SAND2008-6160P

Large Scale Visualization with ParaView

Supercomputing 08 Tutorial

November 17, 2008

Kenneth Moreland John Greenfield W. Alan Scott
Sandia National Laboratories

Utkarsh Ayachit Berk Geveci David DeMarle
Kitware Inc.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Outline

- Introduction
- Basic Usage
- Visualizing Large Models

To Follow Along…

- Install ParaView 3.4.
  - http://www.paraview.org → Download
- Get example material.
  - http://www.paraview.org/Wiki/SC07_ParaView_Tutorial
  - Also available on tutorial CD.

Introduction

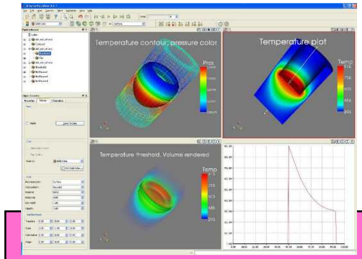
What is ParaView?

- An open-source, scalable, multi-platform visualization application.
- Support for distributed computation models to process large data sets.
- An open, flexible, and intuitive user interface.
- An extensible, modular architecture based on open standards.
- Commercial maintenance and support.

Current ParaView Usage

- Used by academic, government, and commercial institutions worldwide.
  - Downloaded ~3K times/month.
- Used for all ranges of data size.
- Current landmarks of SNL usage:
  - 6 billion structured cells.
  - Billions of AMR cells.
  - 250 million unstructured cells.

ParaView Application Architecture


Temperature contour, pressure color
Temperature plot
Temperature threshold, Volume rendered

ParaView Client | pvpython | Custom App

UI (Qt Widgets, Python Wrappings)

ParaView Server

VTK

OpenGL | MPI | IceT | Etc.

ParaView Development

- Started in 2000 as collaborative effort between Los Alamos National Laboratories and Kitware Inc. (lead by James Ahrens).
  - ParaView 0.6 released October 2002.
- September 2005: collaborative effort between Sandia National Laboratories, Kitware Inc. and CSimSoft to rewrite user interface to be more user friendly and develop quantitative analysis framework.
  - ParaView 3.0 released in May 2007.

Current Funding

ASC™

Sandia National Laboratories

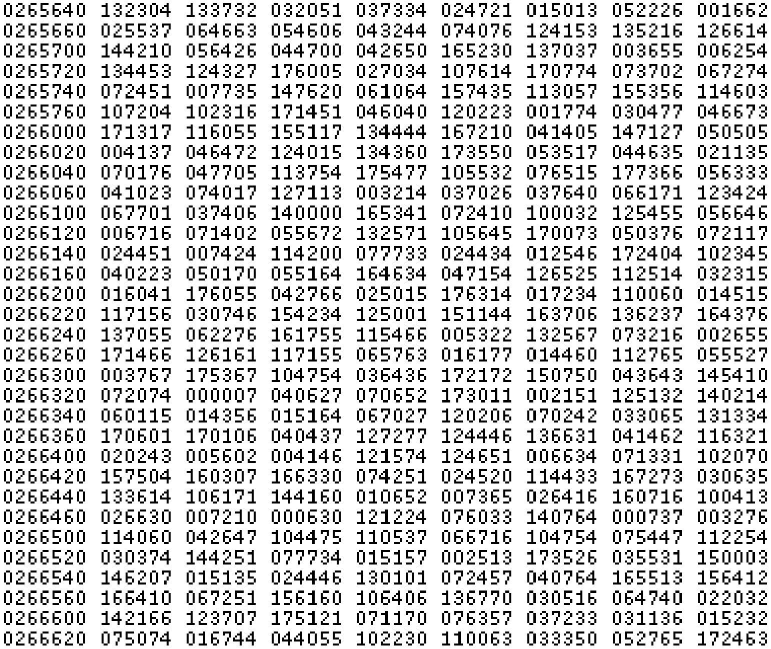
Los Alamos NATIONAL LABORATORY

UltraVis
SciDAC Institute for Ultrascale Visualization

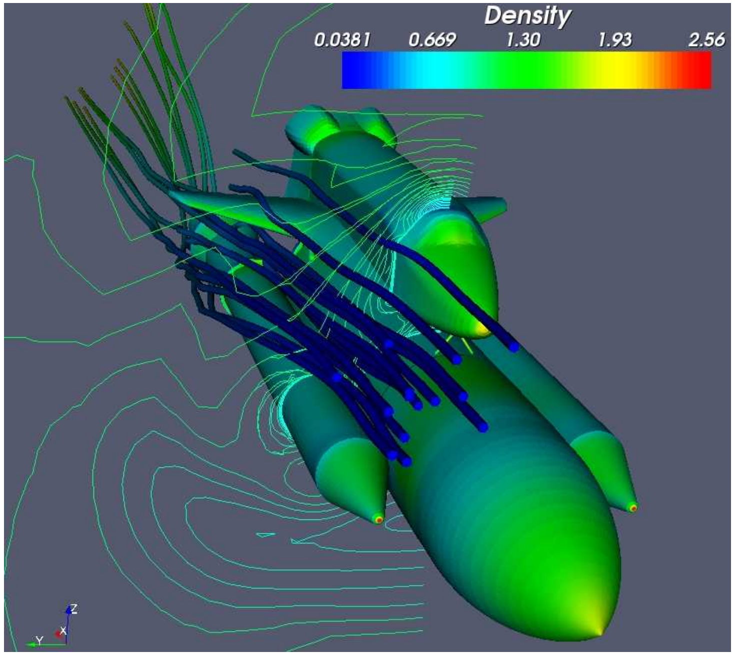
SciDAC
Scientific Discovery through Advanced Computing

- Army SBIR
- ERDC Contract
- US NSF SBIR
- Other contributors
  - Swiss National Supercomputing Centre
- Support Contracts
  - Electricity de France
  - Mirarco
  - Oil Industry

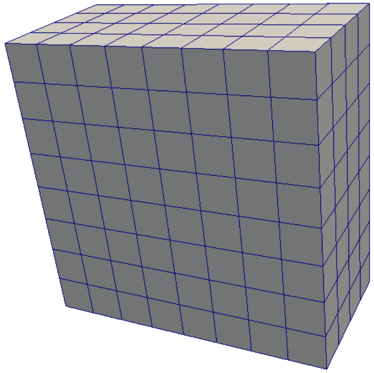
Basics of Visualization



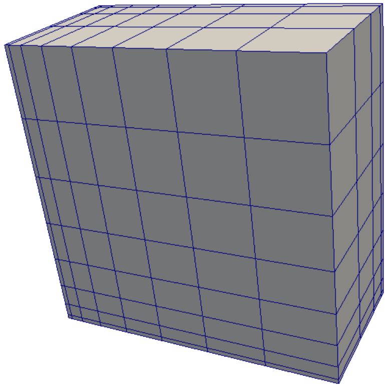
```
0265640 132304 133732 032051 037334 024721 015013 052226 001662
0265660 025537 064663 054606 043244 074076 124153 135216 126614
0265700 144210 056426 044700 042650 165230 137037 003655 006254
0265720 134453 124327 176005 027034 107614 170774 073702 067274
0265740 072451 007735 147620 061064 157435 113057 155356 114603
0265760 107204 102316 171451 046040 120223 001774 030477 046673
0266000 171317 116055 155117 134444 167210 041405 147127 050505
0266020 004137 046472 124015 134360 173550 053517 044635 021135
0266040 070176 047705 113754 175477 105532 076515 177366 056333
0266060 041023 074017 127113 003214 037026 037640 066171 123424
0266100 067701 037406 140000 165341 072410 100032 125455 056646
0266120 006716 071402 055672 132571 105645 170073 050376 072117
0266140 024451 007424 114200 077733 024434 012546 172404 102345
0266160 040223 050170 055164 164634 047154 126525 112514 032315
0266200 016041 176055 042766 025015 176314 017234 110060 014515
0266220 117156 030746 154234 125001 151144 163706 136237 164376
0266240 137055 062276 161755 115466 005322 132567 073216 002655
0266260 171466 126161 117155 065763 016177 014460 112765 055527
0266300 003767 175367 104754 036436 172172 150750 043643 145410
0266320 072074 000007 040627 070652 173011 002151 125132 140214
0266340 060115 014356 015164 067027 120206 070242 033065 131334
0266360 170601 170106 040437 127277 124446 136631 041462 116321
0266400 020243 005602 004146 121574 124651 006634 071331 102070
0266420 157504 160307 166330 074251 024520 114433 167273 030635
0266440 133614 106171 144160 010652 007365 026416 160716 100413
0266460 026630 007210 000630 121224 076033 140764 000737 003276
0266500 114060 042647 104475 110537 066716 104754 075447 112254
0266520 030374 144251 077734 015157 002513 173526 035531 150003
0266540 146207 015135 024446 130101 072457 040764 165513 156412
0266560 166410 067251 156160 106406 136770 030516 064740 022032
0266600 142166 123707 175121 071170 076357 037233 031136 015232
0266620 075074 016744 044055 102230 110063 033350 052765 172463
```



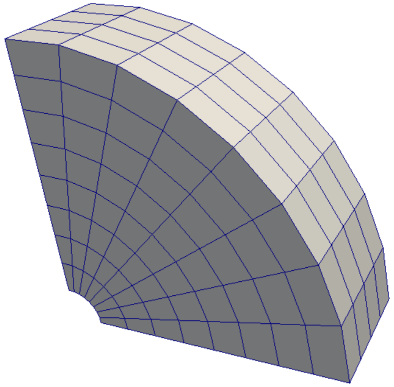
Data Types



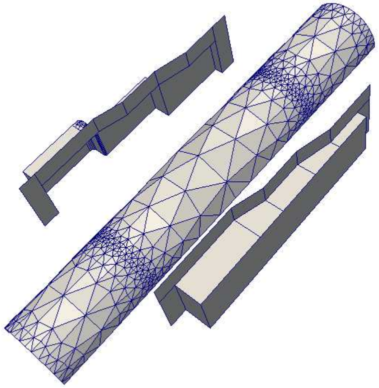
Uniform Rectilinear
(Image Data)



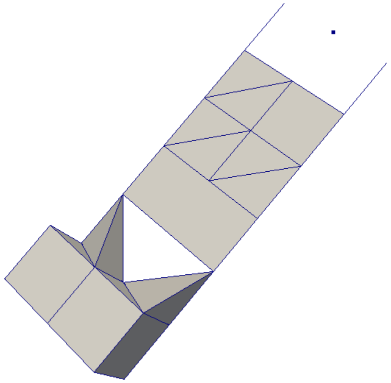
Non-Uniform Rectilinear
(Rectilinear Data)



Curvilinear
(Structured Data)



Polygonal
(Poly Data)



Unstructured Grid

Multi-block

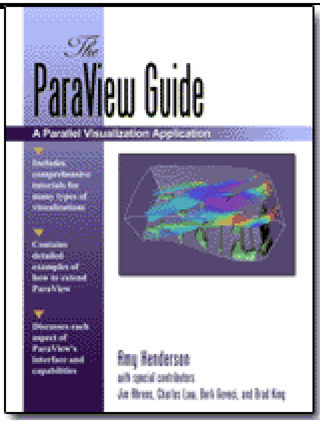
Hierarchical Adaptive
Mesh Refinement
(AMR)

Hierarchical Uniform
AMR

Octree

More Information

- Online Help
- *The ParaView Guide*


The ParaView Guide

A Parallel Visualization Application

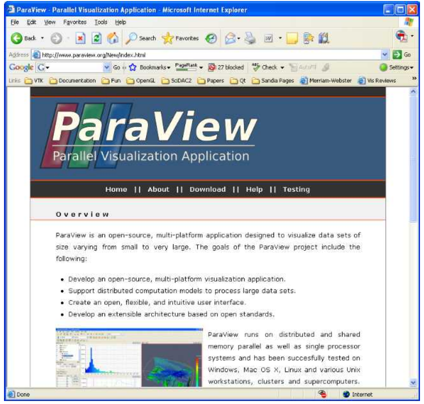
Includes comprehensive tutorials for many types of visualizations

Contains detailed examples of how to extend ParaView

Discusses each aspect of ParaView's interface and capabilities

Amy Henderson
with special contributors
Jim Ahrens, Charles Law, Berk Geveci, and Brad King

- The ParaView web page
  - www.paraview.org


ParaView - Parallel Visualization Application - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://www.paraview.org/New/index.html
Go
Google
Go
Bookmarks
PageRank
27 blocked
Check
AutoFill
Settings
Links VTK Documentation Fun OpenGL SciDAC2 Papers Qt Sandia Pages Merriam-Webster Vis Reviews
ParaView
Parallel Visualization Application
Home || About || Download || Help || Testing
Done
Internet

Overview

ParaView is an open-source, multi-platform application designed to visualize data sets of size varying from small to very large. The goals of the ParaView project include the following:

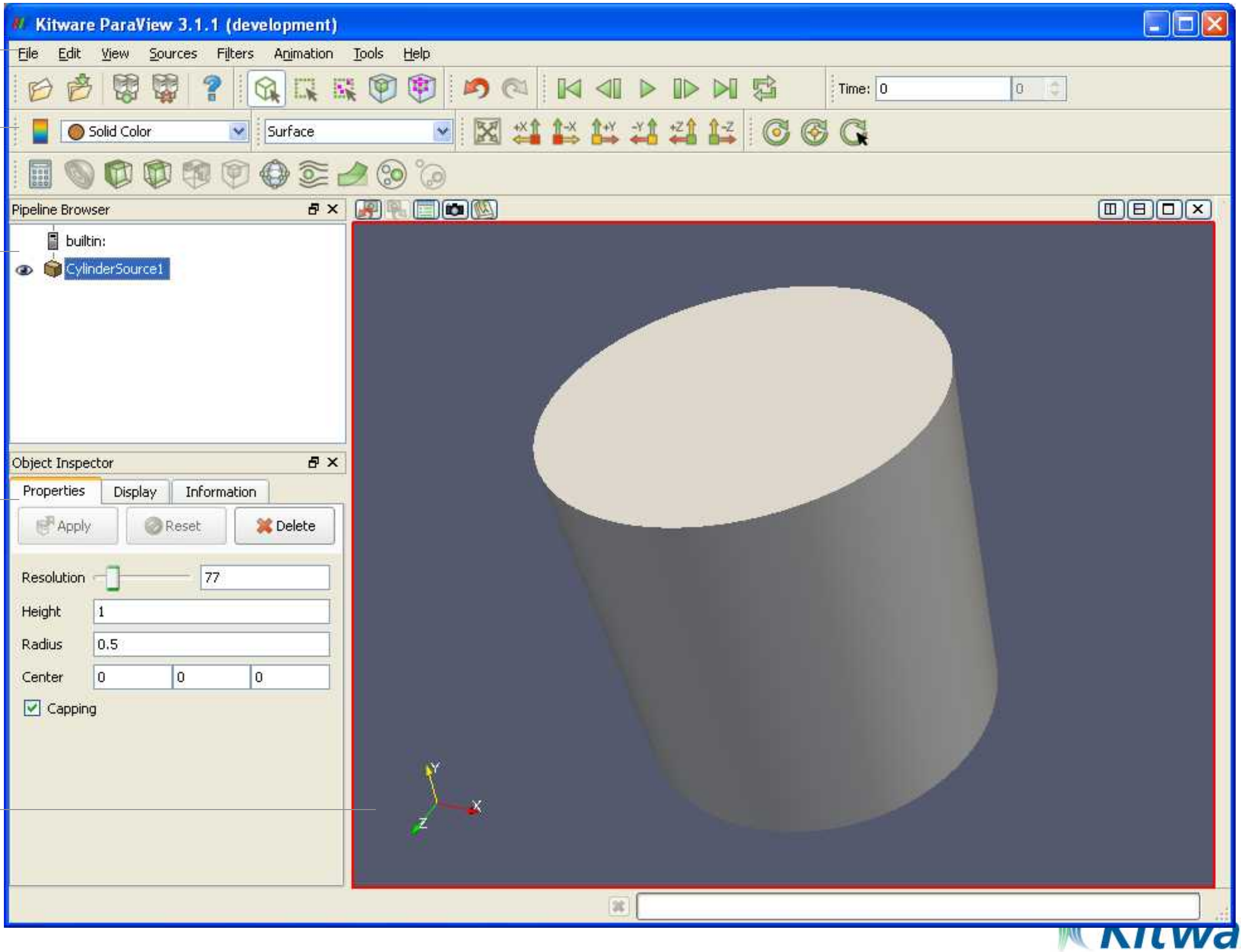
- Develop an open-source, multi-platform visualization application.
- Support distributed computation models to process large data sets.
- Create an open, flexible, and intuitive user interface.
- Develop an extensible architecture based on open standards.

ParaView runs on distributed and shared memory parallel as well as single processor systems and has been succesfully tested on Windows, Mac OS X, Linux and various Unix workstations, clusters and supercomputers.

- ParaView mailing list
  - paraview@paraview.org

Basic Usage

User Interface



Menu Bar

Toolbars

Pipeline Browser

Object Inspector

3D View

Resolution 77

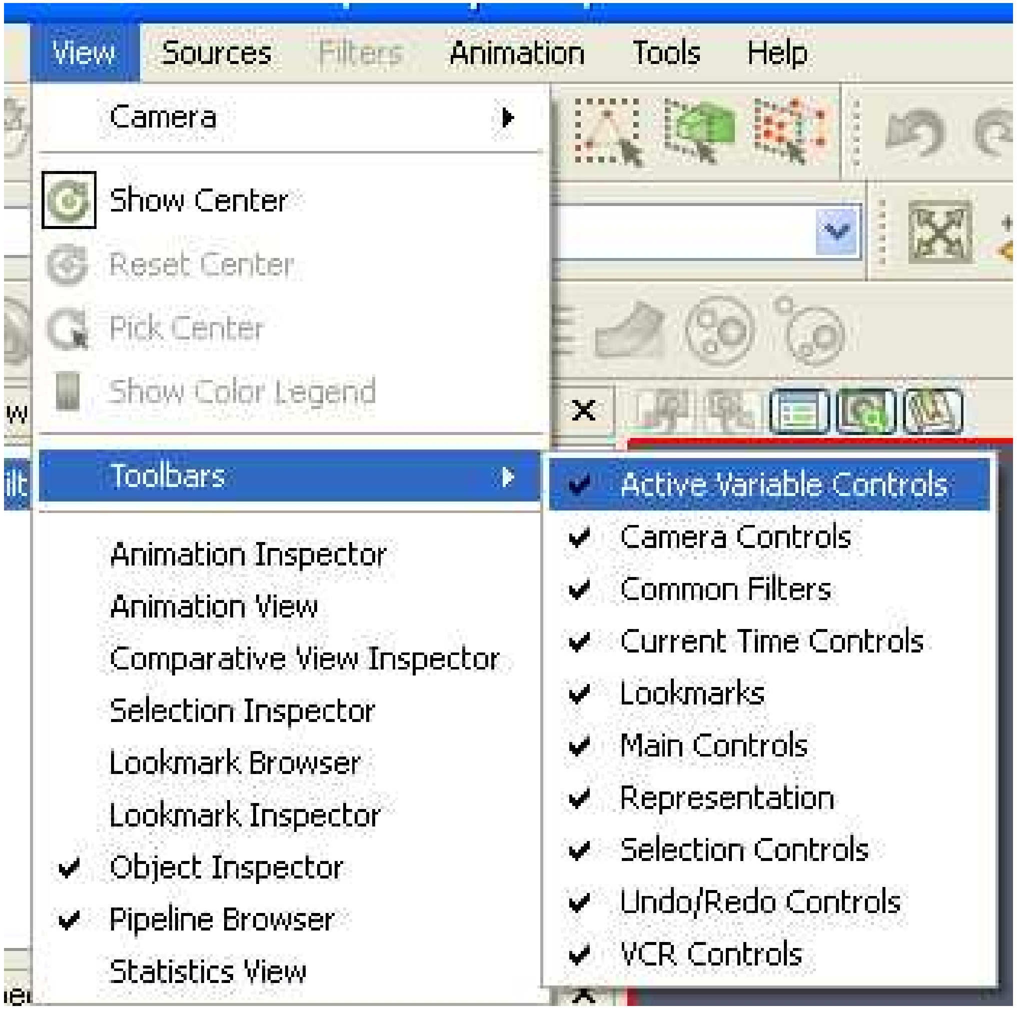
Height 1

Radius 0.5

Center 0 0 0

☑ Capping

Getting Back GUI Components

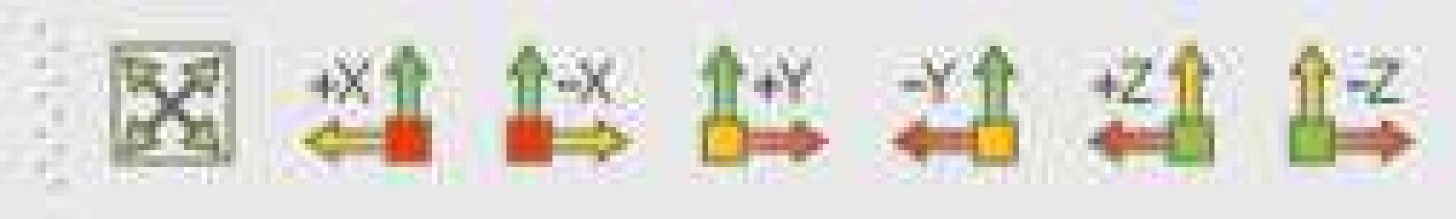


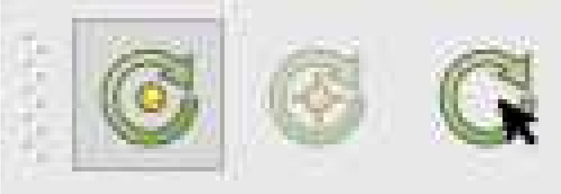
Creating a Cylinder Source

1. Go to the Source menu and select Cylinder.
2. Click the Apply button to accept the default parameters.

Simple Camera Manipulation

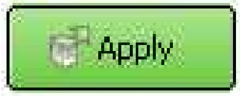
- Drag left, middle, right buttons for rotate, pan, zoom.
  - –Also use Shift, Ctrl, Alt modifiers.



