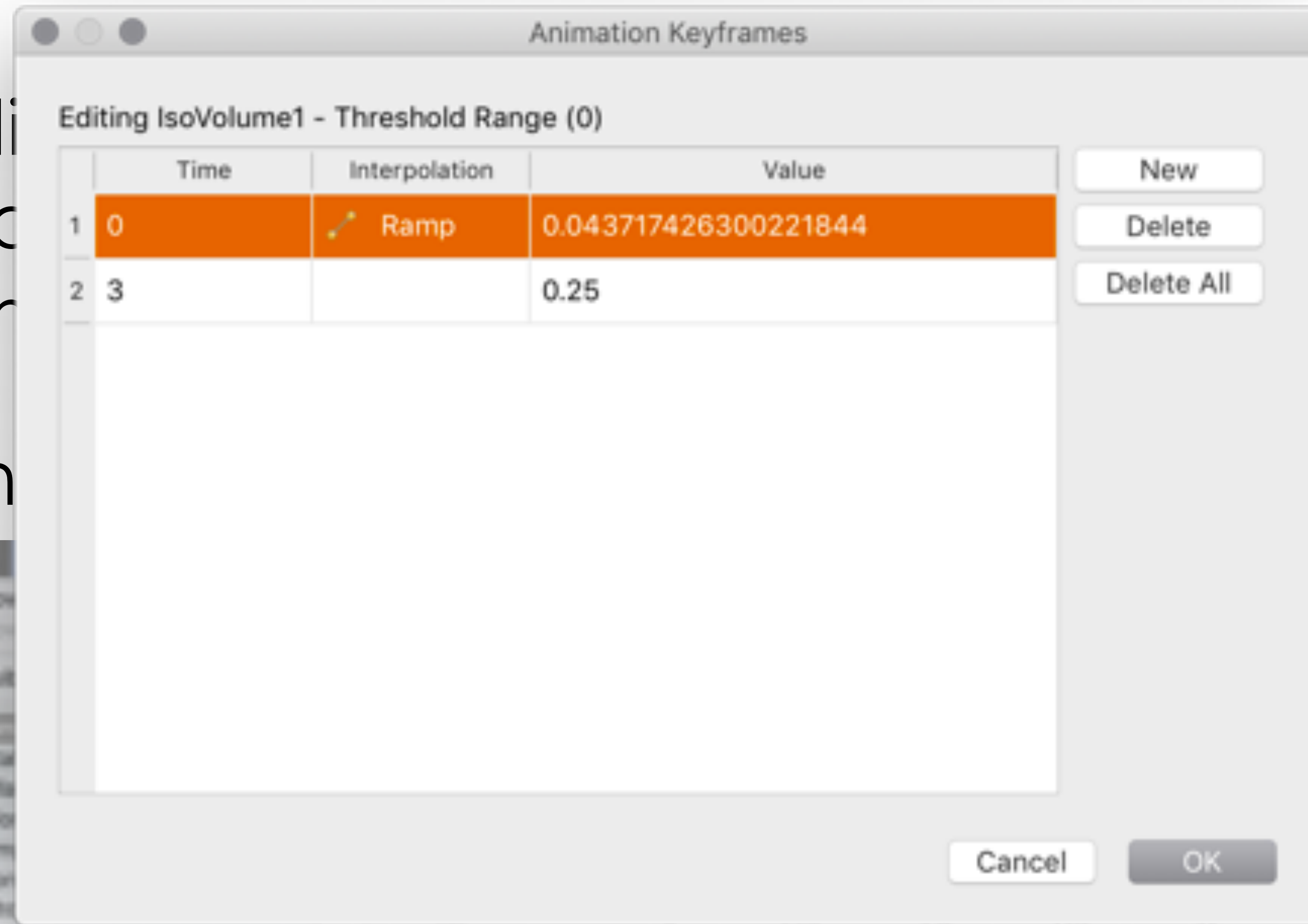
- In addition to playing timeseries data, ParaView can adjust (1) filter parameters like isovalue and (2) camera position and focus during animation.
- The Animation panel lets you choose what to vary.





# Tools for ParFlow

- In addition to the main view, you can add a second view (2) can be used to view the animation.
- The Animation Keyframes window allows you to edit the animation keyframes.

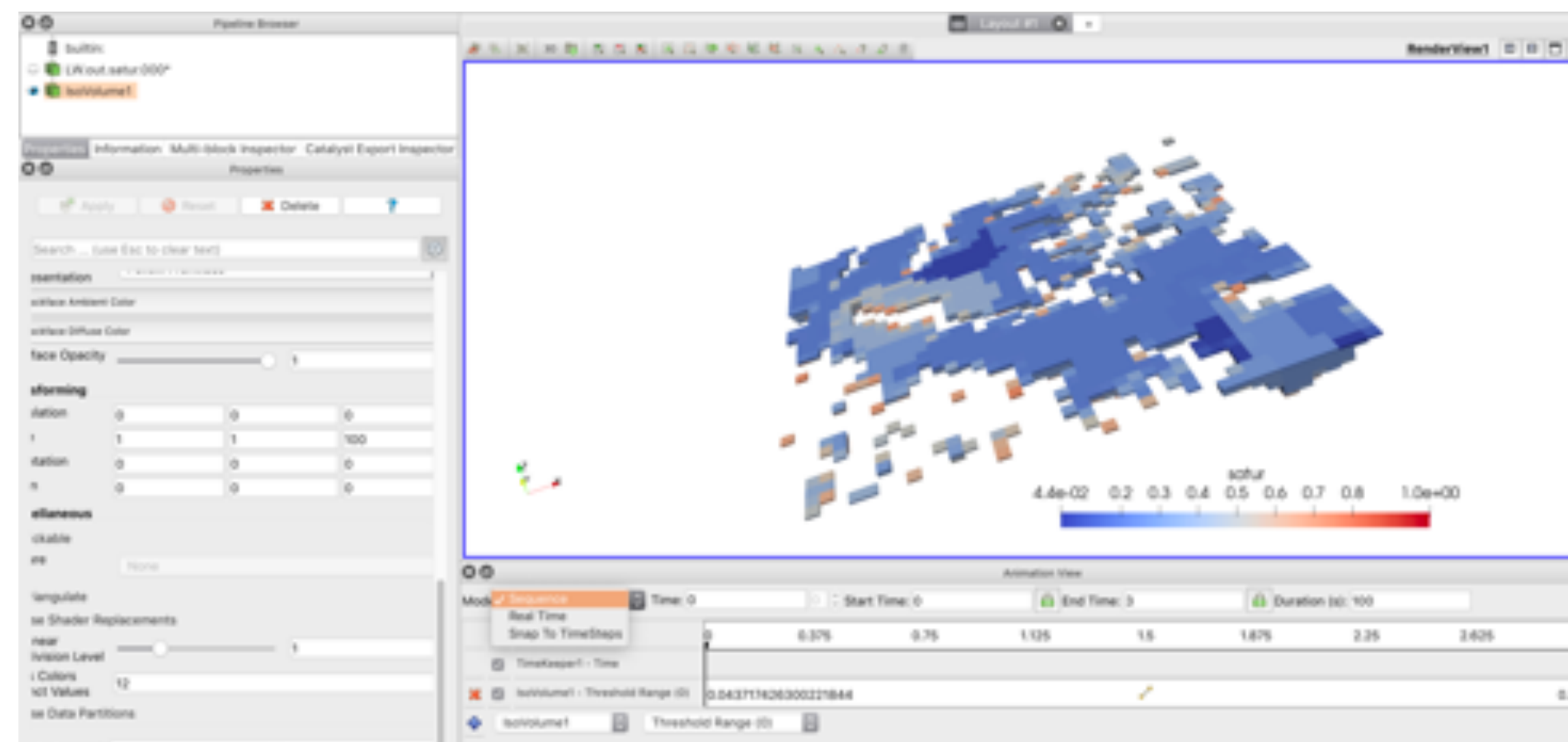


ParaView  
e and  
nation.  
t to vary.



# Tools for ParFlow

- By default, ParaView snaps animation timesteps to times available from sources (readers).
- If you want a movie that has more frames, you can change the animation mode in the panel.