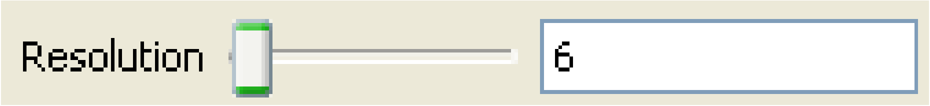


Creating a Cylinder Source

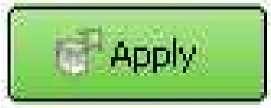
1. Go to the Source menu and select Cylinder.
2. Click the Apply button to accept the default parameters.



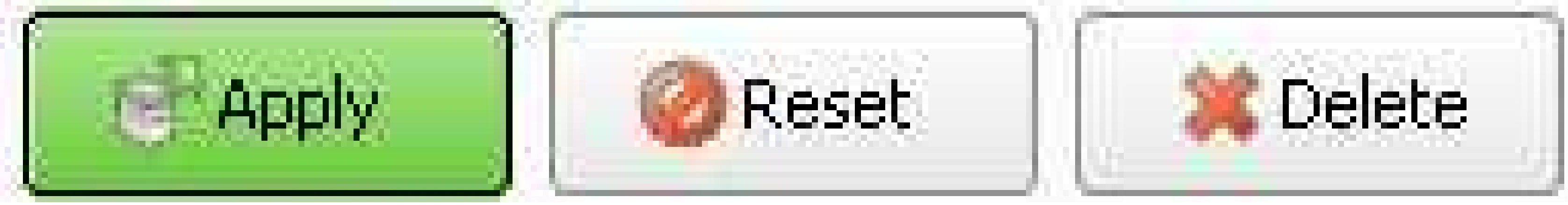
3. Increase the Resolution parameter.

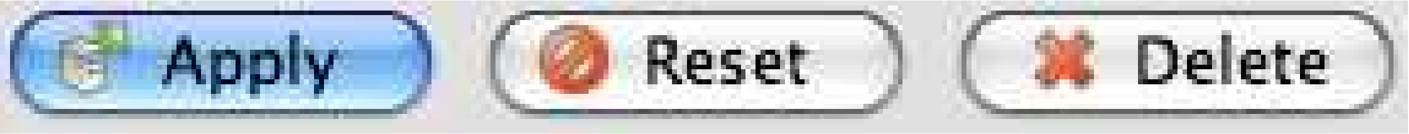

Resolution 6

4. Click the button again.



Pipeline Object Controls





Undo Redo

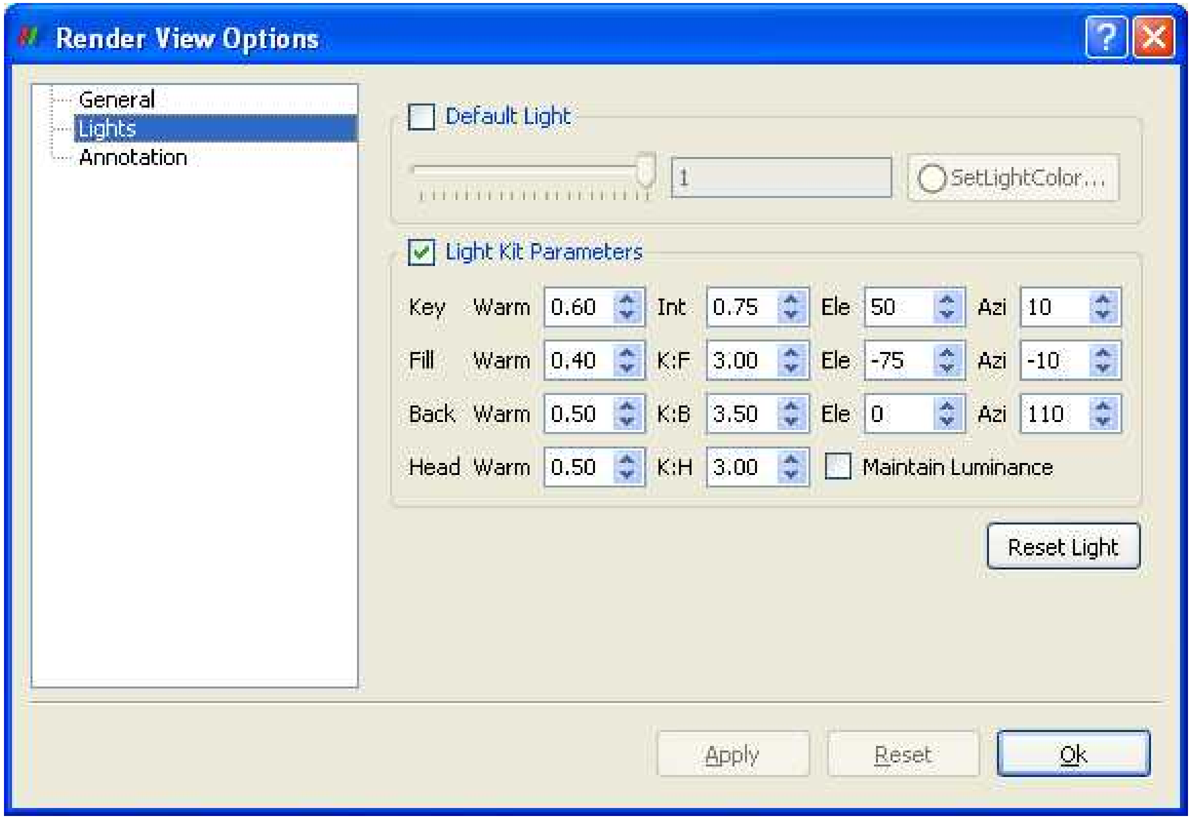
Undo

Redo

Camera Undo

Camera Redo

Render View Options



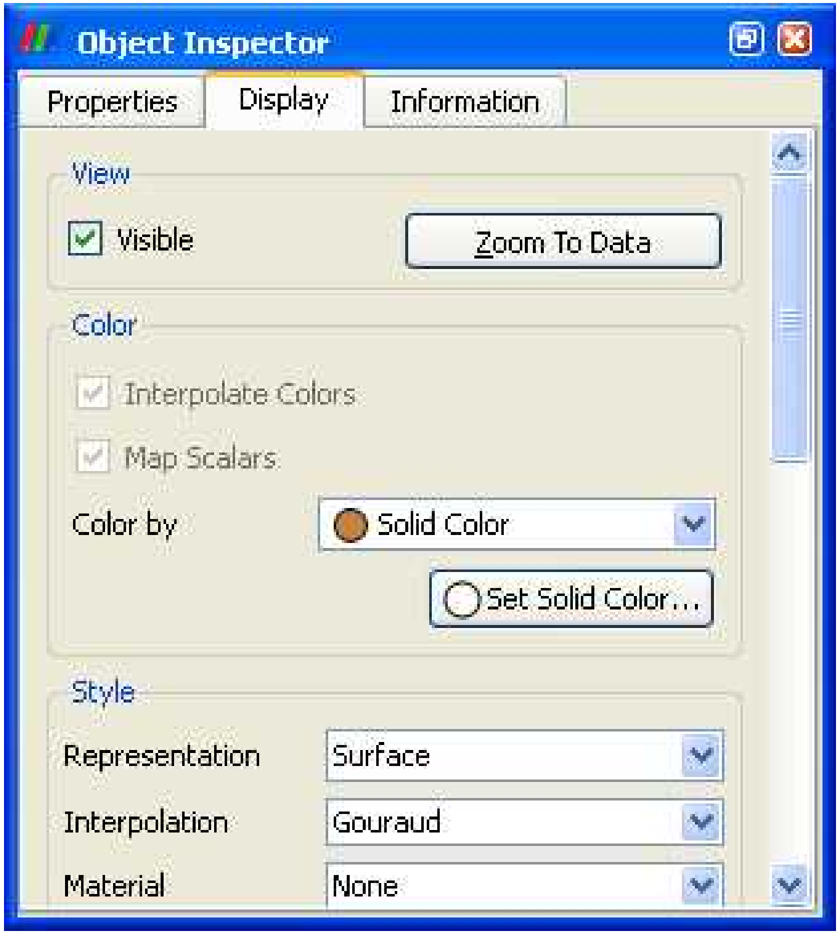
□ Default Light

1 ○ SetLightColor...

☑ Light Kit Parameters

Key	Warm	0.60	Int	0.75	Ele	50	Azi	10
Fill	Warm	0.40	K:F	3.00	Ele	-75	Azi	-10
Back	Warm	0.50	K:B	3.50	Ele	0	Azi	110
Head	Warm	0.50	K:H	3.00	□ Maintain Luminance			

Display Tab



☑ Visible

☑ Interpolate Colors

☑ Map Scalars

Color by: Solid Color

○ Set Solid Color...

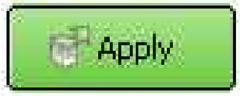
Representation: Surface

Interpolation: Gouraud

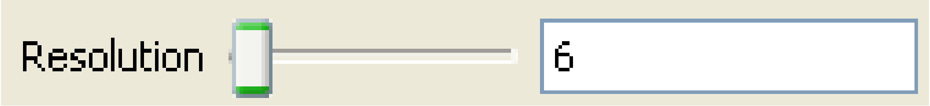
Material: None

Creating a Cylinder Source

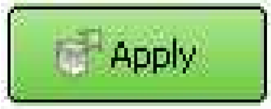
1. Go to the Source menu and select Cylinder.
2. Click the Apply button to accept the default parameters.



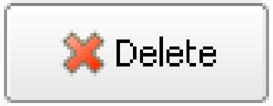
3. Increase the Resolution parameter.



Resolution 6

4. Click the  button again.

5. Delete the Cylinder.

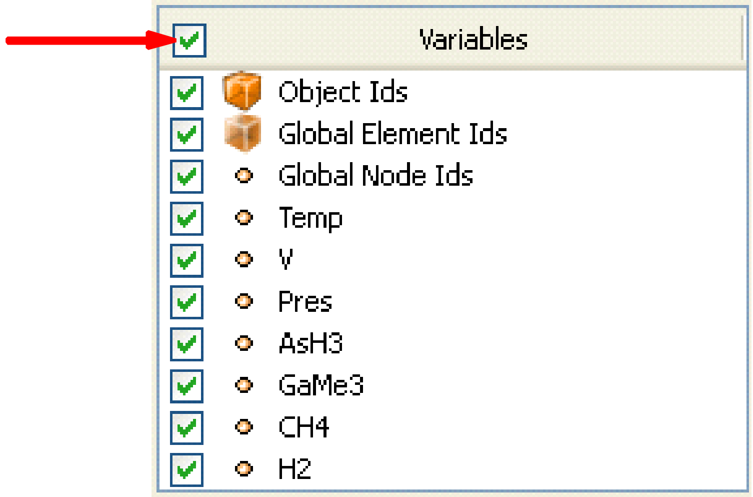


Supported Data Types

- ParaView Data (.pvd)
- VTK (.vtp, .vtu, .vti, .vts, .vtr)
- VTK Multi Block (.vtm, .vtmb, .vtmg, .vthd, .vthb)
- Partitioned VTK (.pvtu, .pvti, .pvts, .pvtr)
- VTK Legacy (.vtk)
- Exodus
- XDMF (.xmf, .xdmf)
- LS-DYNA
- SpyPlot CTH
- EnSight (.case, .sos)
- BYU (.g)
- Protein Data Bank (.pdb)
- XMol Molecule
- PLOT3D
- Digital Elevation Map (.dem)
- VRML (.wrl)
- PLY Polygonal File Format
- Stereo Lithography (.stl)
- Gaussian Cube File (.cube)
- POP Ocean Files
- AVS UCD (.inp)
- Meta Image (.mhd, .mha)
- Facet Polygonal Data
- Phasta Files (.pht)
- PNG Image Files
- Raw Image Files
- Comma Separated Values (.csv)

Load disk_out_ref.ex2

1. Open the file disk_out_ref.ex2.
2. Load all data variables.



☑ Variables

☑ Object Ids

☑ Global Element Ids

☑ Global Node Ids

☑ Temp

☑ V

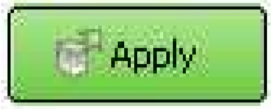
☑ Pres

☑ AsH3

☑ GaMe3

☑ CH4

☑ H2

3. Click 

Data Representation

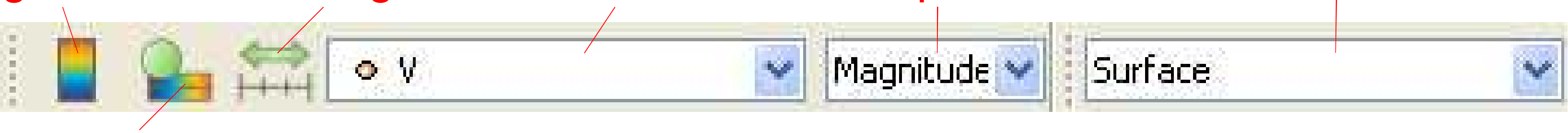
Toggle Color Legend

Reset Scalar Range

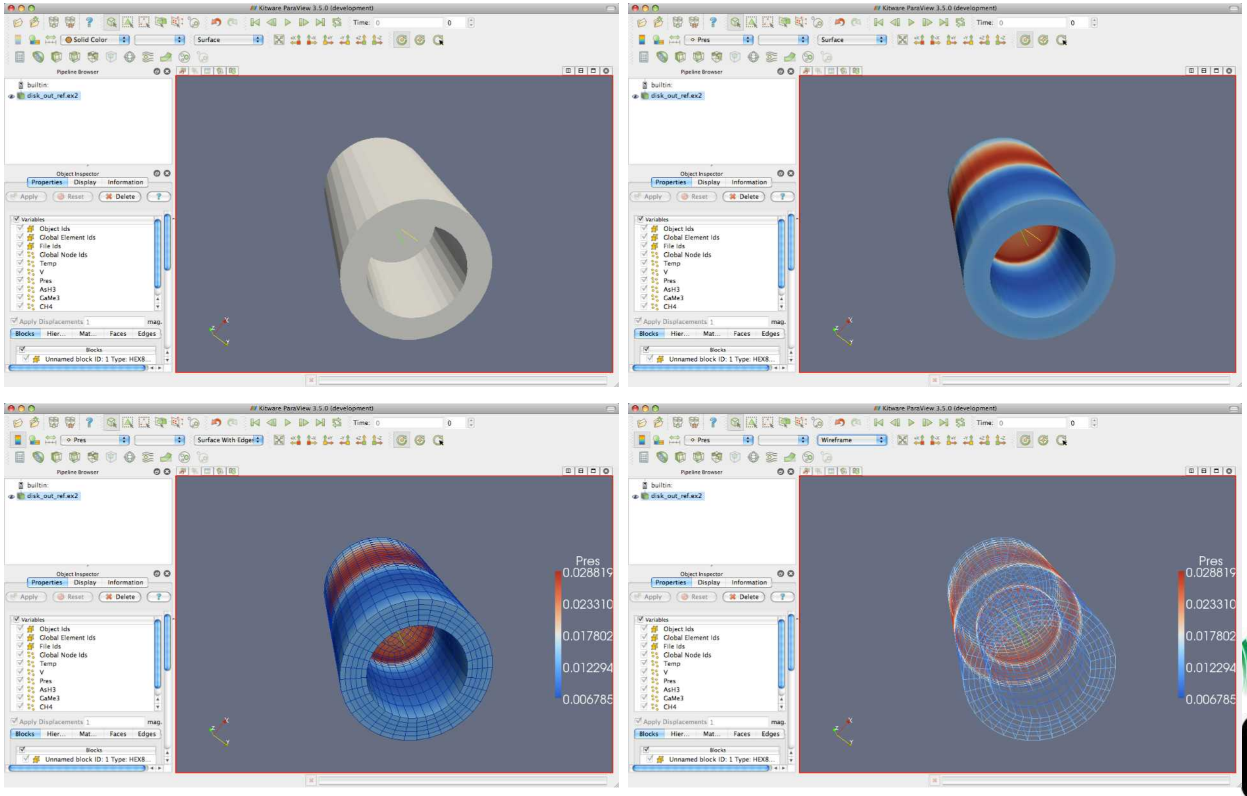
Mapped Variable

Vector Component

Representation



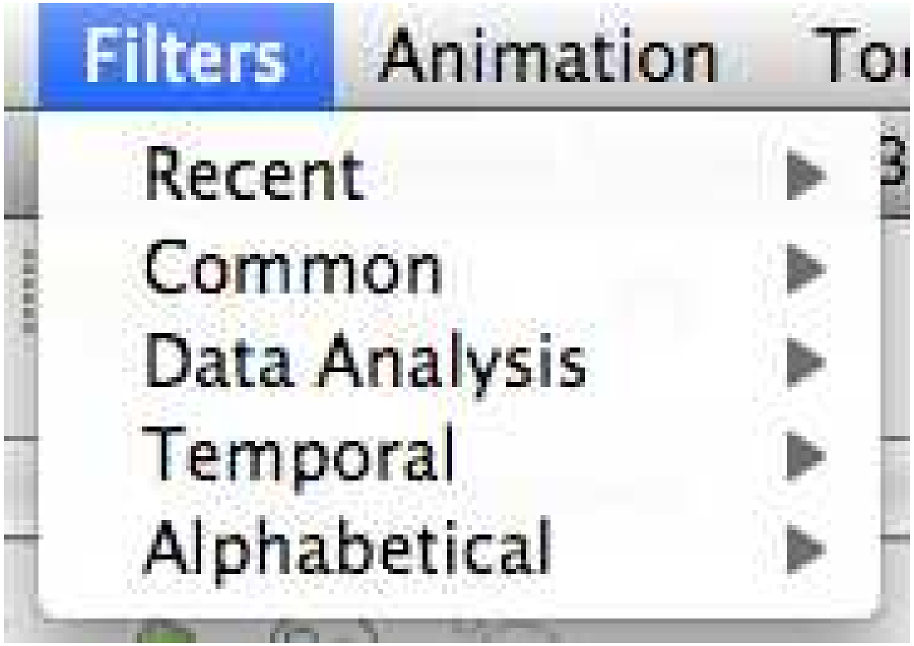
Edit Colors



Common Filters

- Calculator
- Contour
- Clip
- Slice
- Threshold
- Extract Subset
- Glyph
- Stream Tracer
- Warp (vector)
- Group Datasets
- Extract Group

Filters Menu

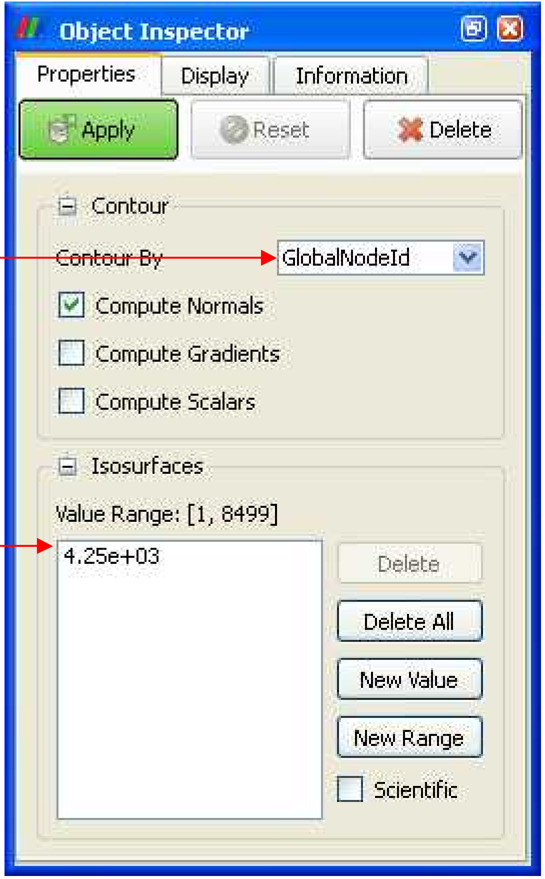


Apply a Filter

1. Make sure that disk_out_ref.ex2 is selected in the pipeline browser.
2. Select the contour filter.

Apply a Filter

3. Change parameters to create an isosurface at Temp = 400K.



⊟ Contour

Contour By: GlobalNodeId

☑ Compute Normals

□ Compute Gradients

□ Compute Scalars

⊟ Isosurfaces

Value Range: [1, 8499]

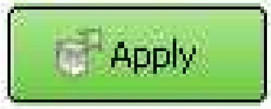
4.25e+03

□ Scientific

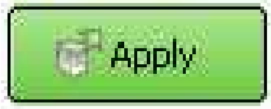
Change to Temp

Change to 400

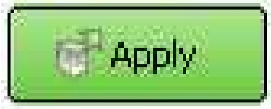
Apply a Filter

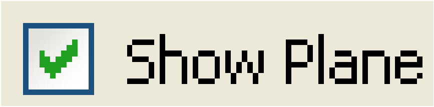
1. Make sure that disk_out_ref.ex2 is selected in the pipeline browser.
2. Select the contour filter.
3. Change parameters to create an isosurface at Temp = 400K.
4. 

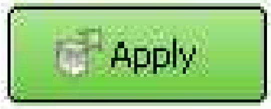
Create a Cutaway Surface

1. Select disk_out_ref.ex2 in the pipeline browser.
2. From the menu bar, select Filters → Alphabetical → Extract Surface.
3. 

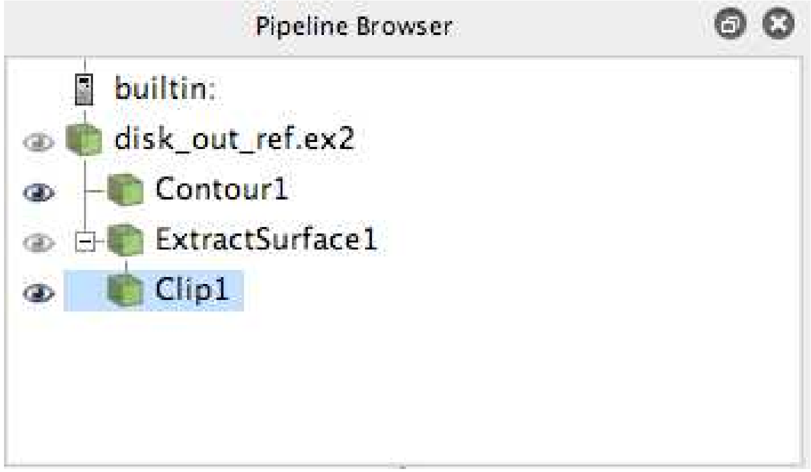
Create a Cutaway Surface

1. Select disk_out_ref.ex2 in the pipeline browser.
2. From the menu bar, select Filters → Alphabetical → Extract Surface.
3. 

4. Create a clip filter.
5. Uncheck Show Plane. 

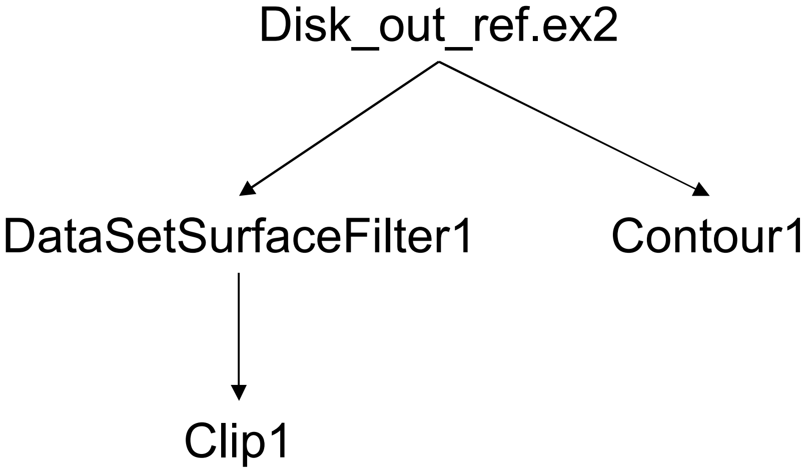
6. 

Pipeline Browser Structure

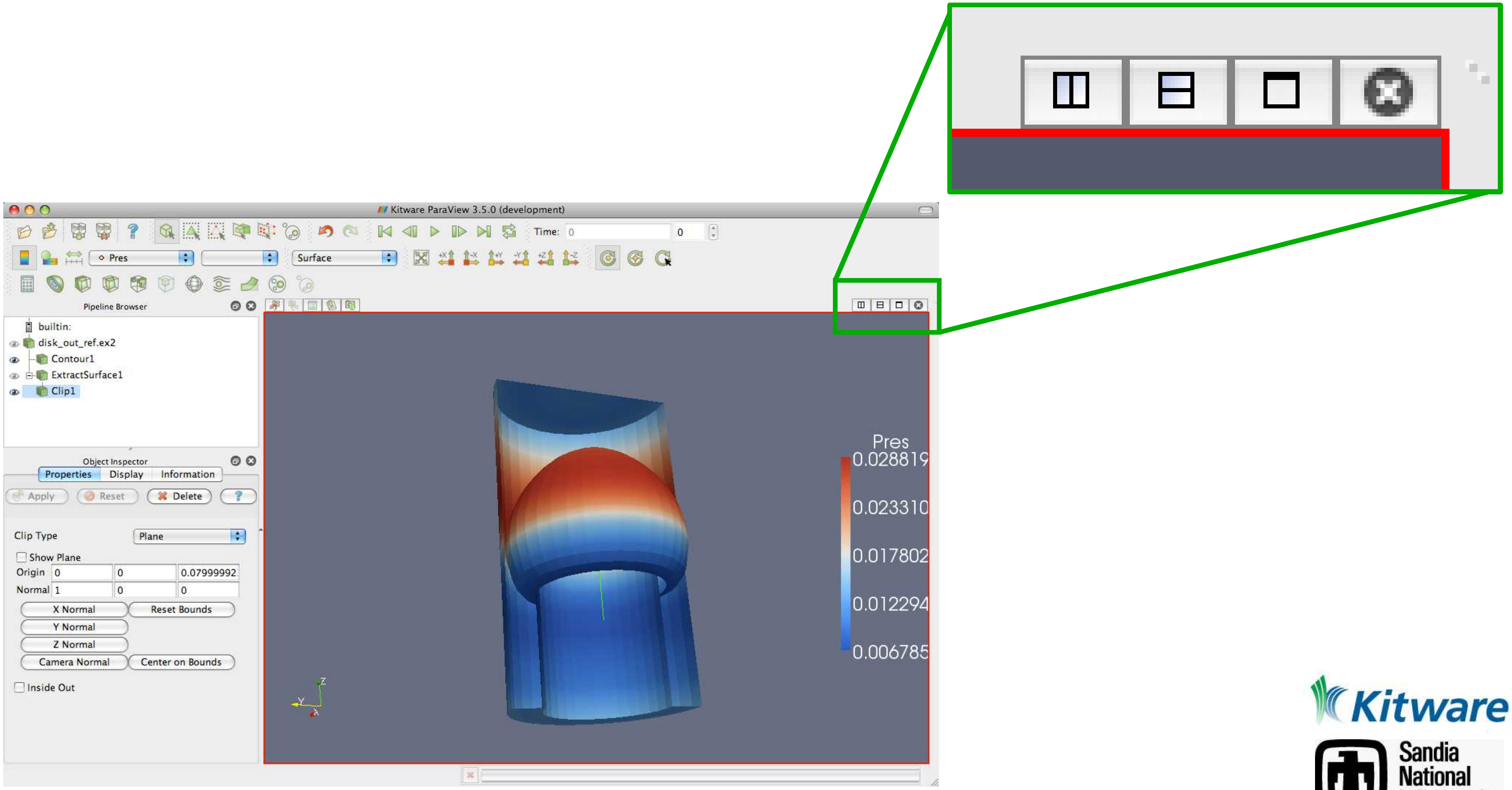


Pipeline Browser

builtin:
disk_out_ref.ex2
Contour1
ExtractSurface1
Clip1



Multiview



Kitware ParaView 3.5.0 (development)

Time: 0 0

Pres

Surface

Pipeline Browser

- builtin:
- disk_out_ref.ex2
- Contour1
- ExtractSurface1
- Clip1

Object Inspector

Clip Type: Plane

□ Show Plane

Origin	0	0	0.07999992
Normal	1	0	0

□ Inside Out

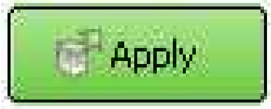
Multiview

1. Split the view horizontally. ◫
2. Make disk_out_ref.ex2 visible. 👁
3. Color surface by Temp.

Multiview

1. Split the view horizontally.
2. Make disk_out_ref.ex2 visible.
3. Color surface by Temp.
4. Add clip filter.
5. Uncheck Show Plane.

☑ Show Plane

6. 

Apply

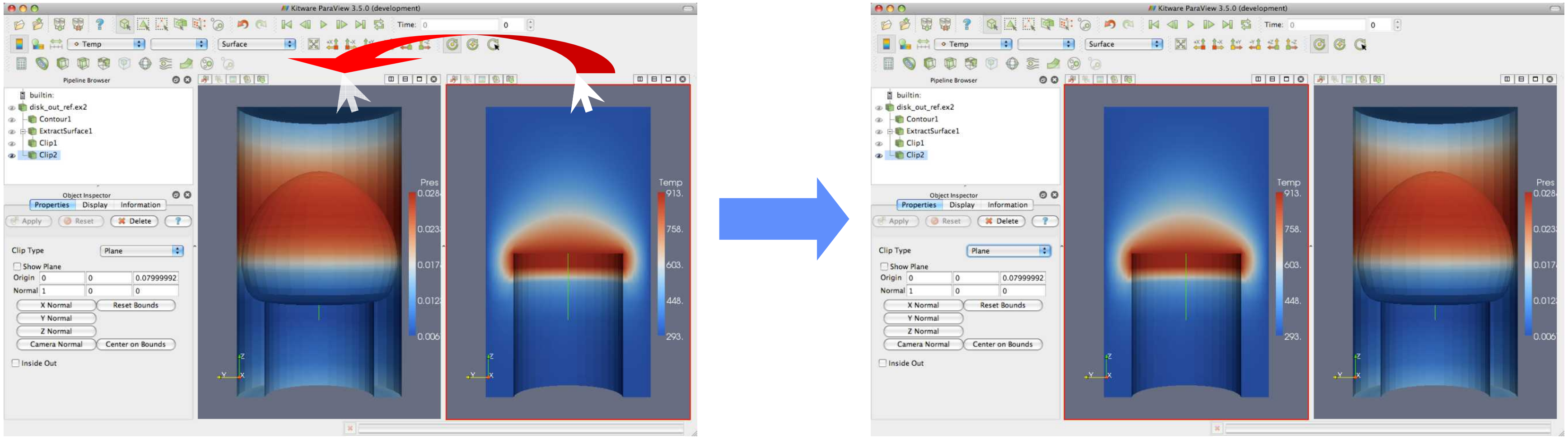
Multiview

1. Split the view horizontally.
2. Make disk_out_ref.ex2 visible.
3. Color surface by Temp.
4. Add clip filter.
5. Uncheck Show Plane. ☑ Show Plane
6. Apply
7. Right-click view, Link Camera…
8. Click other view.

Multiview

1. Split the view horizontally.
2. Make disk_out_ref.ex2 visible.
3. Color surface by Temp.
4. Add clip filter.
5. Uncheck Show Plane. ☑ Show Plane
6. Apply
7. Right-click view, Link Camera…
8. Click other view.
9. Click +X

Modifying Views

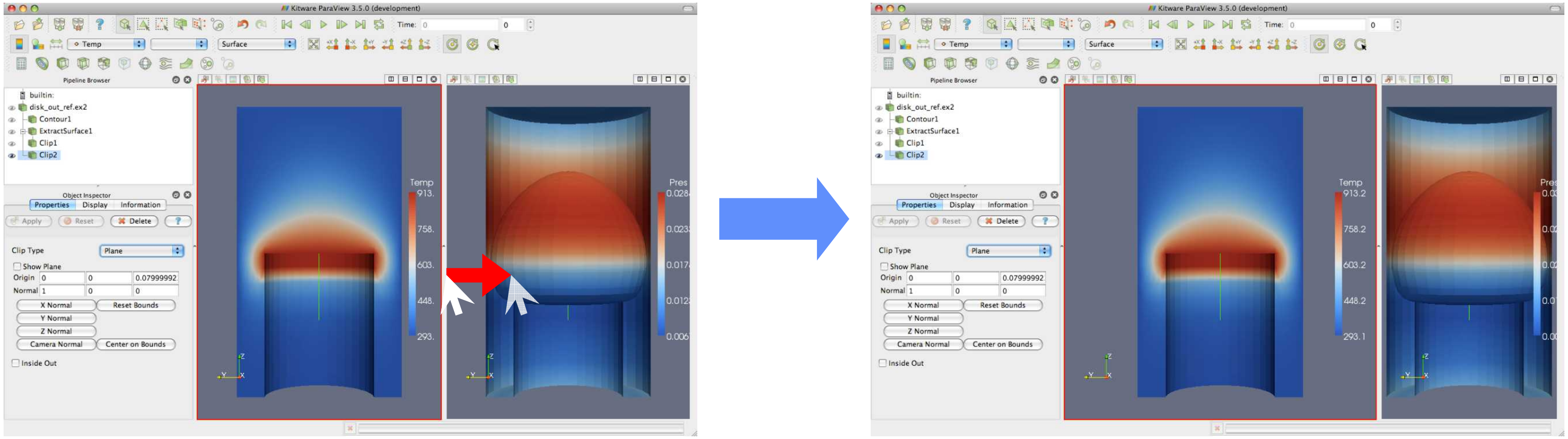


Clip Type: Plane

Origin: 0 0 0.07999992

Normal: 1 0 0

Modifying Views



Kitware ParaView 3.5.0 (development)

Time: 0

Temp

Surface

Pipeline Browser

- builtin:
- disk_out_ref.ex2
- Contour1
- ExtractSurface1
- Clip1
- Clip2

Object Inspector

Properties | Display | Information

Apply | Reset | Delete | ?

Clip Type: Plane

□ Show Plane

Origin: 0 | 0 | 0.07999992

Normal: 1 | 0 | 0

X Normal | Reset Bounds

Y Normal

Z Normal

Camera Normal | Center on Bounds

□ Inside Out

Streamlines

1. Split view vertically, maximize
2. Make disk_out_ref.ex2 visible
3. Select disk_out_ref.ex2.
4. Add stream tracer.
5. Apply

Streamlines

1. Split view vertically, maximize
2. Make disk_out_ref.ex2 visible
3. Select disk_out_ref.ex2.
4. Add stream tracer.
5. Apply
6. Select Filters → Alphabetical → Tube
7. Apply

Getting Fancy

1. Select StreamTracer1.
2. Add glyph filter.
3. Change Vectors to V.
4. Change Glyph Type to Cone.
5.

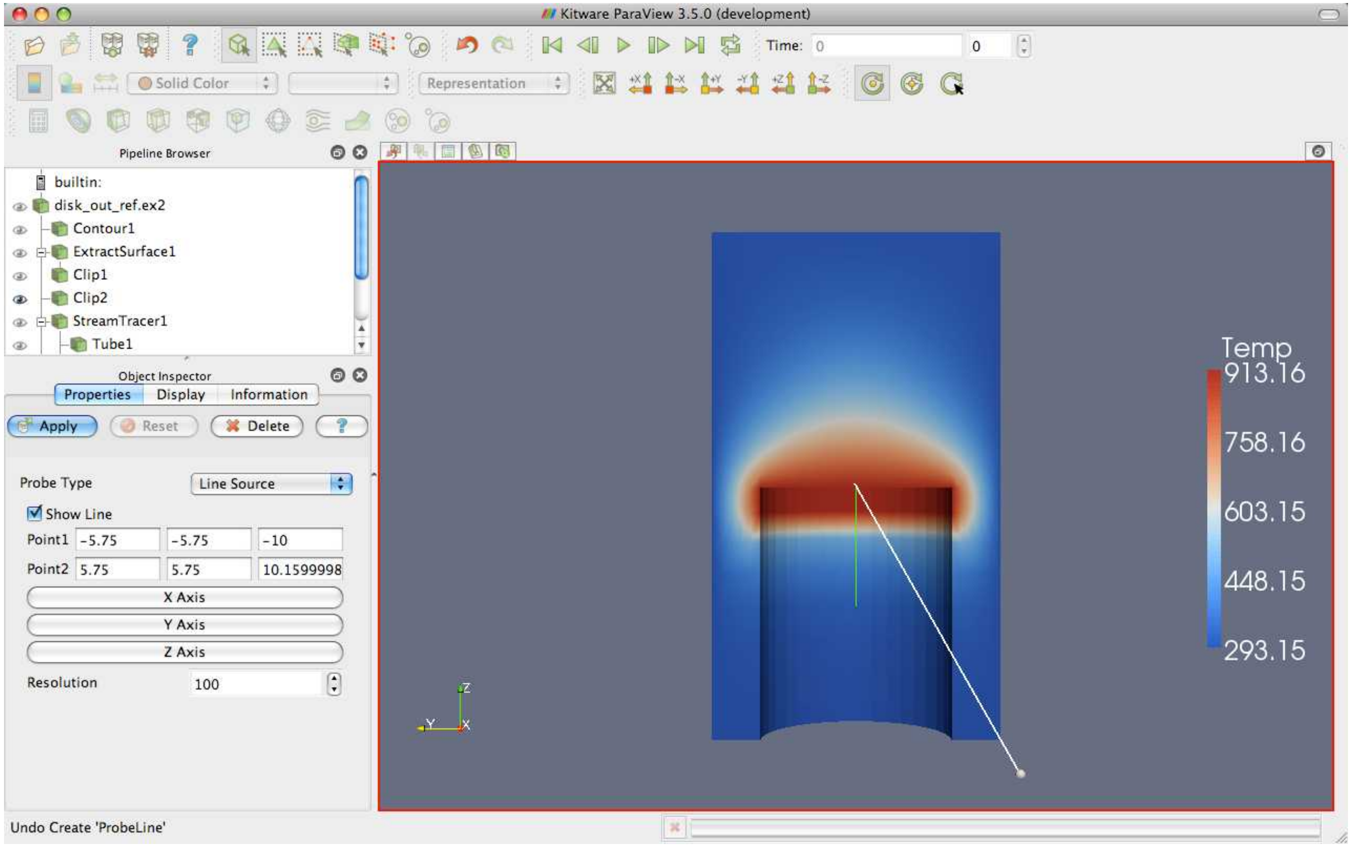
Getting Answers

- Where is the air moving the fastest? Near the disk or away from it? At the center of the disk or near its edges?
- Which way is the plate spinning?
- At the surface of the disk, is air moving toward the center or away from it?

Plotting

1. Select disk_out_ref.ex2
2. Filters → Data Analysis → Plot Over Line.

3D Widgets



Kitware ParaView 3.5.0 (development)

Pipeline Browser

- builtin:
- disk_out_ref.ex2
  - Contour1
  - ExtractSurface1
  - Clip1
  - Clip2
  - StreamTracer1
    - Tube1

Object Inspector

Probe Type: Line Source

☑ Show Line

Point1	-5.75	-5.75	-10
Point2	5.75	5.75	10.1599998

Resolution: 100

Undo Create 'ProbeLine'

Plotting

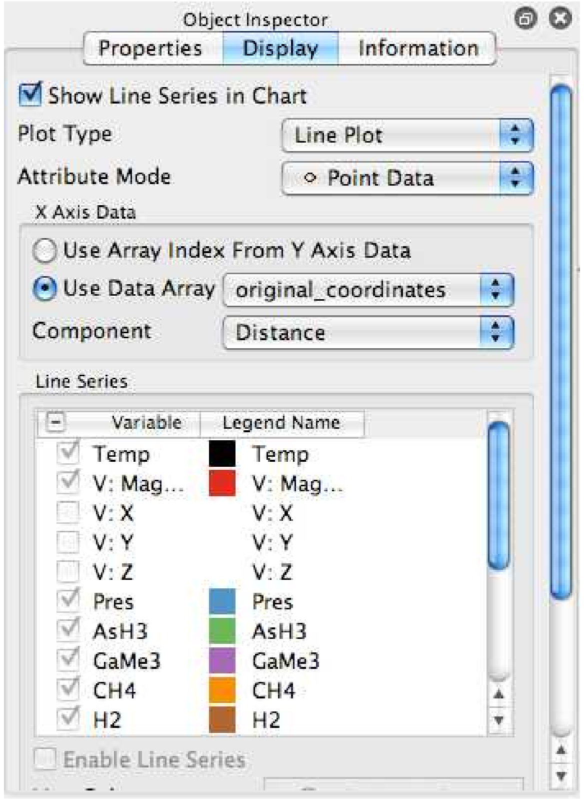
1. Select disk_out_ref.ex2
2. Filters → Data Analysis → Plot Over Line.
3. Once line satisfactorily located, Apply

Interacting with Plots

- Left, middle, right buttons to pan, zoom.
- Reset view to plot ranges.

Plots are Views

- Move them like Views.
- Save screenshots (+ vector pdf).