Pipeline Browser

atur (000)

Multi-block Inspector Catalyst Export Inspector

Properties

Reset Delete

End to Clear Text

0 0 0

1 1 100

0 0 0

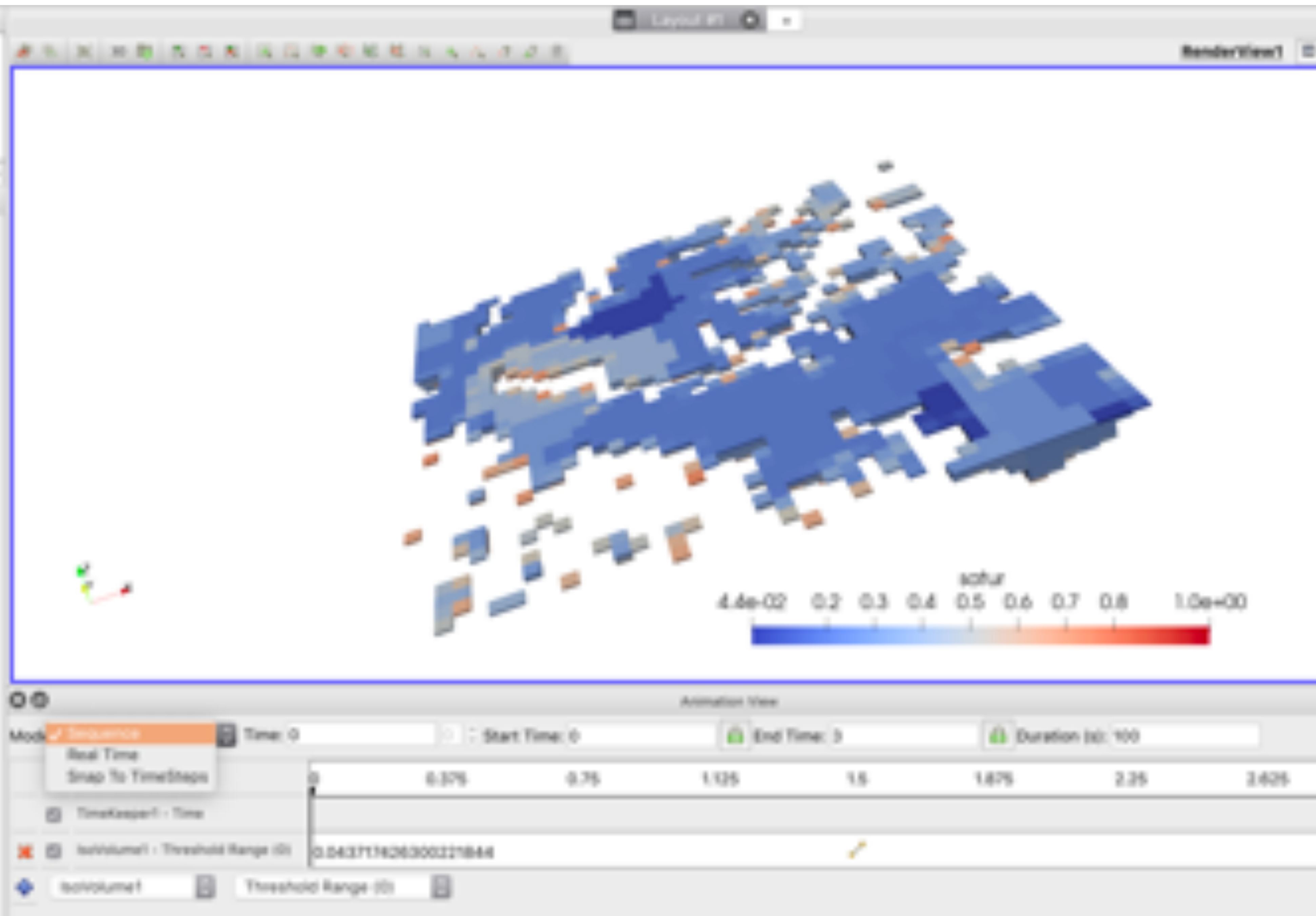
0 0 0

None

placements

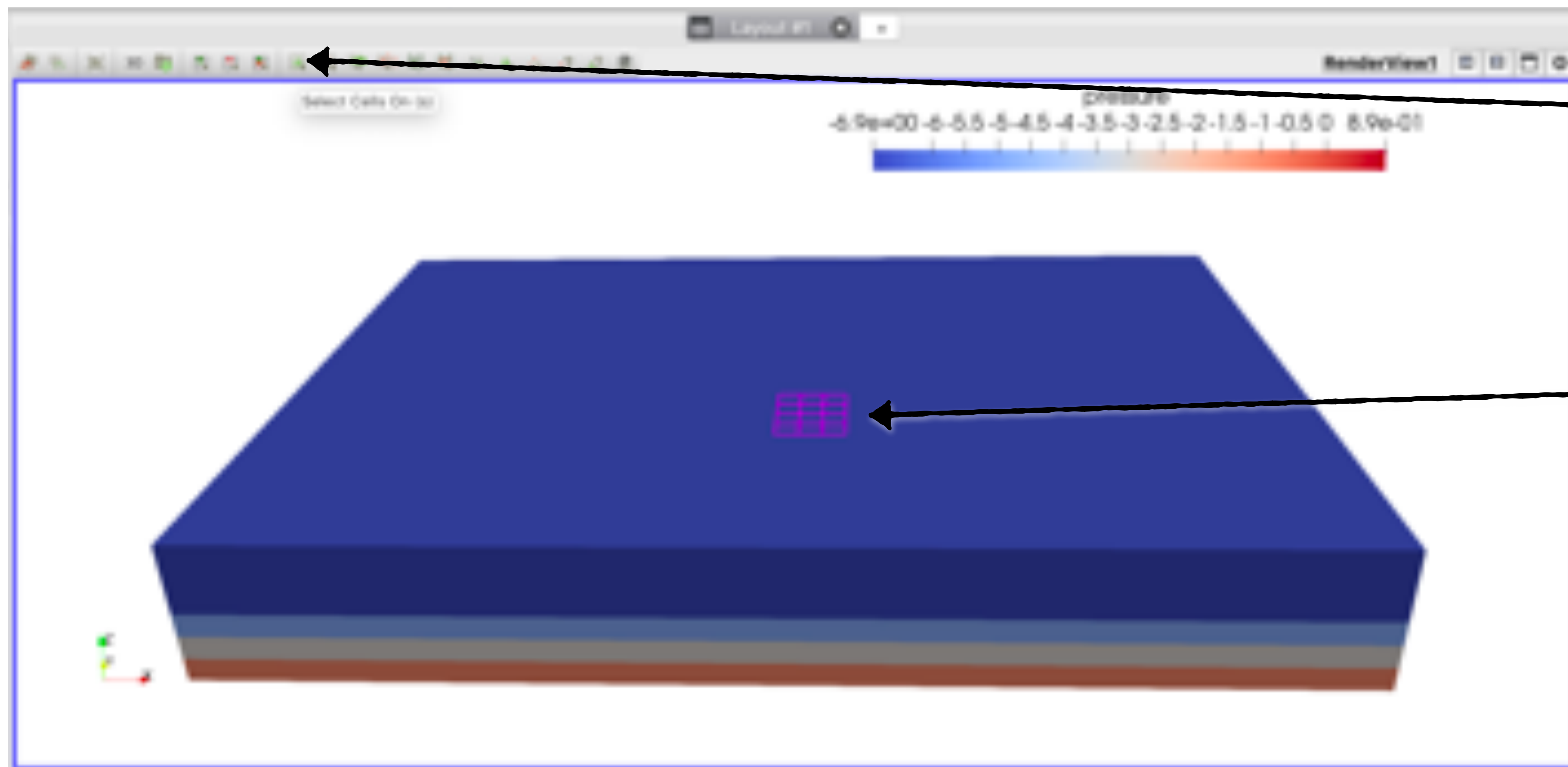
12

one



# Tools for ParFlow

- Now let's plot values of a selection over time.



Type 's' or click the Select Cells On button.

Click-drag on the dataset to choose cells. They will be highlighted in magenta.