☑ Show Line Series in Chart

Plot Type: Line Plot

Attribute Mode: ○ Point Data

X Axis Data

○ Use Array Index From Y Axis Data

● Use Data Array: original_coordinates

Component: Distance

Line Series

Variable	Legend Name
☑ Temp	■ Temp
☑ V: Mag...	🟥 V: Mag...
□ V: X	V: X
□ V: Y	V: Y
□ V: Z	V: Z
☑ Pres	🟦 Pres
☑ AsH3	🟩 AsH3
☑ GaMe3	🟪 GaMe3
☑ CH4	🟧 CH4
☑ H2	🟫 H2

□ Enable Line Series

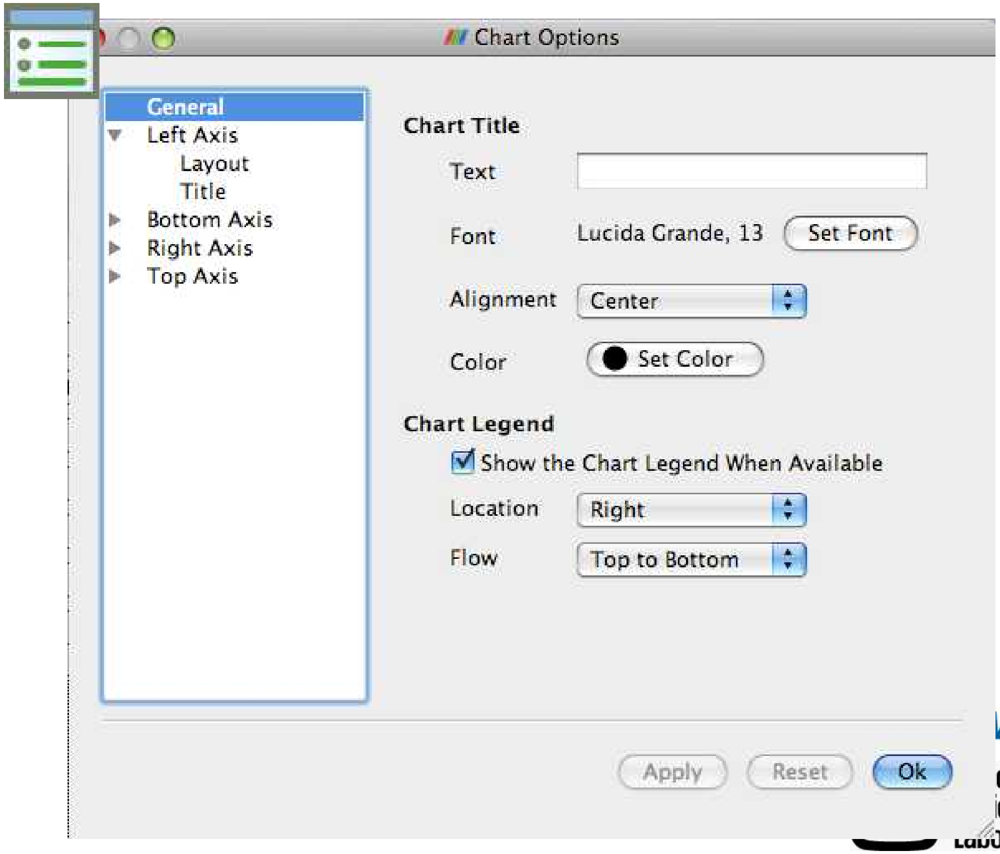


Chart Title

Text:

Font: Lucida Grande, 13

Alignment: Center

Color: ● Set Color

Chart Legend

☑ Show the Chart Legend When Available

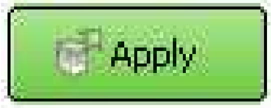
Location: Right

Flow: Top to Bottom

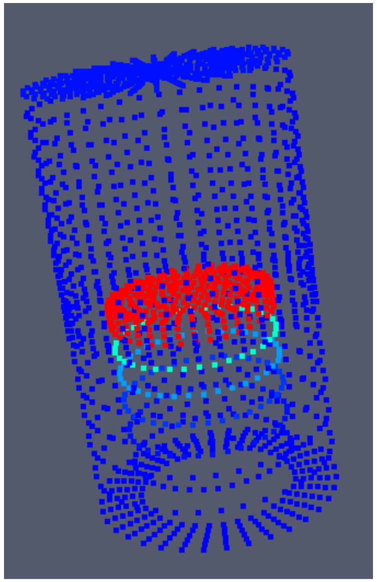
Adjusting Plots

1. Place plot with view split, delete, resize, and swap.
2. In Display tab, turn off all variables except Temp and Pres.
3. Select Pres in the Display tab.
4. Change Chart Axis to Bottom – Right.

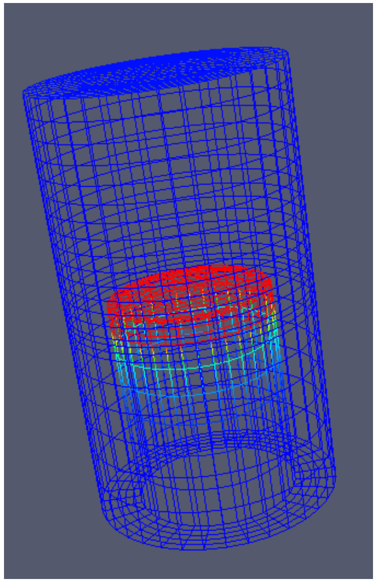
Histogram / Bar Chart

1. Select disk_out_ref.ex2.
2. Filters → Data Analysis → Histogram
3. Change scalars to Temp.
4. 

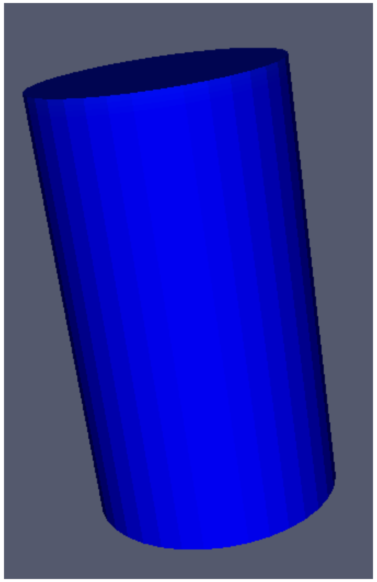
Geometry Representations



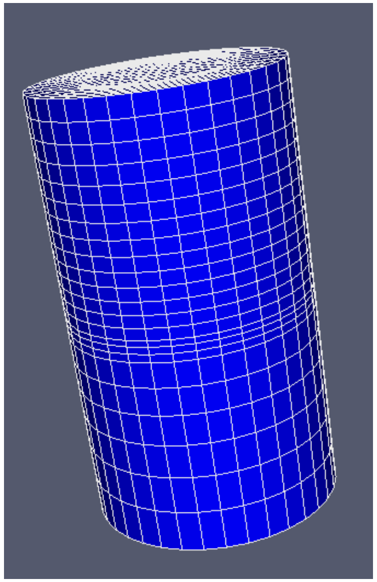
Points



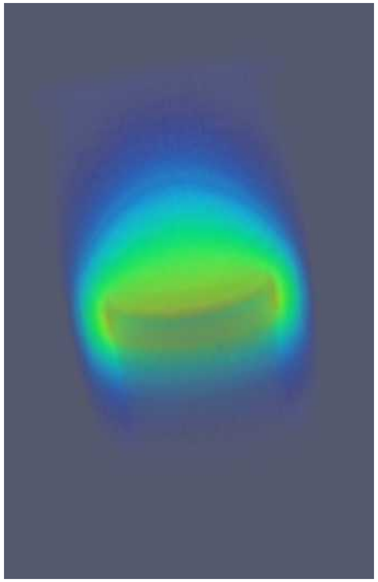
Wireframe



Surface



Surface with Edges



Volume

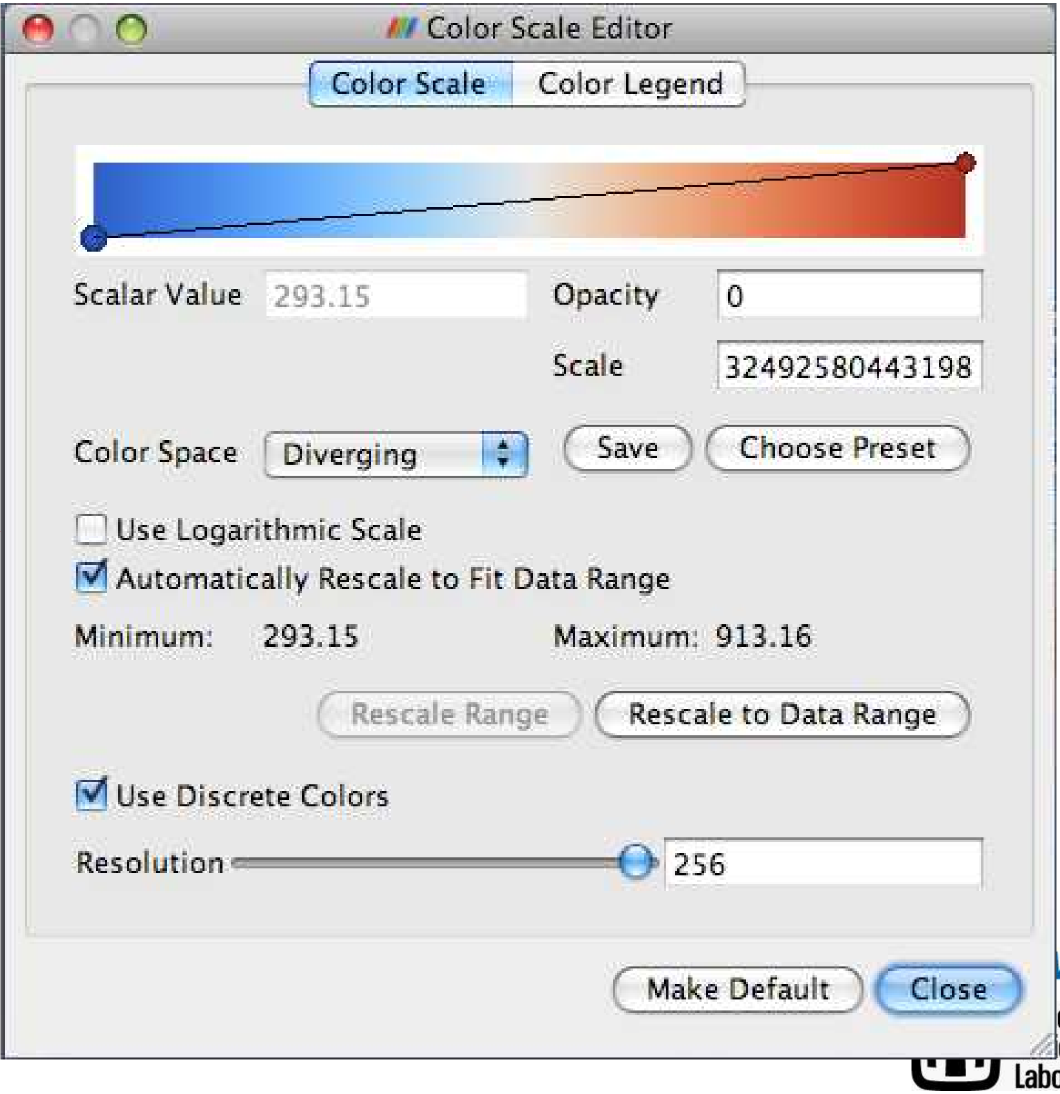
Volume Rendering

1. Select view with temp on clipped mesh.
2. Delete visible clip filter.
3. Make sure disk_out_ref.ex2 selected.
4. Change variable viewed to Temp.
5. Change representation to Volume.

Volume Rendering + Surface Geometry

1. Select view showing streamlines.
2. Make disk_out_ref.ex2 visible. 👁
3. Change variable viewed to Temp.
4. Change representation to Volume.

Transfer Function Editor



Color Scale Editor

Color Scale | Color Legend

Scalar Value 293.15

Opacity 0

Scale 32492580443198

Color Space: Diverging

Save | Choose Preset

☐ Use Logarithmic Scale

☑ Automatically Rescale to Fit Data Range

Minimum: 293.15

Maximum: 913.16

Rescale Range | Rescale to Data Range

☑ Use Discrete Colors

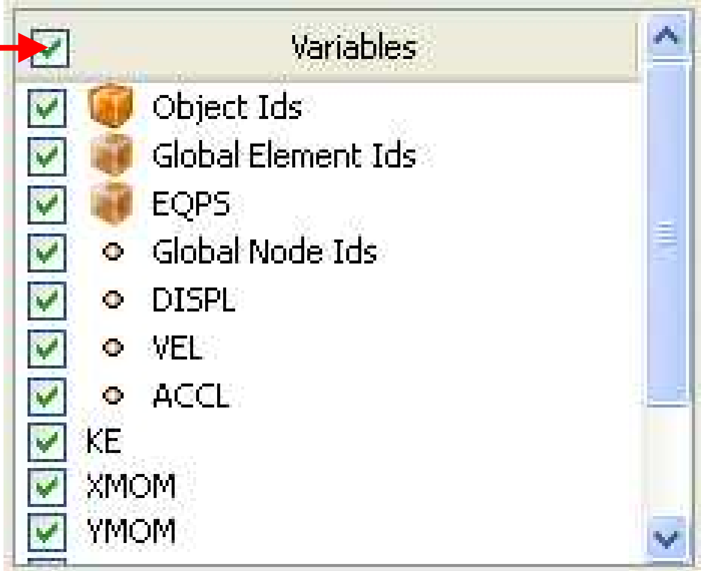
Resolution 256

Make Default | Close

Reset ParaView

Loading Data with Time

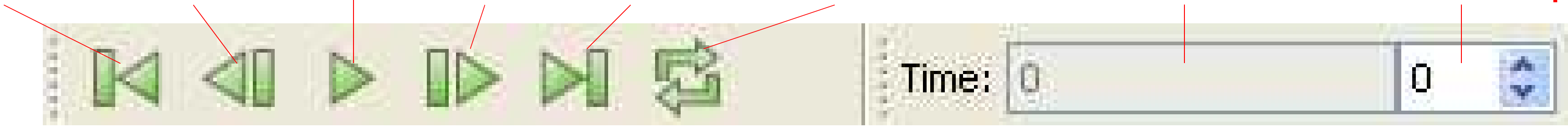
1. Open the file can.ex2.
2. Select all variables.
3. Apply
4. +Y
5. ▶



☑ Variables

- ☑ Object Ids
- ☑ Global Element Ids
- ☑ EQPS
- ☑ Global Node Ids
- ☑ DISPL
- ☑ VEL
- ☑ ACCL
- ☑ KE
- ☑ XMOM
- ☑ YMOM

Animation Toolbar



First Frame

Previous Frame

Play

Next Frame

Last Frame

Loop Animation

Current Time

Current Time Step

Time: 0

0

Animation Pitfall

1. Go to first time step. ⏮
2. Turn on EQPS variable.
3. Turn on color legend. ▮
4. Play ▶ (or skip to last time step ⏭).

Animation Pitfall

1. Go to first time step. ⏮
2. Turn on EQPS variable.
3. Turn on color legend. ▮
4. Play ▶ (or skip to last time step ⏭).
5. Fix with Rescale to Data Range. ↔

Selection

Surface Cell Selection

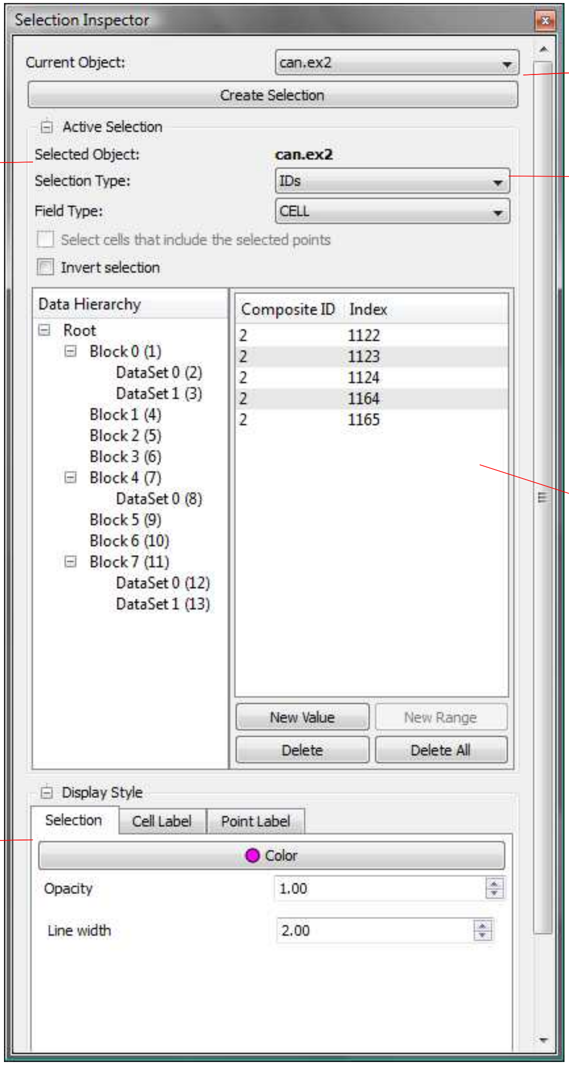
Surface Point Selection

Through Cell Selection

Through Point Selection

Block Selection

Selection Inspector



Current Object: can.ex2

Selected Object: **can.ex2**

Selection Type: IDs

Field Type: CELL

□ Select cells that include the selected points

□ Invert selection

Composite ID	Index
2	1122
2	1123
2	1124
2	1164
2	1165

Opacity 1.00

Line width 2.00

Create new selection

Active selection properties

Selection type

Selected cells ids

Selection display Properties/Labeling

Selections

1. Open the Selection Inspector (View → Selection Inspector).
2. Make various rubber-band selections.
3. Observe results in Selection Inspector.
4. Play with the Invert Selection and Show Frustum options.

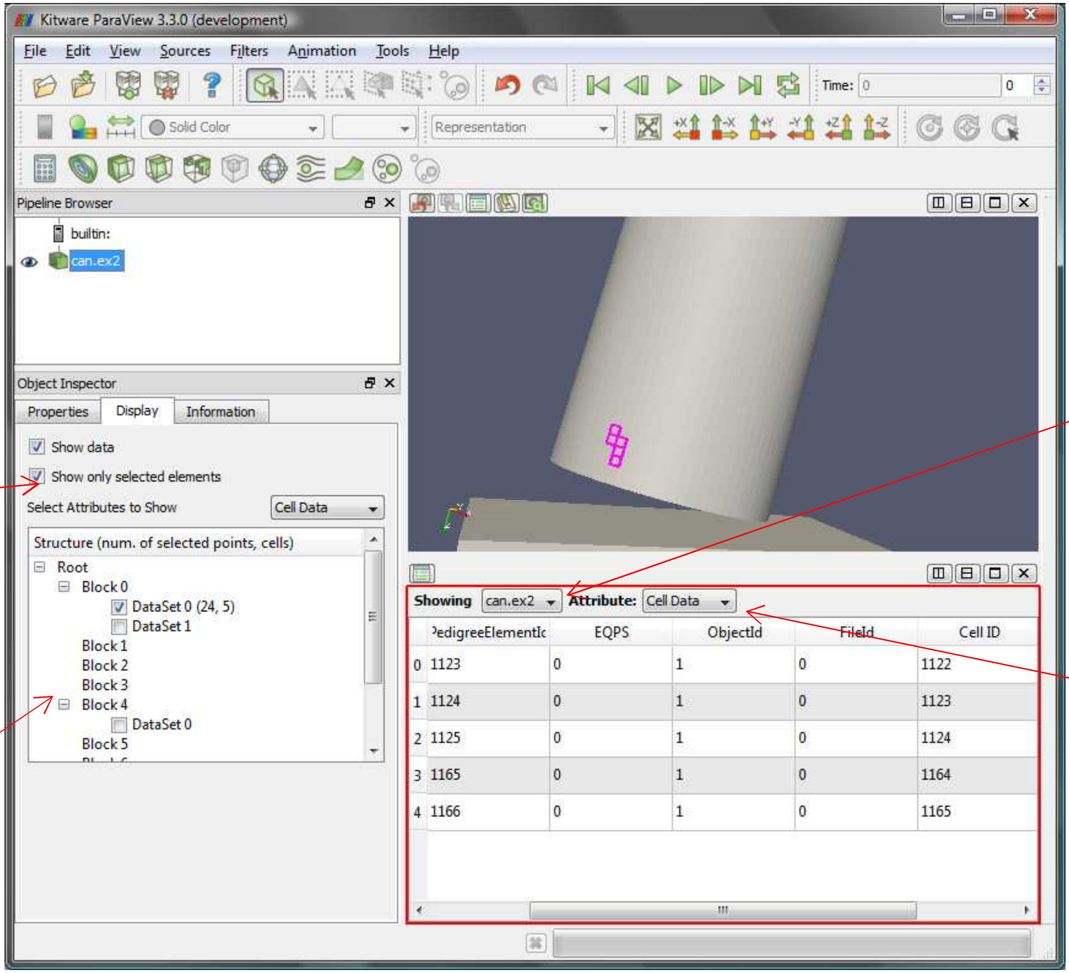
Frustum vs. Id Selections

1. Make a Select Cells Through
2. Turn on Show Frustum in Selection Inspector. Rotate 3D view.
3. Play ▷
4. Change the Selection Type to IDs.
5. Play ▷

Spreadsheet View

1. Split the view (◫ or ⊟).
2. In new view, click Spreadsheet View.

Spreadsheet View



☑ Show data

☑ Show only selected elements

Select Attributes to Show: Cell Data

Structure (num. of selected points, cells)

- Root
  - Block 0
    - ☑ DataSet 0 (24, 5)
    - □ DataSet 1
  - Block 1
  - Block 2
  - Block 3
  - Block 4
    - □ DataSet 0
  - Block 5

Showing: can.ex2   Attribute: Cell Data

	edigreeElementIc	EQPS	ObjectId	FileId	Cell ID
0	1123	0	1	0	1122
1	1124	0	1	0	1123
2	1125	0	1	0	1124
3	1165	0	1	0	1164
4	1166	0	1	0	1165

Show only selected items

Select block to inspect

What is shown in the view

Attribute shown

Spreadsheet View

1. Split the view (◫ or ⊟).
2. In new view, click Spreadsheet View.
3. For Attribute, select Cell Data.
4. Find selected rows in spreadsheet.
5. In Display panel, turn on Show only selected elements.

Selecting in Spreadsheet View

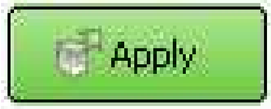
1. Uncheck Show only selected elements.
2. Select a few rows in the spreadsheet.
3. Find selection in 3D view.
4. Click Cell Label tab in Selection Inspector.
5. Check Visible.
6. Change Label Mode to EQPS.

Plot Selection Over Time

1. Filters → Data Analysis → Plot Selection Over Time
2. Click Copy Active Selection in Object Inspector.
3.

4. In Display panel, select different blocks to plot.

Extracting a Selection

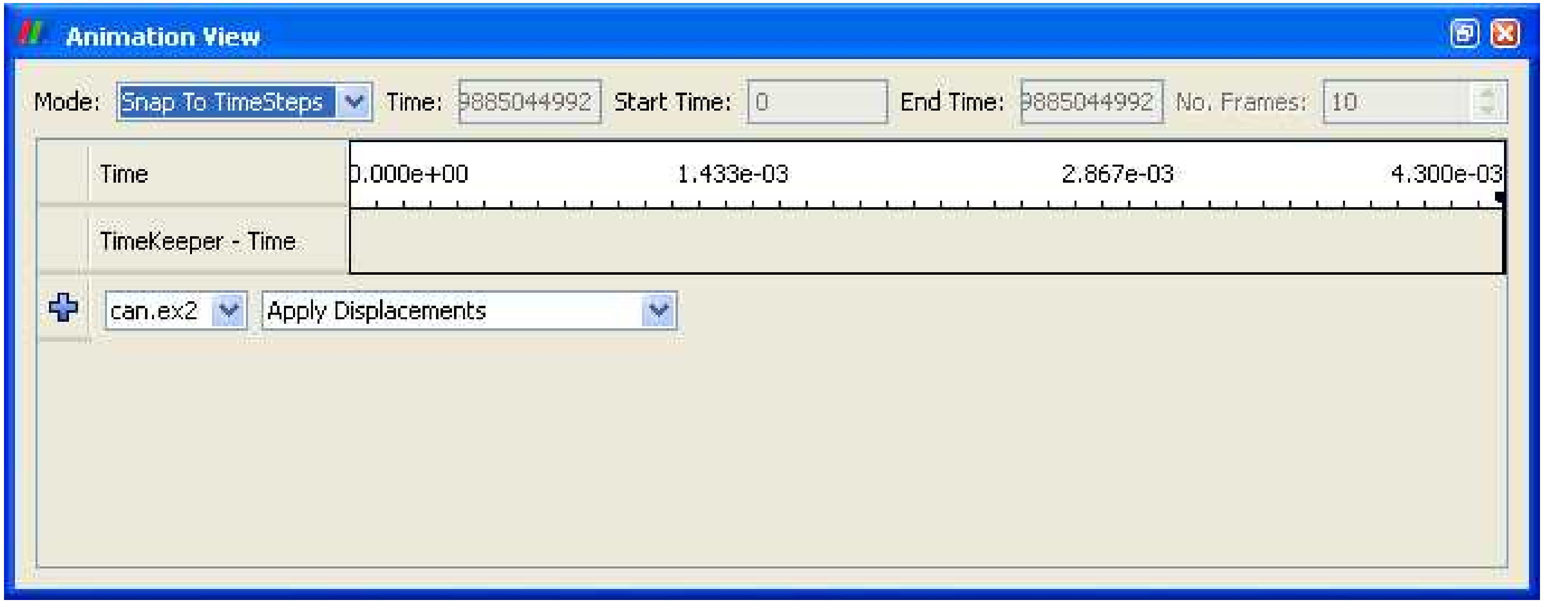
1. Turn off cell labels.
2. Perform a sizeable selection.
3. Filters → Data Analysis → Extract Selection
4. Click Copy Active Selection.
5. 

Cleanup

1. Close the Selection Inspector.
2. Delete the Plot and Spreadsheet views.
3. Delete the PlotSelectionOverTime1 and ExtractSelection1.

Animation View

View → Animation View



Animation View

Mode: Snap To TimeSteps

Time: 9885044992

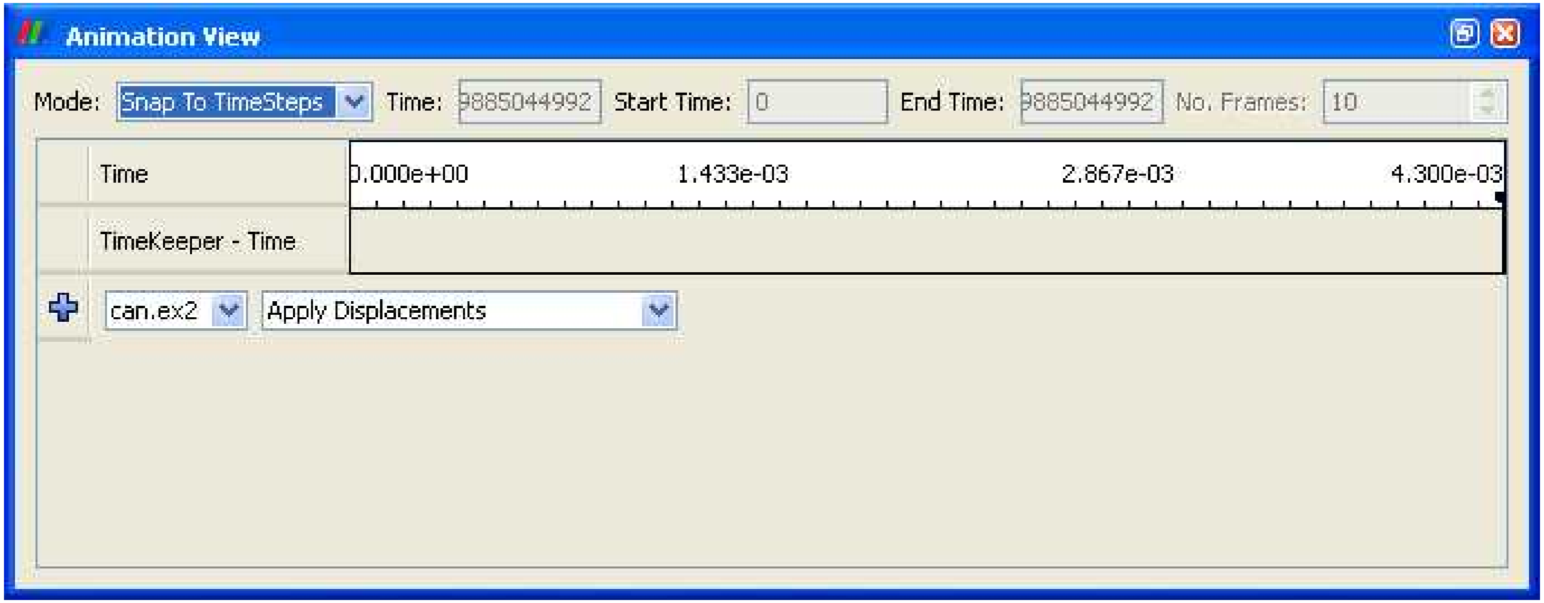
Start Time: 0

End Time: 9885044992

No. Frames: 10

Animation View

View → Animation View



Animation View

Mode: Snap To TimeSteps

Time: 9885044992

Start Time: 0

End Time: 9885044992

No. Frames: 10

Time

TimeKeeper - Time

can.ex2

Apply Displacements

Animation Modes: Sequence, Real Time, and Snap To TimeSteps

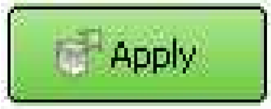
Changing Animation Timing

1. Change animation mode to Real Time.
    - Default animation time is 10 sec.
2. ▶

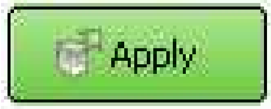
Changing Animation Timing

1. Change animation mode to Real Time.
   - Default animation time is 10 sec.
2. ▷
3. Change animation time to 60 sec.
4. ▷ again.

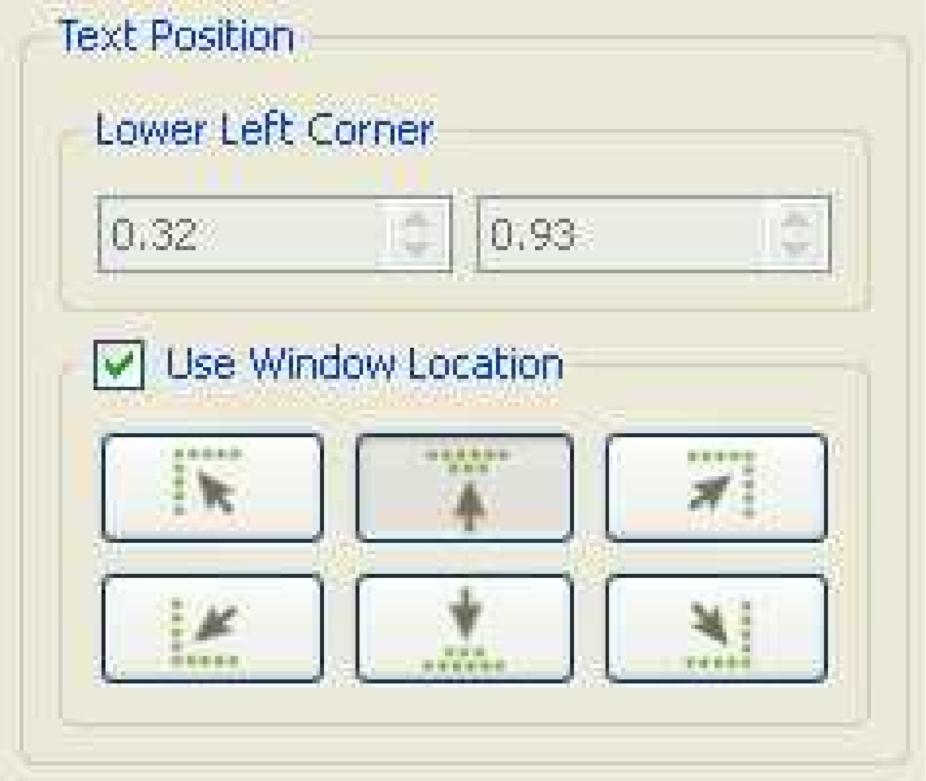
Smoothing the Animation

1. Filters → Alphabetical → Temporal Interpolator
2. Change mode back to Real Time.
3. Split view. Show can.ex2 in one and TemporalInterpolator1 in the other. Link the cameras.
4. 

Adding Text Annotation

1. Sources → Text
2. Type a message in text edit box
3. 

Text Position



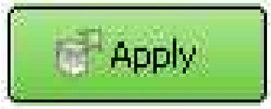
Text Position

Lower Left Corner

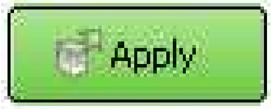
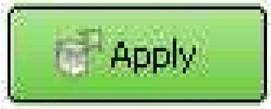
0.32 0.93

☑ Use Window Location

Annotate Time

1. Sources → Annotate Time
2. 

Annotate Time

1. Sources → Annotate Time
2. 
   Apply
3. Select can.ex2
4. Filters → Alphabetical → Annotate Time
5. 
   Apply
6. Move annotation around.

Reset ParaView

Make an Animation

1. Sources → Sphere, Apply
2. Make animation view visible.
3. Change No. Frames to 50.
4. Select Sphere1, Start Theta, press ✚

Make an Animation

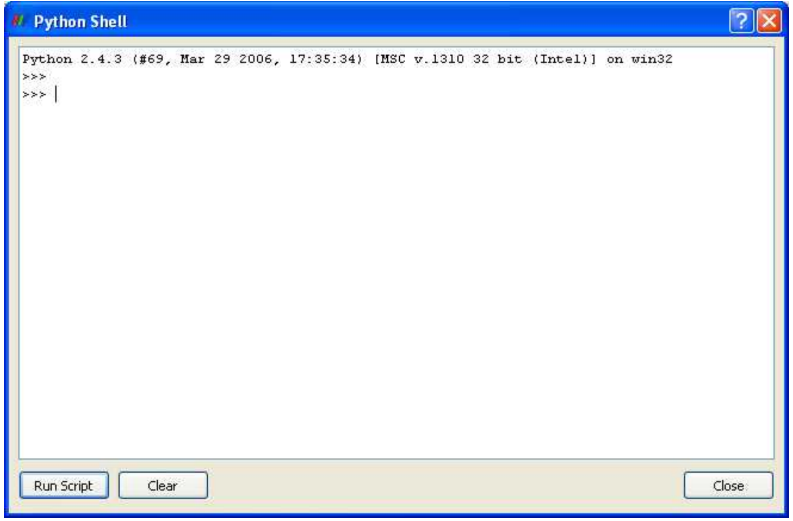
1. Sources → Sphere, Apply
2. Make animation view visible.
3. Change No. Frames to 50.
4. Select Sphere1, Start Theta, press ✚
5. Double-click Sphere1 – Start Theta
6. Make New keyframe.
7. First keyframe→360, second keyframe→0.
8. Click OK.

Animating Two Properties

1. Open Sphere1 – Start Theta.
2. Delete the first keyframe (at time 0).
3. Click OK.
4. Create Sphere1 – End Theta.
5. Open Sphere1 – End Theta.
6. Change second key frame time to 0.5.

Scripting Options

- Tools → Python Shell



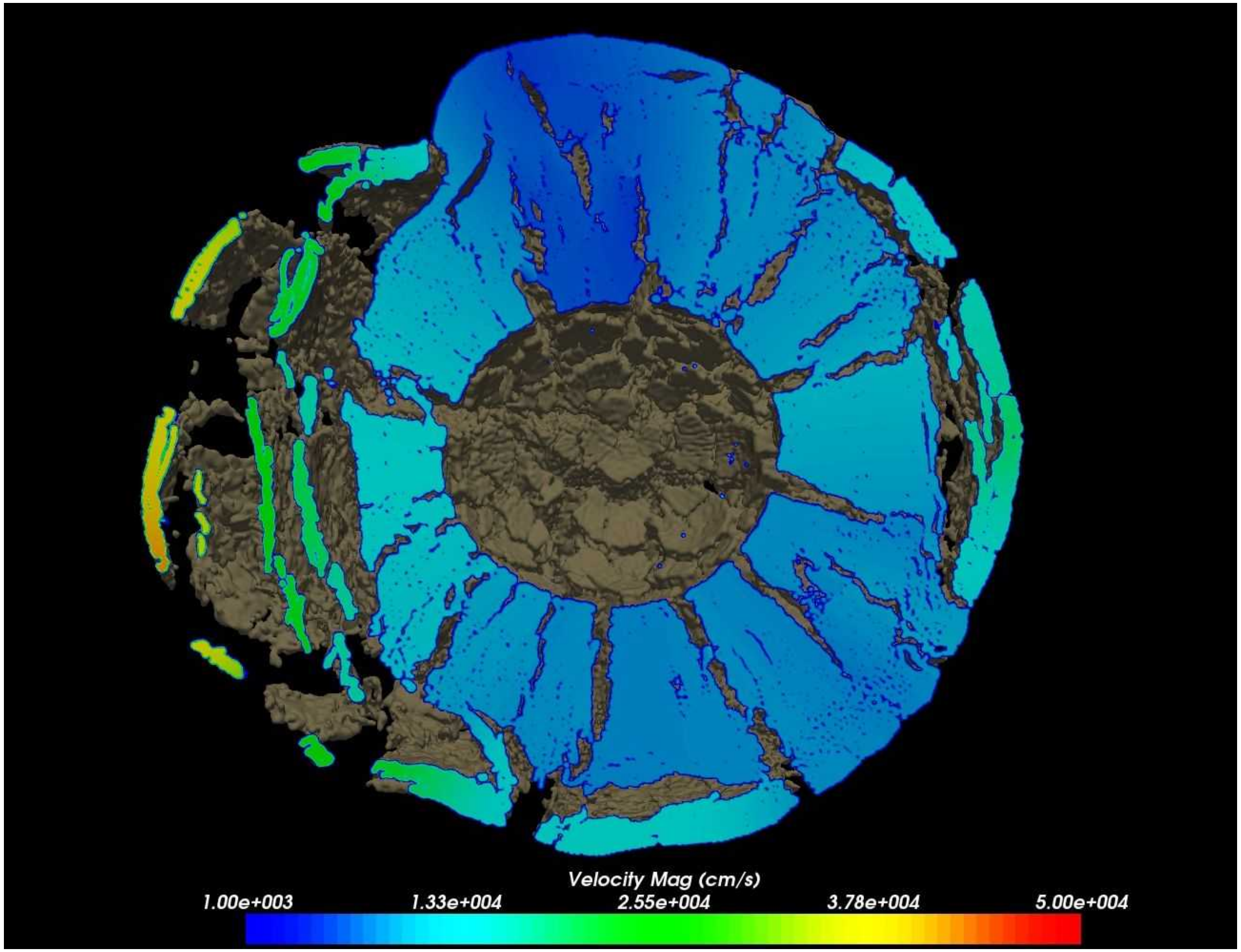
```
Python 2.4.3 (#69, Mar 29 2006, 17:35:34) [MSC v.1310 32 bit (Intel)] on win32
>>>
>>> |
```

- Filters → Data Analysis → Programmable Filter
- pvpython, pvbatch

Visualizing Large Models

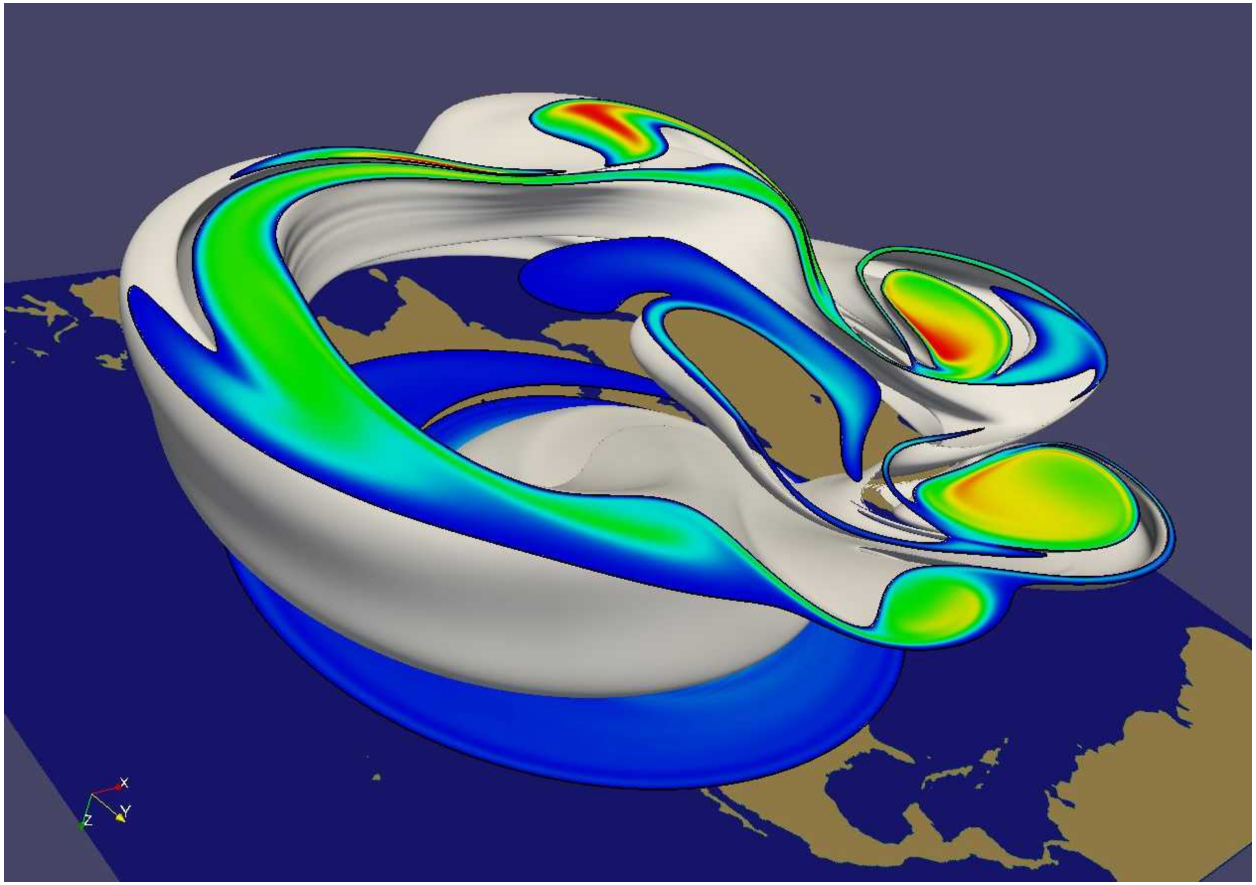
Golevka Asteroid vs. 10 Megaton Explosion

- CTH shock physics, over 1 billion cells



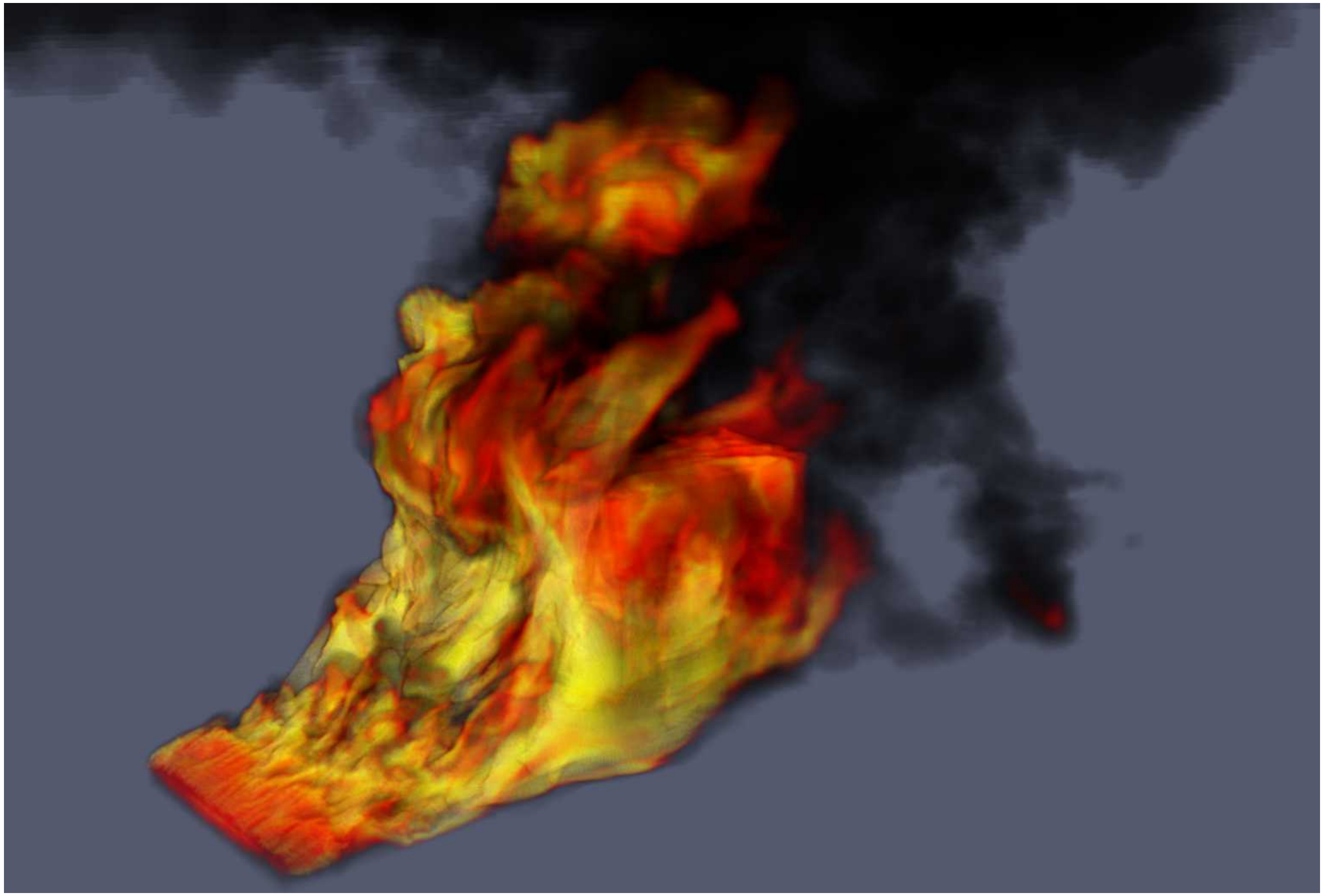
Polar Vortex Breakdown

- SEAM Climate Modeling, 1 billion cells (500 million cells visualized).