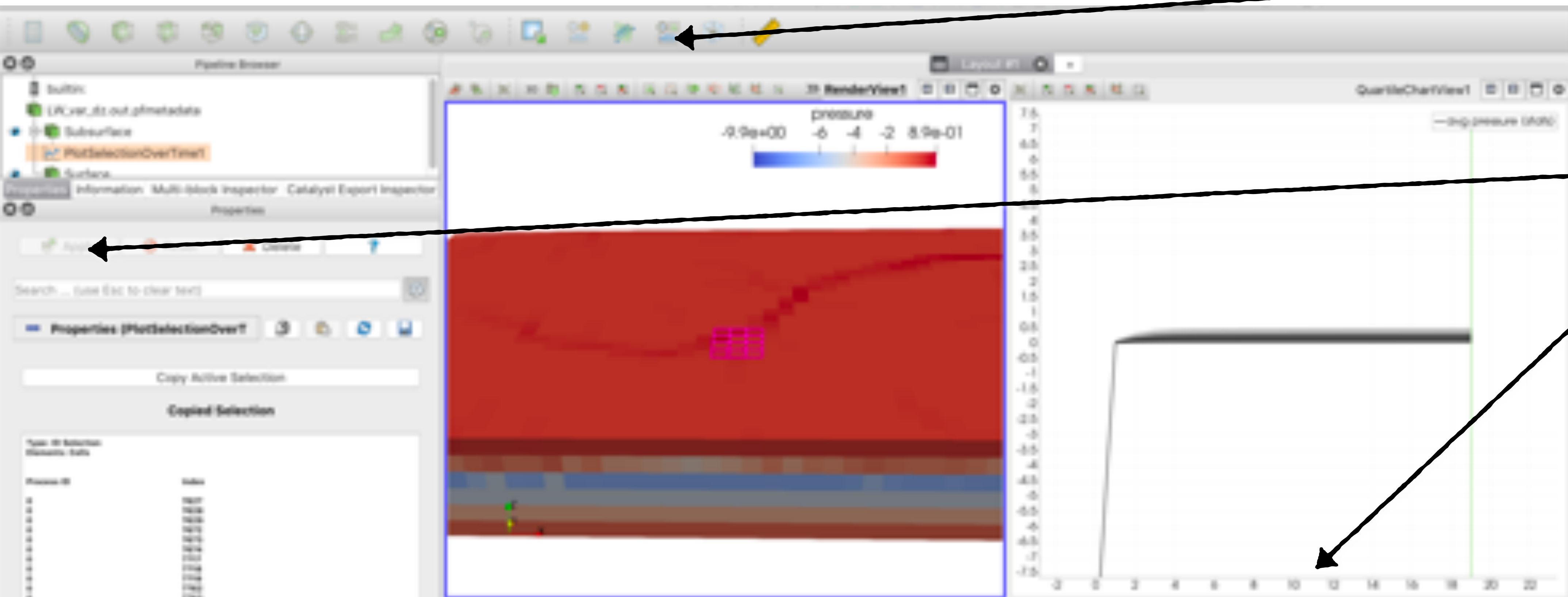
# Tools for ParFlow

- Now let's plot values of a selection over time.

Click on the "Plot Selection Over Time" toolbar button.

Click Apply in the Properties panel.

New chart view appears with timestep on the horizontal axis.



# Tools for ParFlow

- Loading plugins
  - Reading and animating data
  - Plotting selections over time
- Colormaps for stream flow
  - Simple calculations

**TODO**

# Overview

- Data types & filter pipelines in VTK & ParaView 20 min
- Visualization is composition 20 min
- Tools for ParFlow 45 min
- Going further with Python and NumPy 45 min
- Catalyst and web demonstrations 20 min