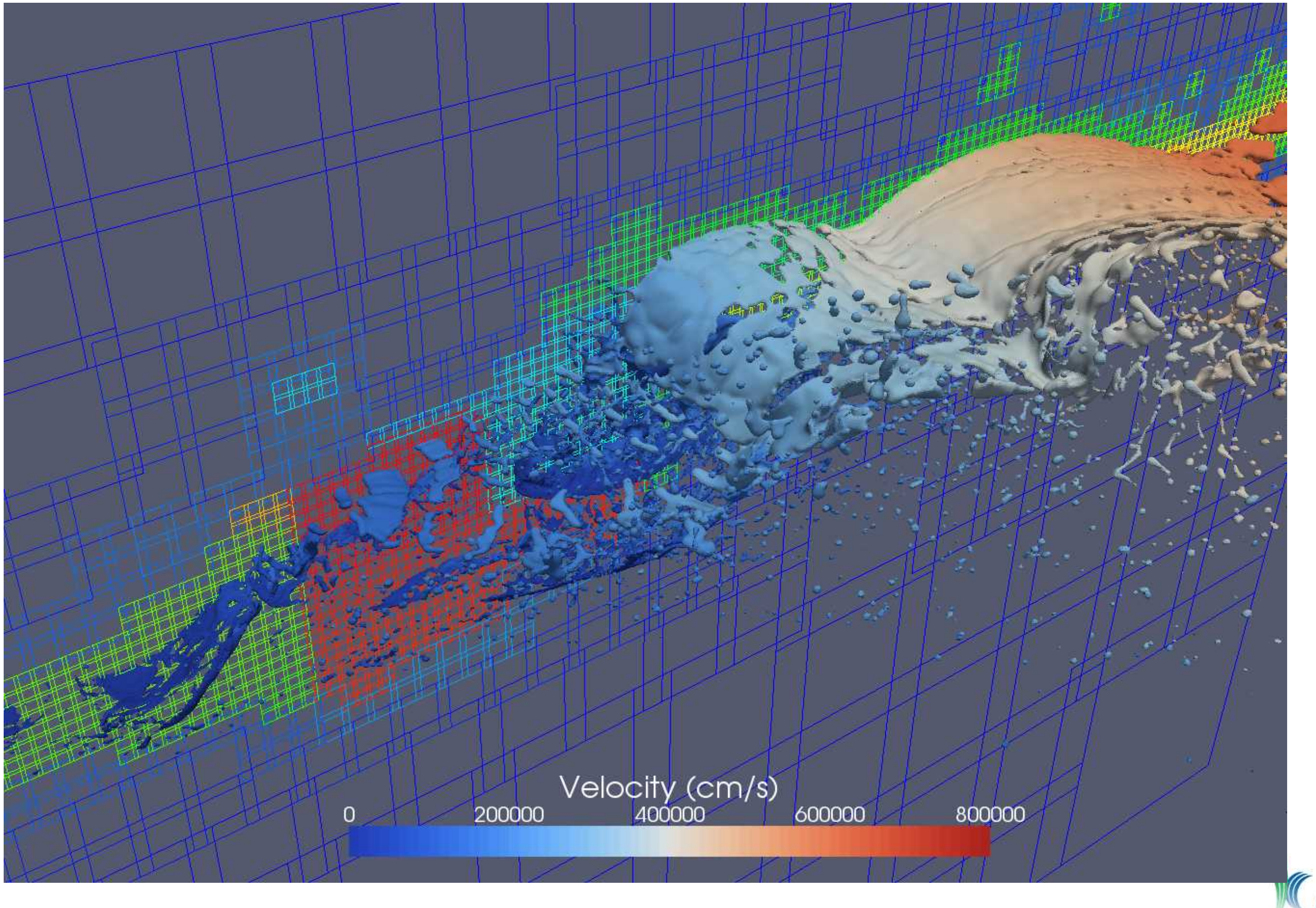


Objects-in-Crosswind Fire

- Coupled SIERRA/Fuego/Syrinx/Calore, 10 million unstructured hexahedra



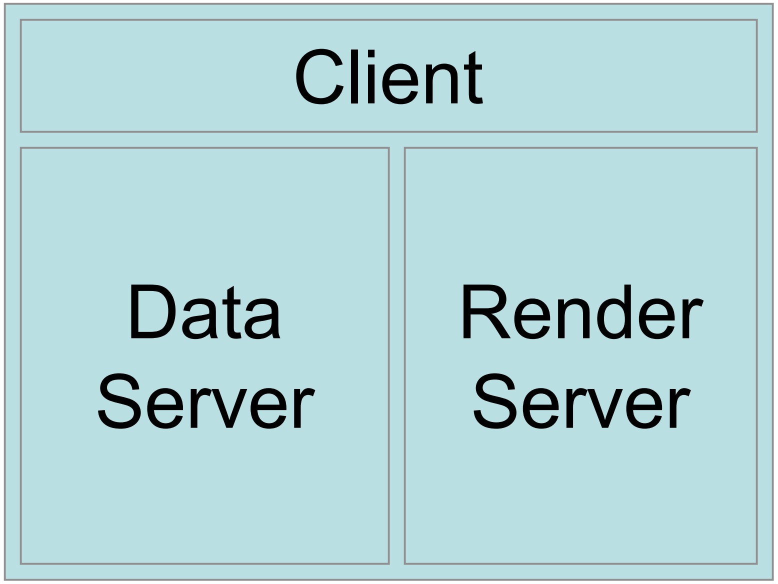
Large Scale AMR



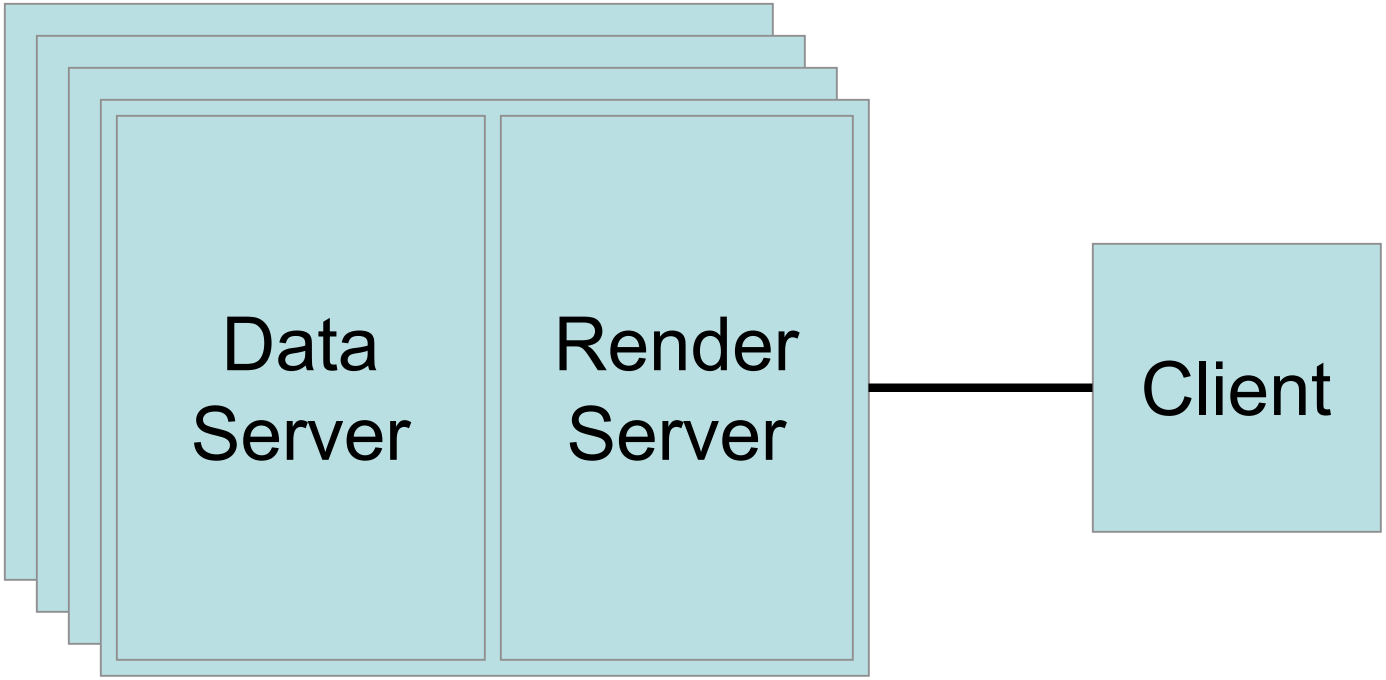
ParaView Architecture

- Three tier
  - –Data Server
  - –Render Server
  - –Client

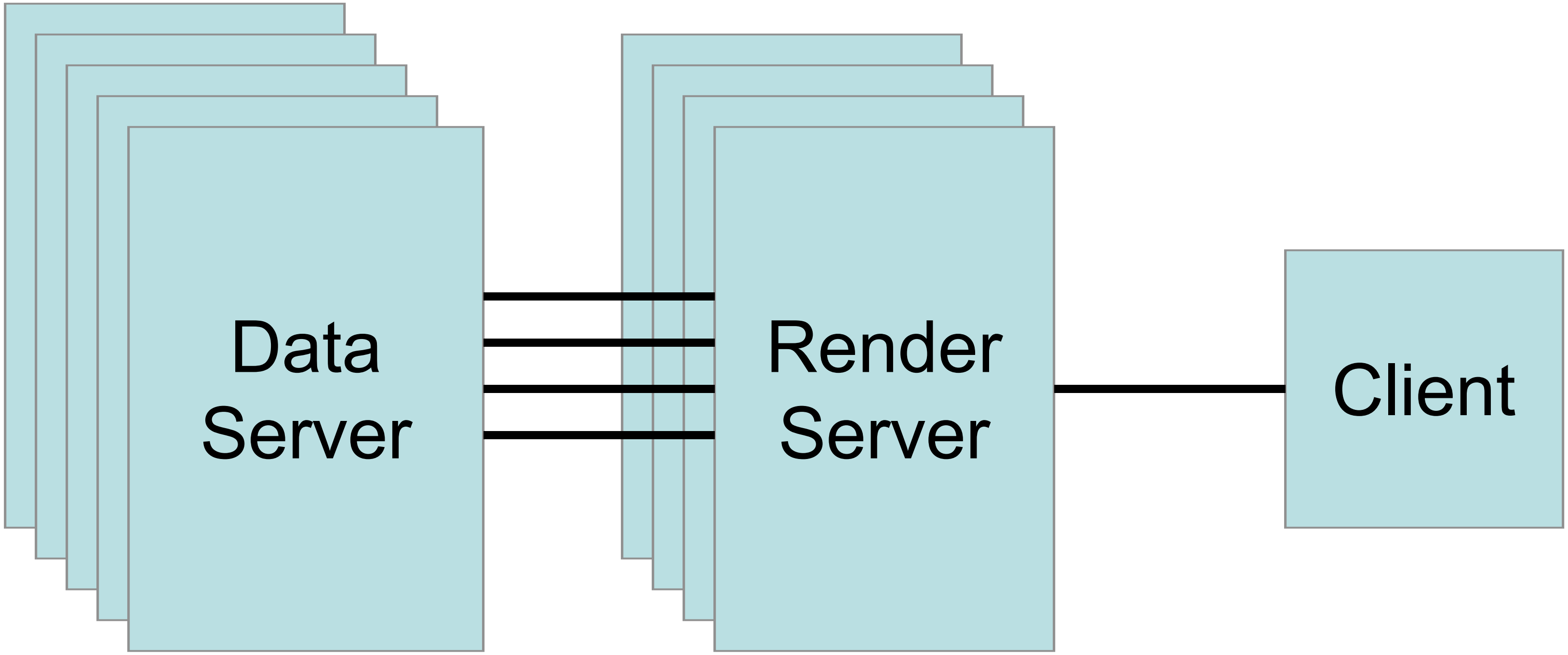
Standalone



Client-Server



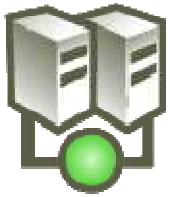
Client-Render Server-Data Server

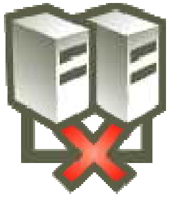


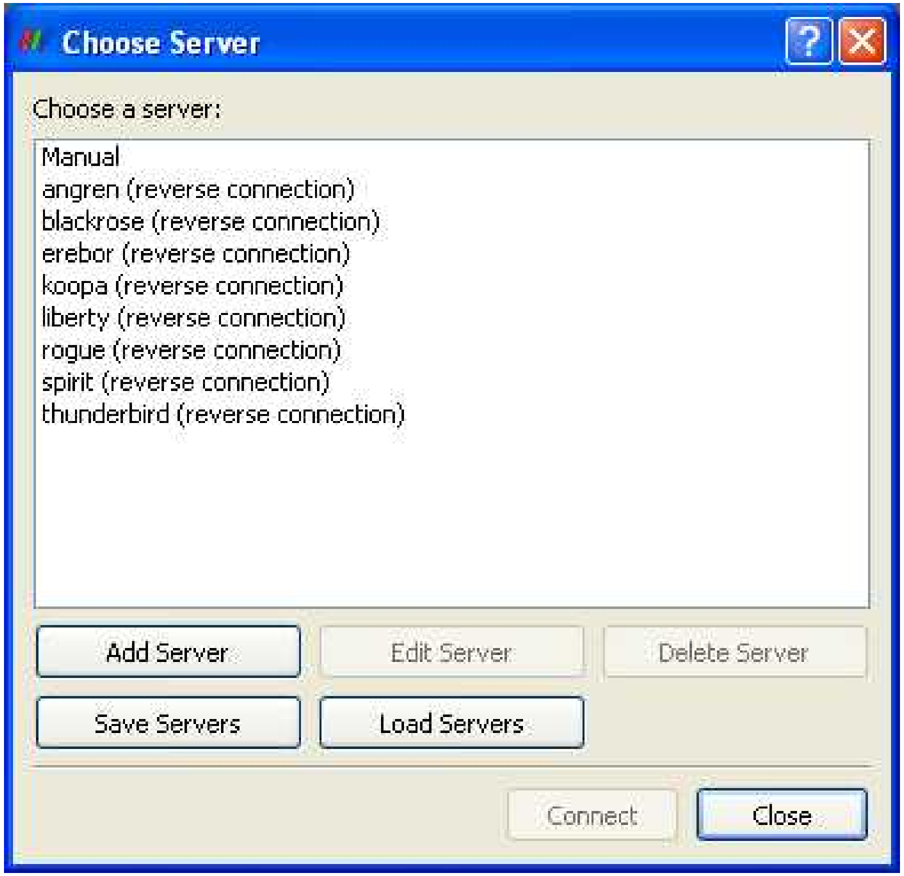
Requirements for Installing ParaView Server

- C++
- CMake (www.cmake.org)
- MPI
- OpenGL (or Mesa3D www.mesa3d.org)
- Qt 4.3 (optional)
- Python (optional)
- http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server#Compiling

Connecting to a ParaView Server







Choose Server

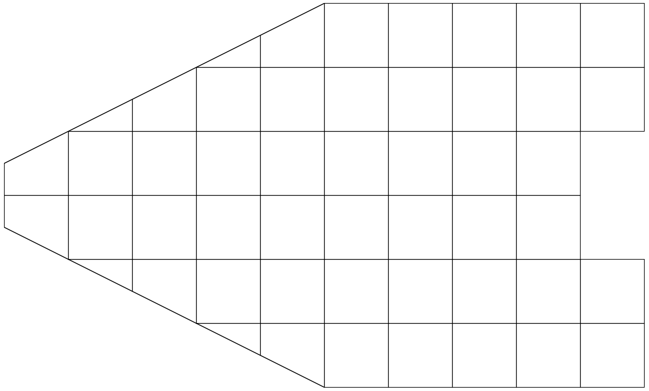
Choose a server:

Manual
angren (reverse connection)
blackrose (reverse connection)
erebor (reverse connection)
koopa (reverse connection)
liberty (reverse connection)
rogue (reverse connection)
spirit (reverse connection)
thunderbird (reverse connection)

http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server#Running_the_Server

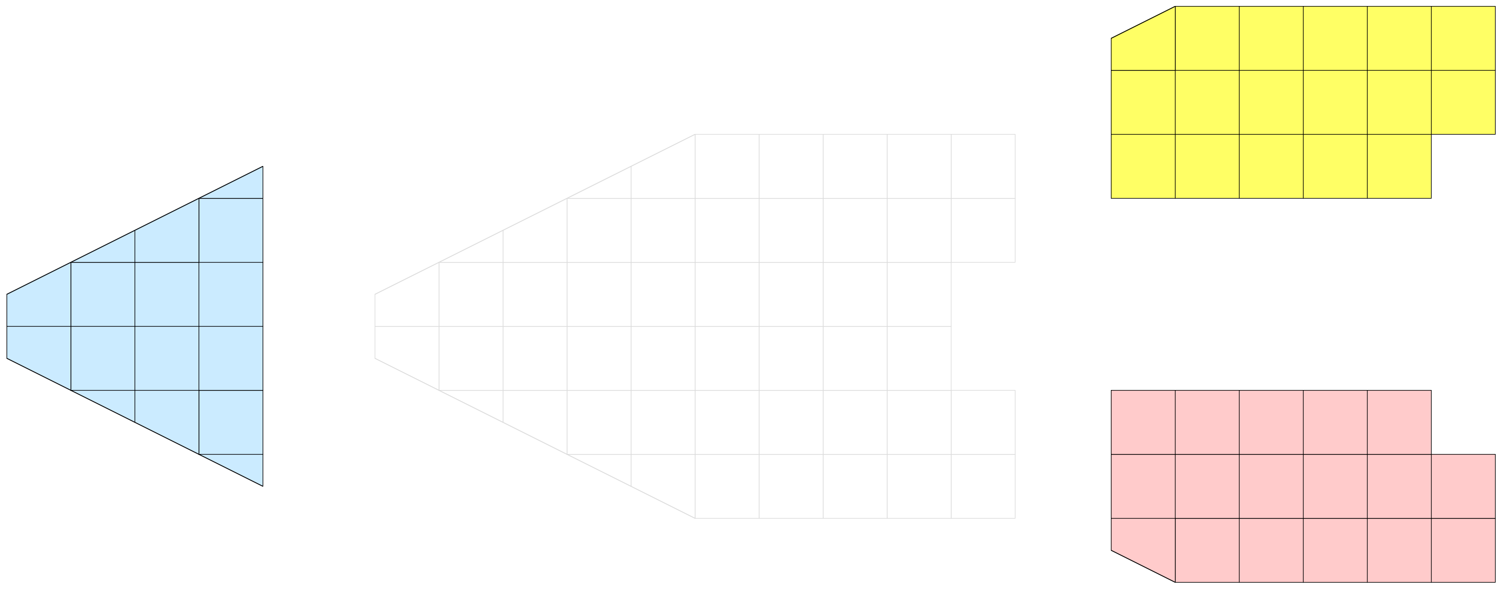
Data Parallel Pipelines

- Duplicate pipelines run independently on different partitions of data.