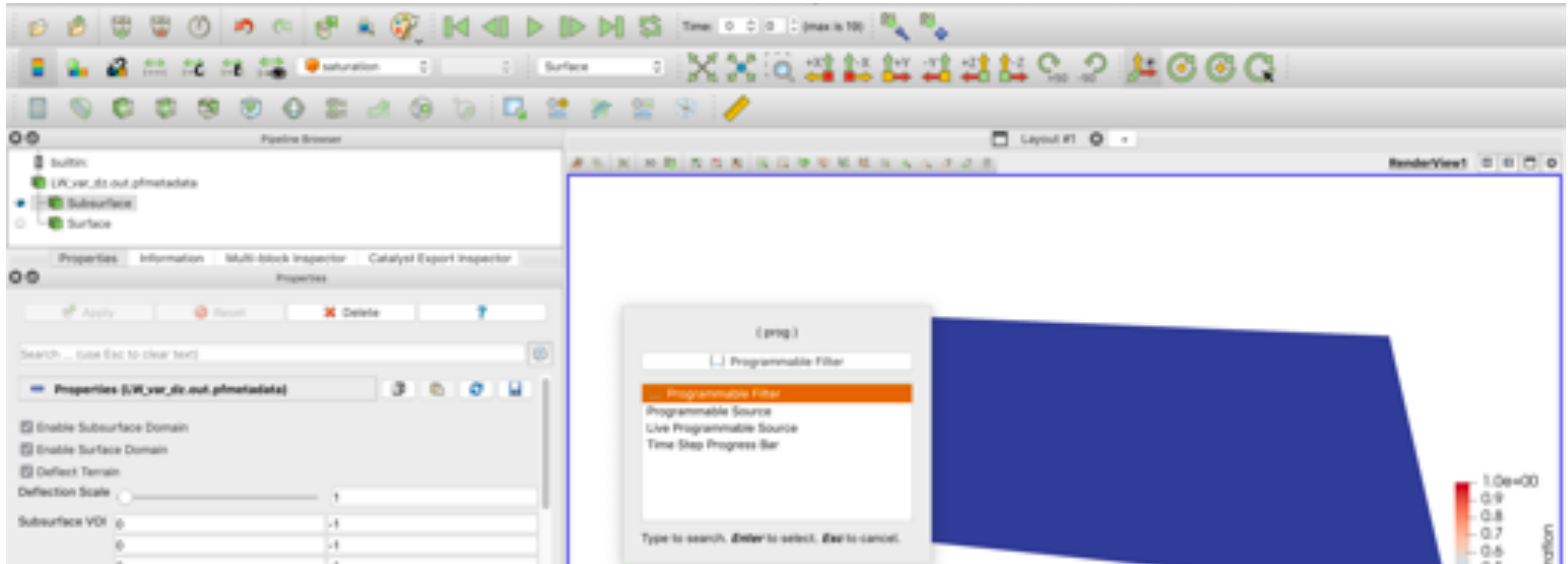
# Python and NumPy

- ParaView makes it easy to write filters in Python and use them interactively.
- Step 1: Python Programmable Filter
- Step 2: Python filter plugins
- Step 3: Fancier computations



# Programmable Filter

- Let's sum the saturation over the entire volume so we can plot the total saturation sum over time.



# Programmable Filter

- Enter a script into the programmable filter's editor, which can be detached from the panel.
- Click apply, then look at the information panel to confirm that the filter added a new array.



# Programmable Filter

- Enter a script into the programmable filter's editor, which can be detached from the panel.
- Click apply, then look at the information panel to confirm that the filter added a new array.





# Programmable Filter

This marks the new array as a time-varying array so that the filter will attempt to plot it.

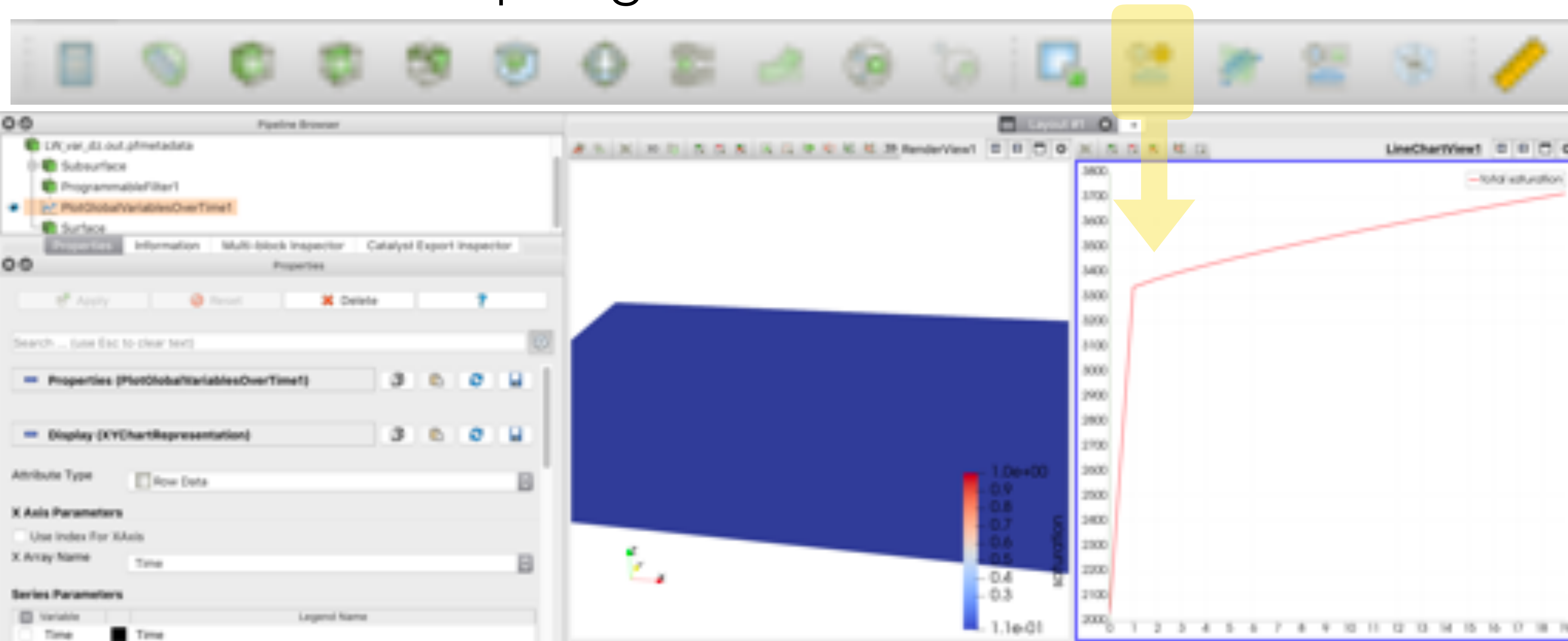
```
from paraview.vtk.util.numpy_support import \
    import numpy_to_vtk, vtk_to_numpy
import numpy as np
import vtk

pIn = self.GetInput()
pOut = self.GetOutput()
# Copy the input to the output:
pOut.ShallowCopy(pIn)
# Now sum the saturation using numpy:
cd = pIn.GetCellData()
sat = vtk_to_numpy(cd.GetArray('saturation'))
ss = np.sum(sat)
# Add the result as field data:
arr = numpy_to_vtk(ss)
arr.SetName('total saturation')
arr.GetInformation().Set(
    vtk.vtkIndexedReader.GLOBAL_TEMPORAL_VARIABLE(), 1)
pOut.GetFieldData().AddArray(arr)
```



# Programmable Filter

- Now we can plot global data over time.



# Python Filter Plugins

- To reuse a Python programmable filter, encapsulate it into a custom plugin.
- Plugins can be autoloaded and filters added to a toolbar, so access to the filter is easy.

# Python and NumPy

- ParaView makes it easy to write filters in Python and use them interactively.
- Step 1: Python Programmable Filter
- Step 2: Python filter plugins
- Step 3: Fancier computations

**TODO**

# Tools for ParFlow

- Loading plugins
- Reading and animating data
- Plotting selections over time
- Colormaps for stream flow
- Simple calculations



# Overview

- Data types & filter pipelines in VTK & ParaView 20 min
- Visualization is composition 20 min
- Tools for ParFlow 45 min
- Going further with Python and NumPy 45 min
- Catalyst and web demonstrations 20 min

Catalyst, Cinema, ParaViewWeb

# Acknowledgments



This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0019609





# Kitware



- Software process
- Scientific visualization & HPC
- Cloud data & analytics
- Medical computing
- Computer vision

# Kitware



- Collaborative R&D
- Custom software

- Support
- Training