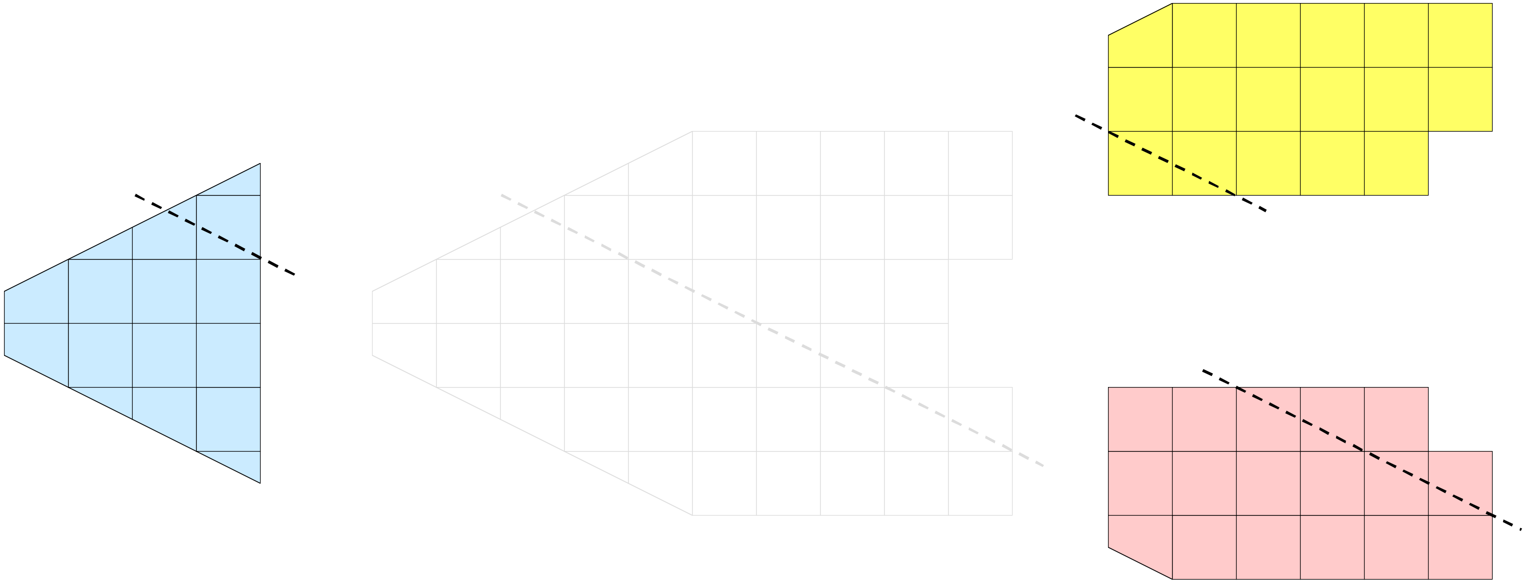
Data Parallel Pipelines

- Duplicate pipelines run independently on different partitions of data.



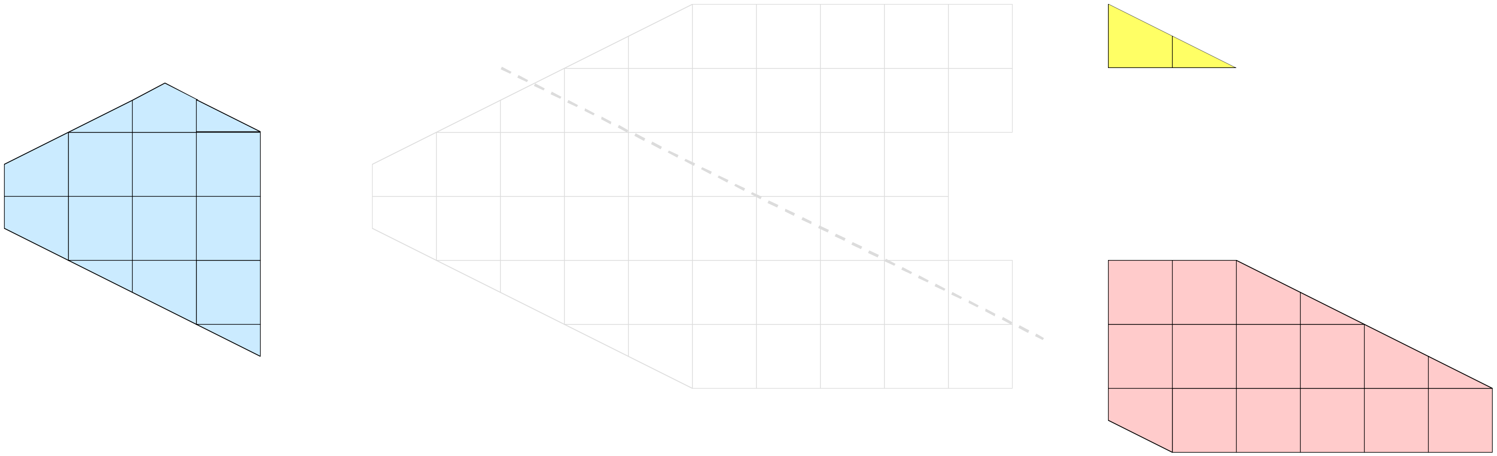
Data Parallel Pipelines

- Some operations will work regardless.
  - Example: Clipping.



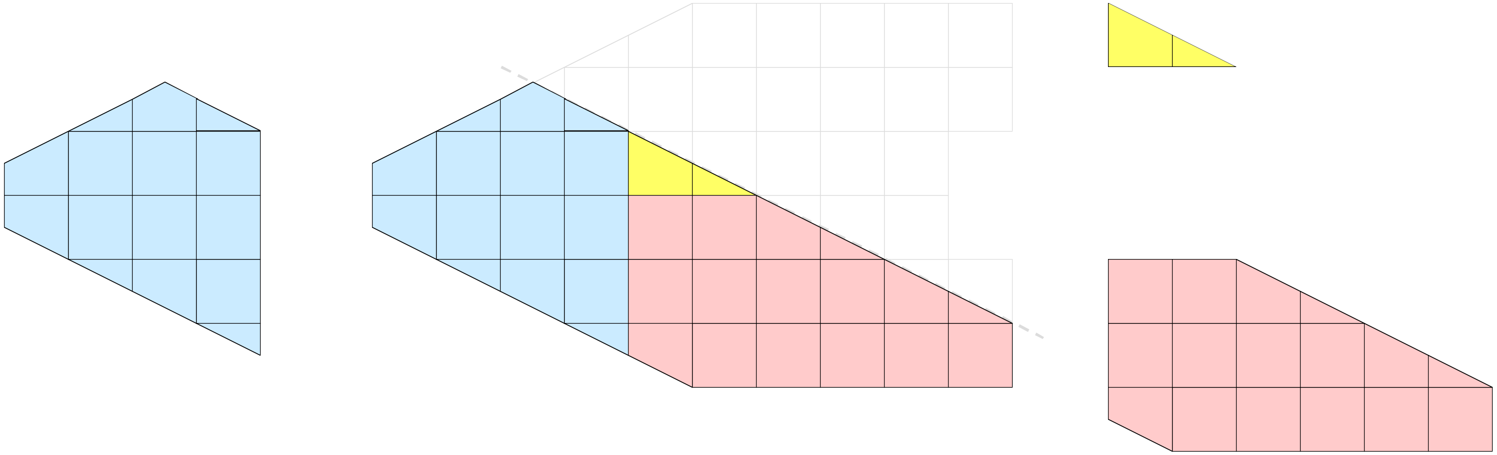
Data Parallel Pipelines

- Some operations will work regardless.
  - –Example: Clipping.



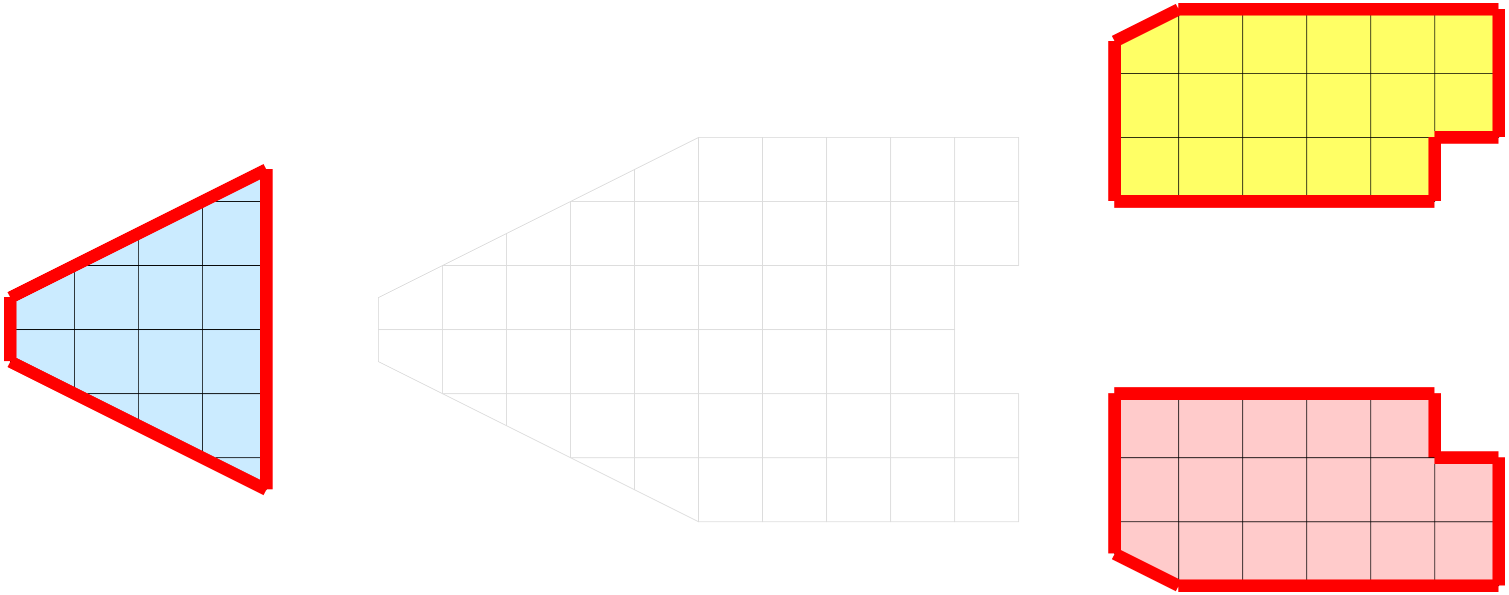
Data Parallel Pipelines

- Some operations will work regardless.
  - –Example: Clipping.



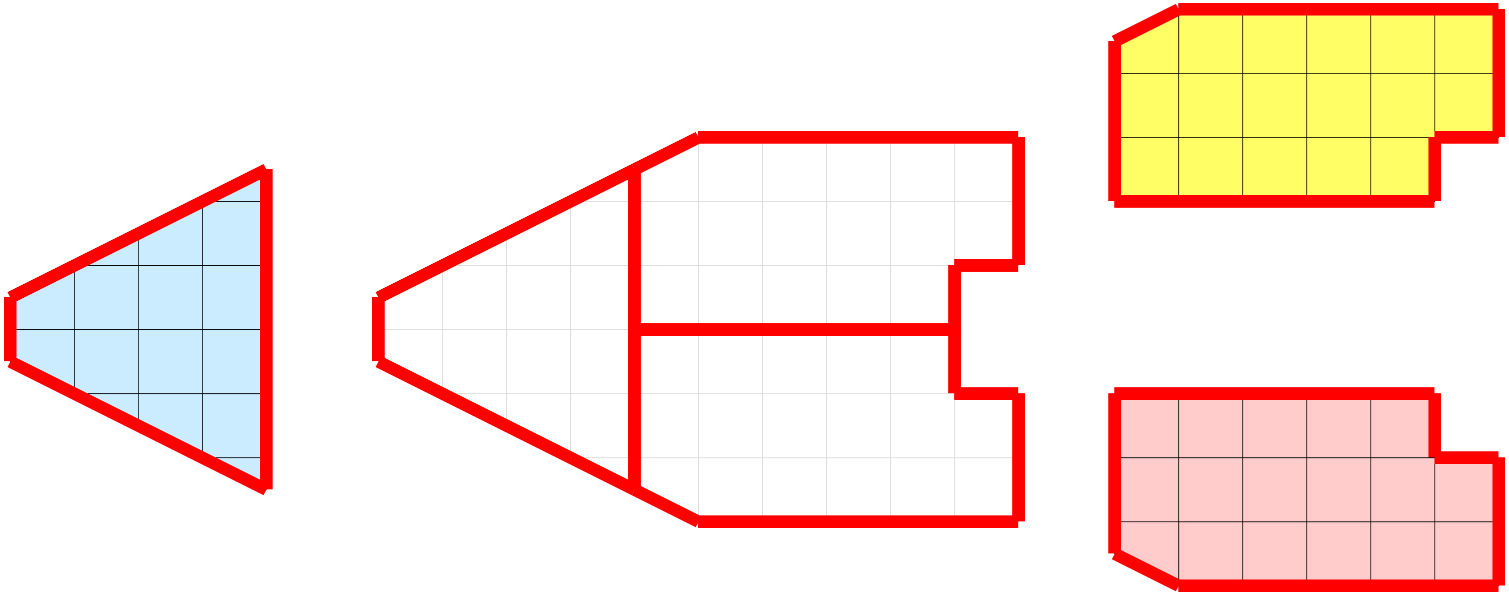
Data Parallel Pipelines

- Some operations will have problems.
  - –Example: External Faces



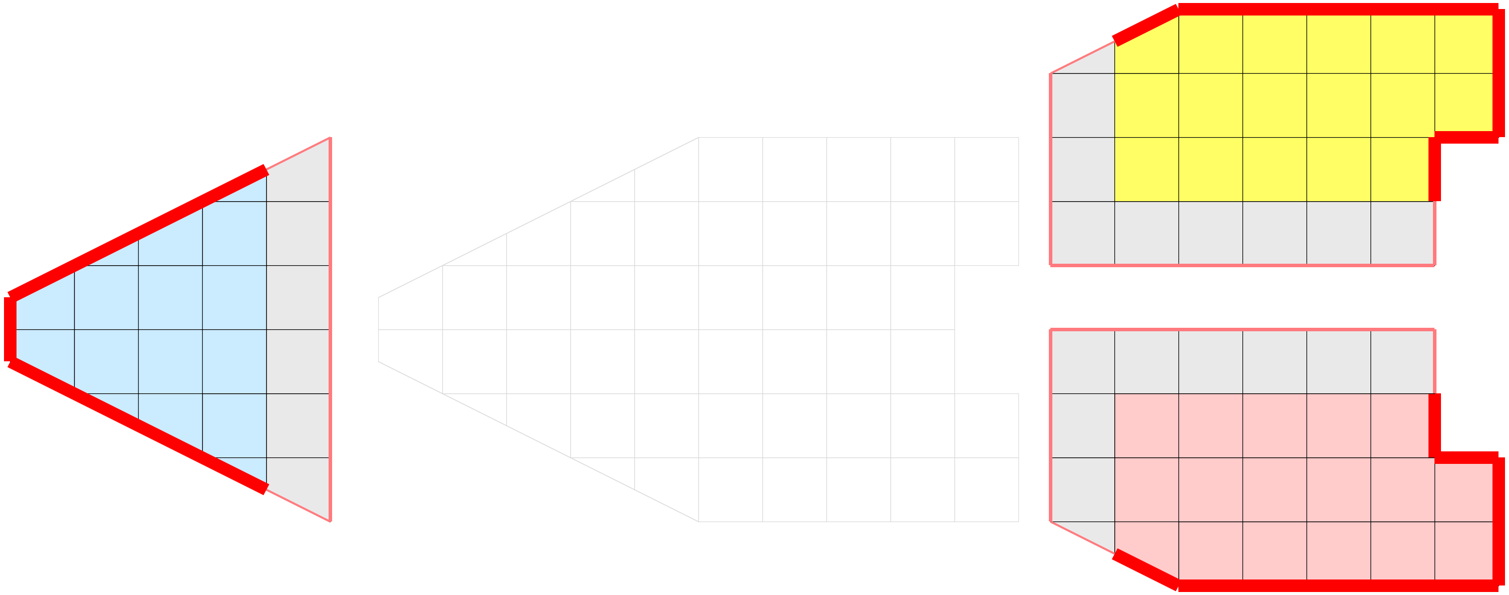
Data Parallel Pipelines

- Some operations will have problems.
  - Example: External Faces



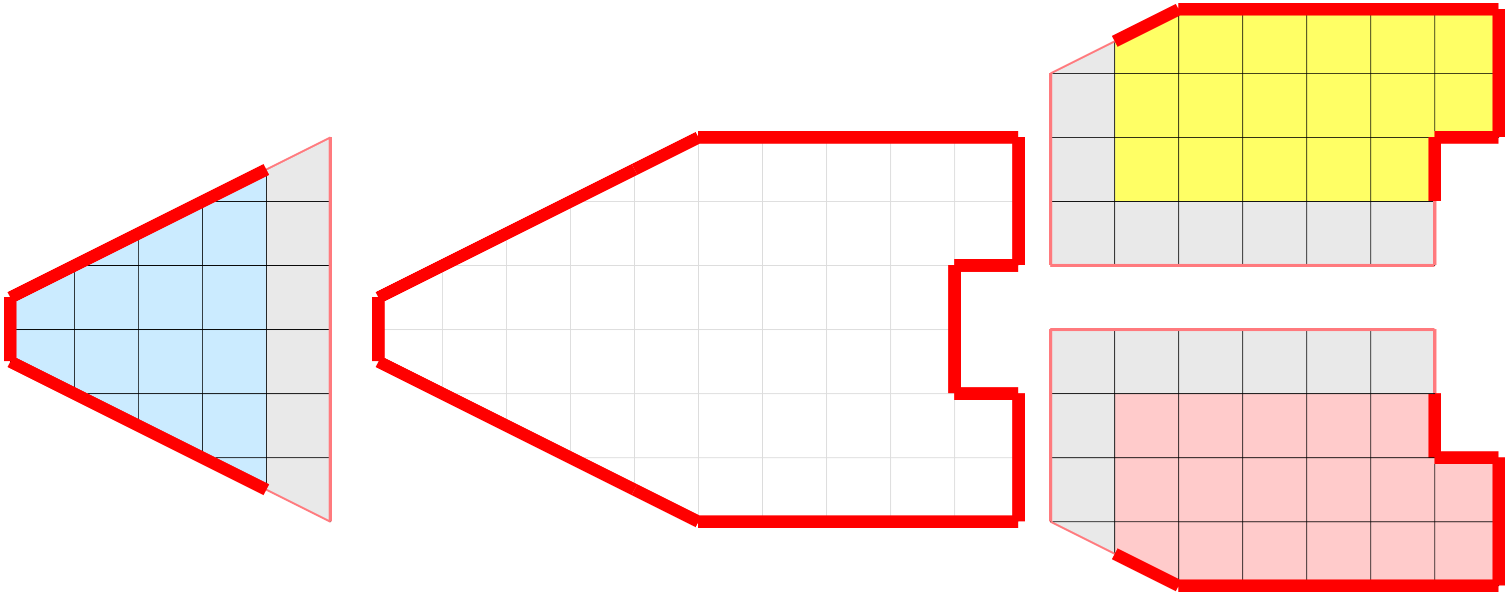
Data Parallel Pipelines

- Ghost cells can solve most of these problems.



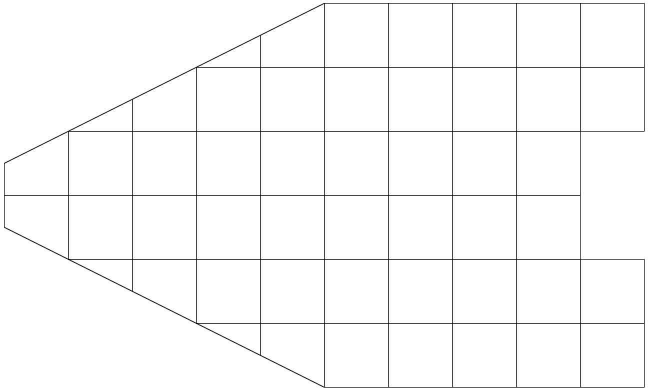
Data Parallel Pipelines

- Ghost cells can solve most of these problems.



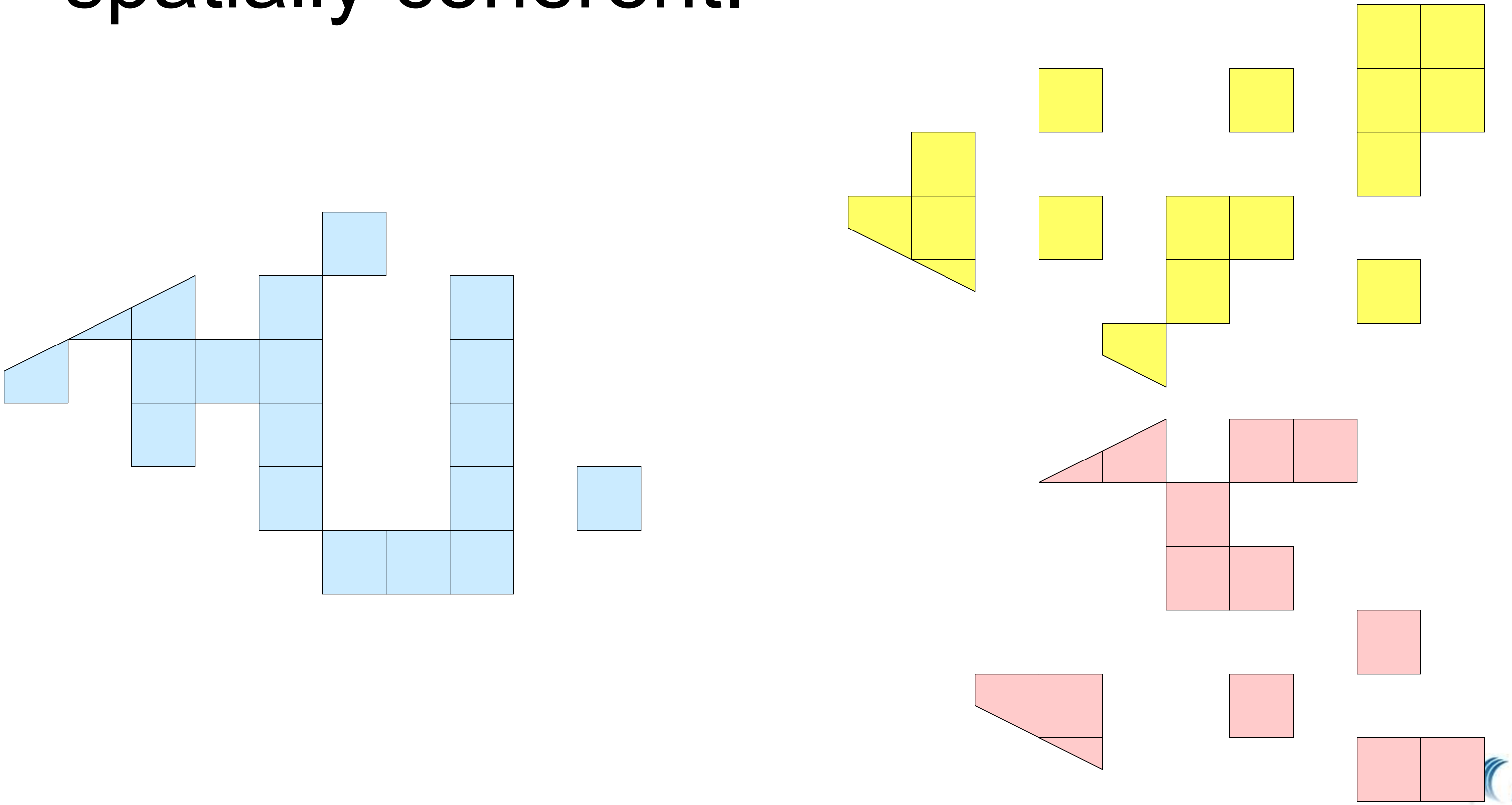
Data Partitioning

- Partitions should be load balanced and spatially coherent.



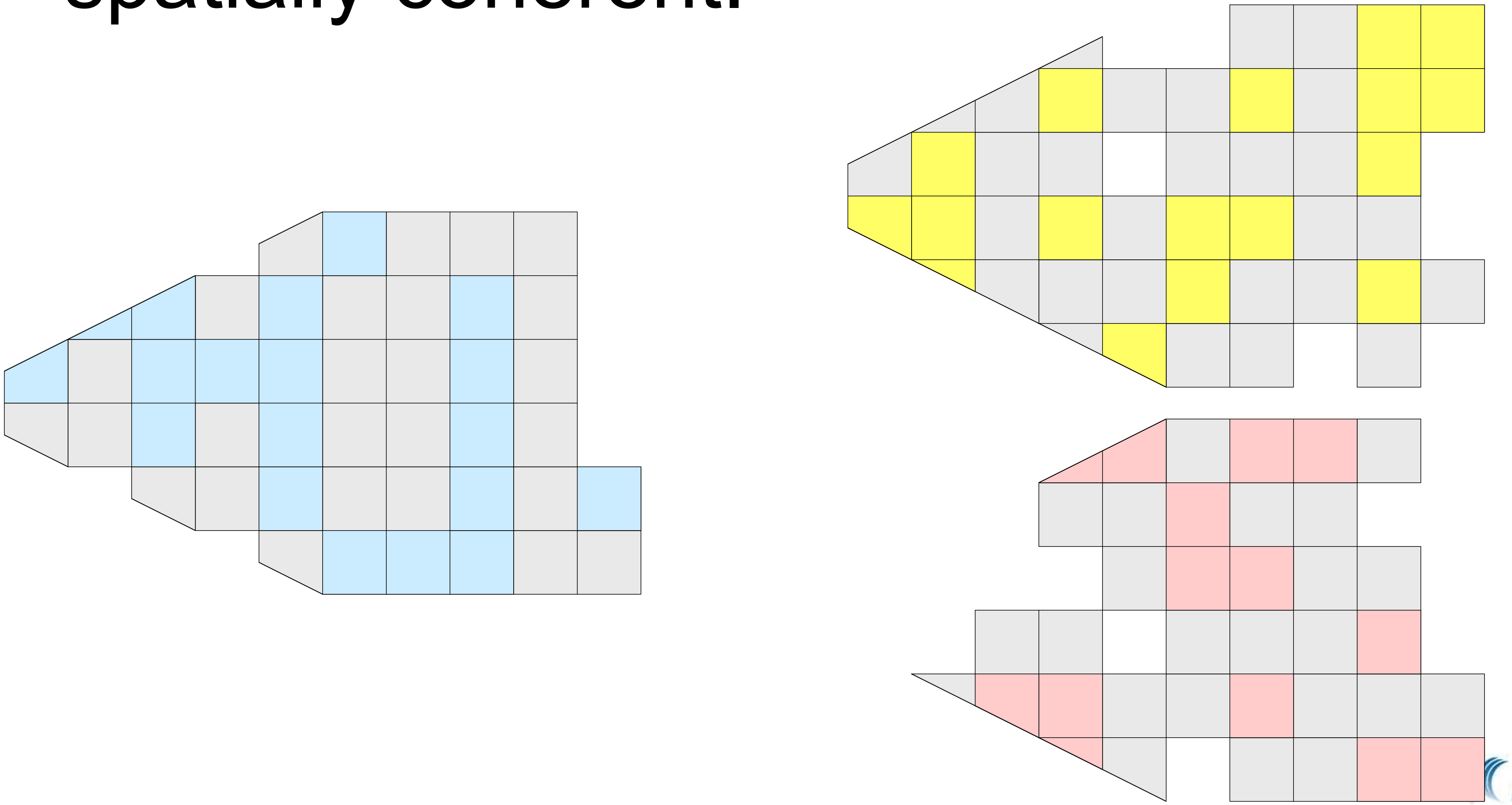
Data Partitioning

- Partitions should be load balanced and spatially coherent.



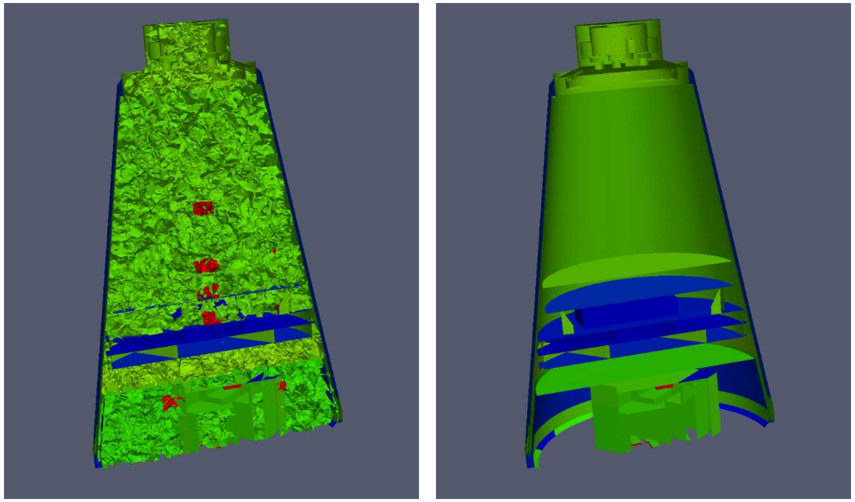
Data Partitioning

- Partitions should be load balanced and spatially coherent.



Load Balancing/Ghost Cells

- Automatic for Structured Meshes.
- Partitioning/ghost cells for unstructured is “manual.”
- Use the D3 filter for unstructured
  - (Filters → Alphabetical → D3)



Job Size Rules of Thumb

- Structured Data
  - –Try for max 20 M cell/processor.
  - –Shoot for 5 – 10 M cell/processor.
- Unstructured Data
  - –Try for max 1 M cell/processor.
  - –Shoot for 250 – 500 K cell/processor.

Avoiding Data Explosion

- Pipeline may cause data to be copied, created, converted.
- This advice **only for dealing with very large amounts of data**.
  - –Remaining available memory is low.

Topology Changing, No Reduction

- Append Datasets
- Append Geometry
- Clean
- Clean to Grid
- Connectivity
- D3
- Delaunay 2D/3D
- Extract Edges
- Linear Extrusion
- Loop Subdivision
- Reflect
- Rotational Extrusion
- Shrink
- Smooth
- Subdivide
- Tessellate
- Tetrahedralize
- Triangle Strips
- Triangulate

Topology Changing, Moderate Reduction

- Clip
- Decimate
- Extract Cells by Region
- Extract Selection
- Quadric Clustering
- Threshold

Similar: Extract Subset

Topology Changing, Dimension Reduction

- Cell Centers
- Contour
- Extract CTH Fragments
- Extract CTH Parts
- Extract Surface
- Feature Edges
- Mask Points
- Outline (curvilinear)
- Slice
- Stream Tracer

Adds Field Data

- Block Scalars
- Calculator
- Cell Data to Point Data
- Compute Derivatives
- Curvature
- Elevation
- Generate Ids
- Gen. Surface Normals
- Gradient
- Level Scalars
- Median
- Mesh Quality
- Octree Depth Limit
- Octree Depth Scalars
- Point Data to Cell Data
- Process Id Scalars
- Random Vectors
- Resample with dataset
- Surface Flow
- Surface Vectors
- Texture Map to…
- Transform
- Warp (scalar)
- Warp (vector)

Total Shallow Copy or Output Independent of Input

- Annotate Time
- Append Attributes
- Extract Block
- Extract Datasets
- Extract Level
- Glyph
- Group Datasets
- Histogram
- Integrate Variables
- Normal Glyphs
- Outline
- Outline Corners
- Plot Global Variables Over Time
- Plot Over Line
- Plot Selection Over Time
- Probe Location
- Temporal Shift Scale
- Temporal Snap-to-Time-Steps
- Temporal Statistics

Special Cases

- Temporal Filters
  - Temporal Interpolator
  - Particle Tracer
  - Temporal Cache
- Programmable Filter {...}

Culling Data

- Reduce dimensionality early.
  - –Contour and slice “see” inside volumes.
- Prefer data reduction of extraction.
  - –Slice instead of Clip.
  - –Contour instead of Threshold.
- Only extract when reducing an order of magnitude or more.
  - –Can still run into troubles.

Culling Data

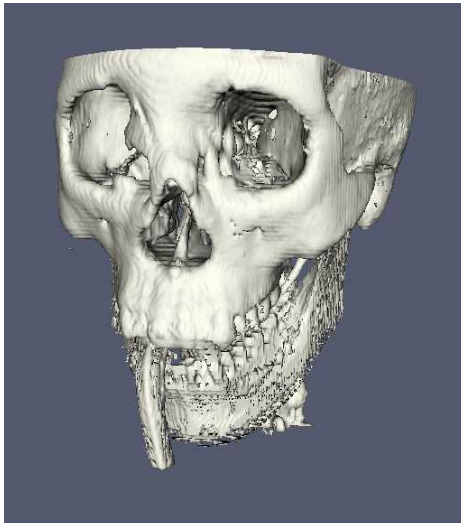
- Experiment with subsampled data.
  - Extract Subset
- Use caution.
  - Subsampled data may be lacking.
  - Use full data to draw final conclusions.

Rendering Modes

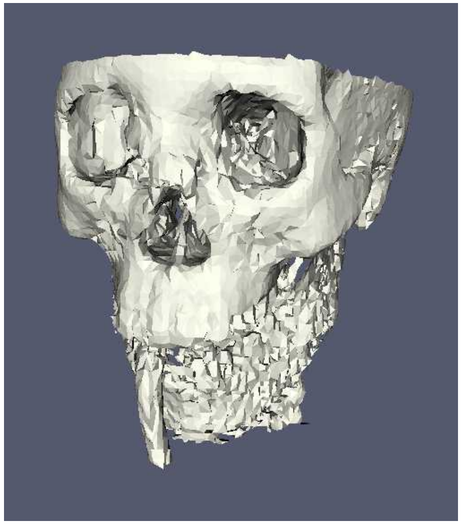
- Still Render
  - Full detail render.
- Interactive Render
  - Sacrifices detail for speed.
  - Provides quick rendering rate.
  - Used when interacting with 3D view.

Level of Detail (LOD)

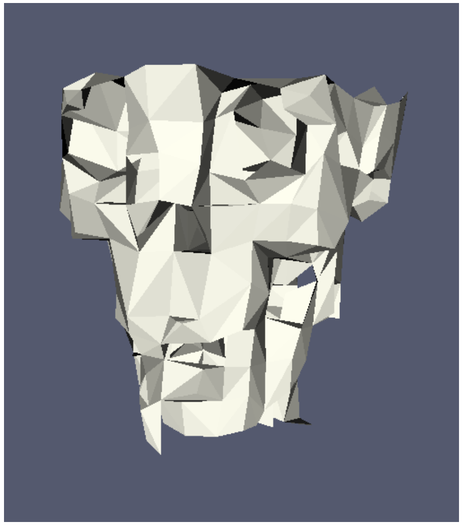
- Geometric decimation.
- Used only with Interactive Render



Original Data



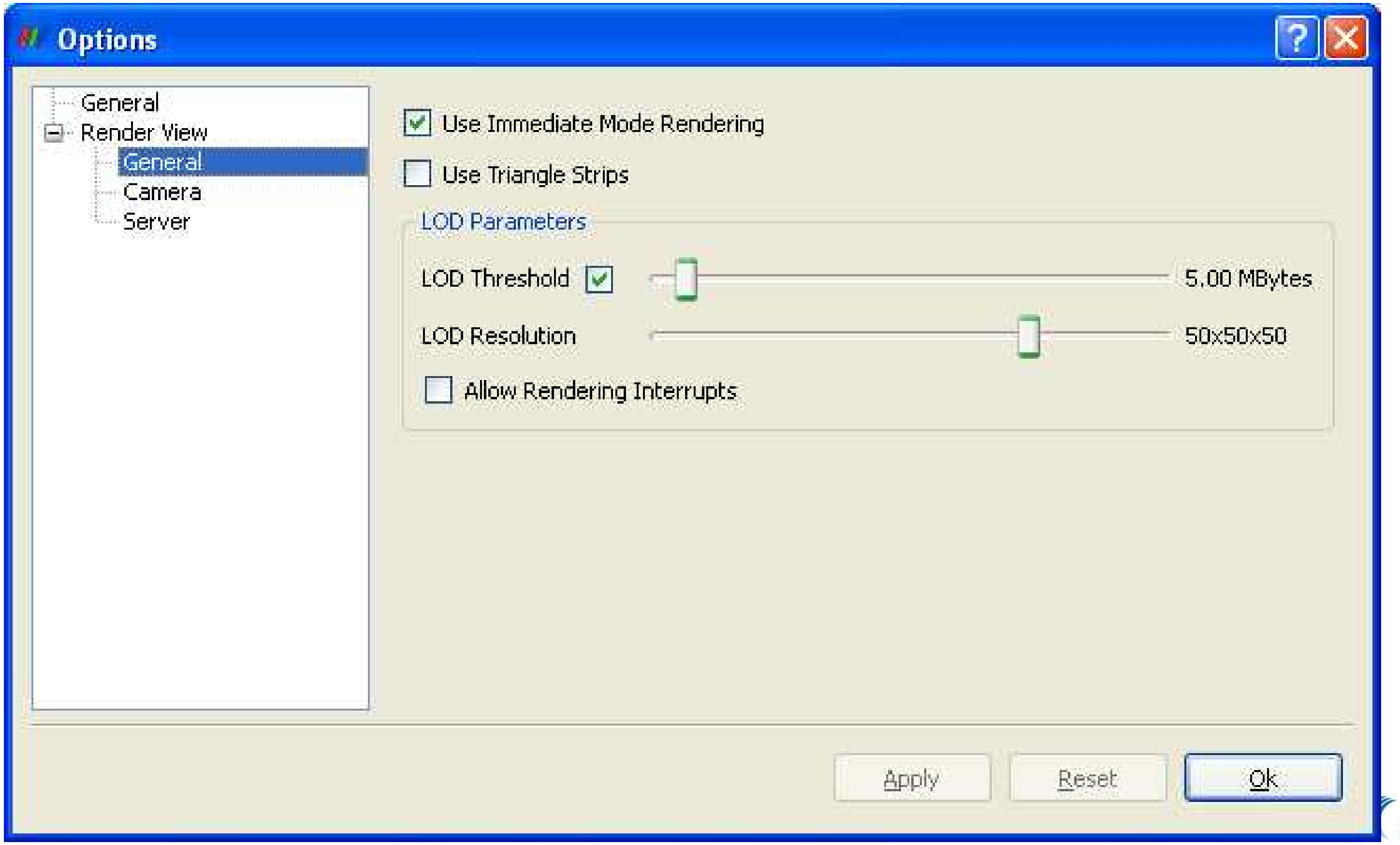
Divisions: 50x50x50



Divisions: 10x10x10

3D Rendering Parameters

Edit → Settings, Render View → General



☑ Use Immediate Mode Rendering

□ Use Triangle Strips

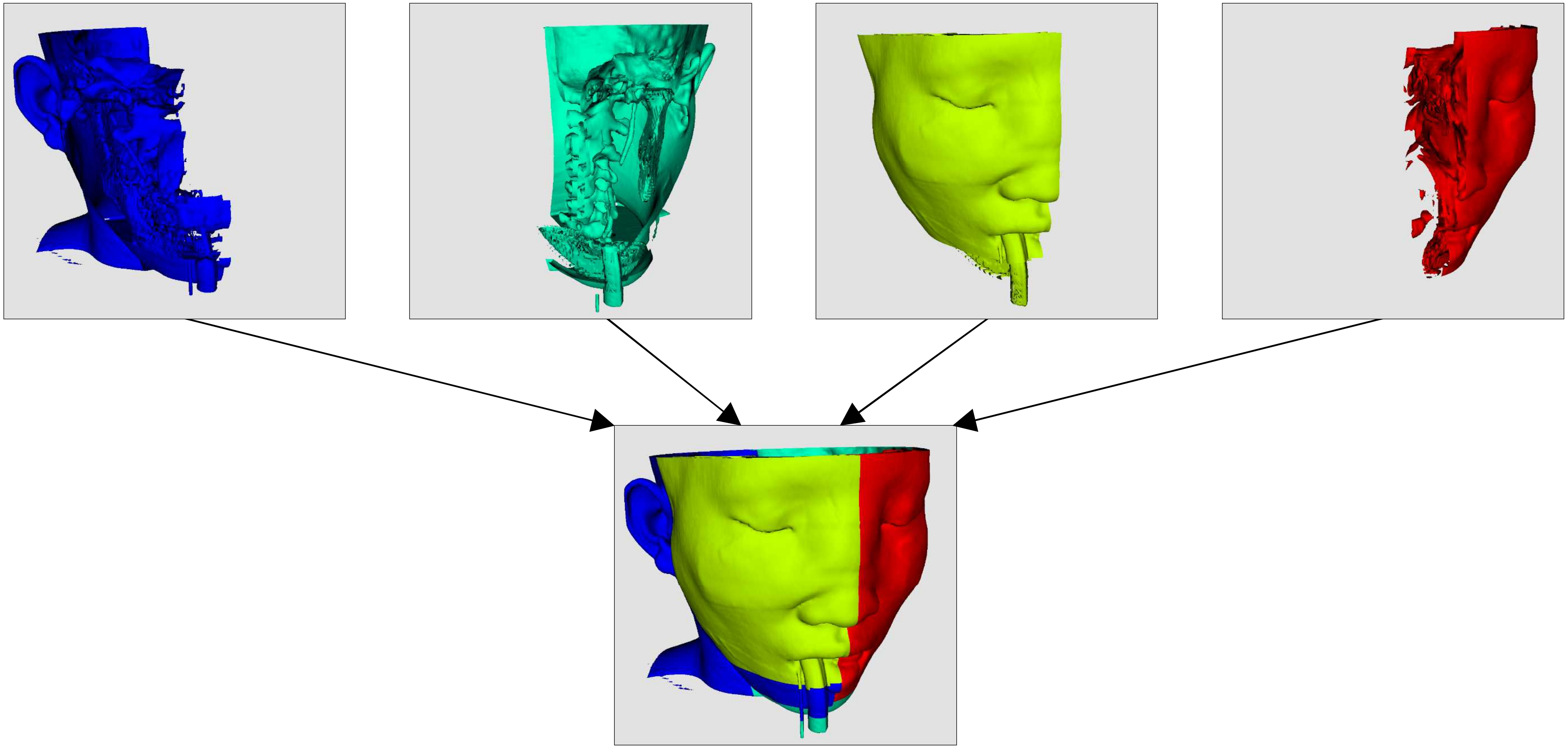
LOD Parameters

LOD Threshold ☑ 5.00 MBytes

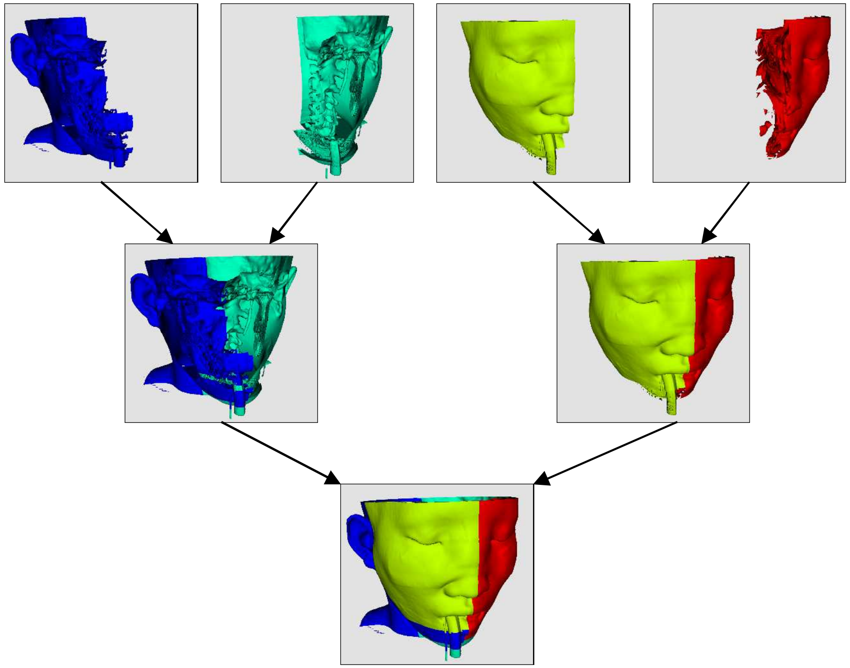
LOD Resolution 50x50x50

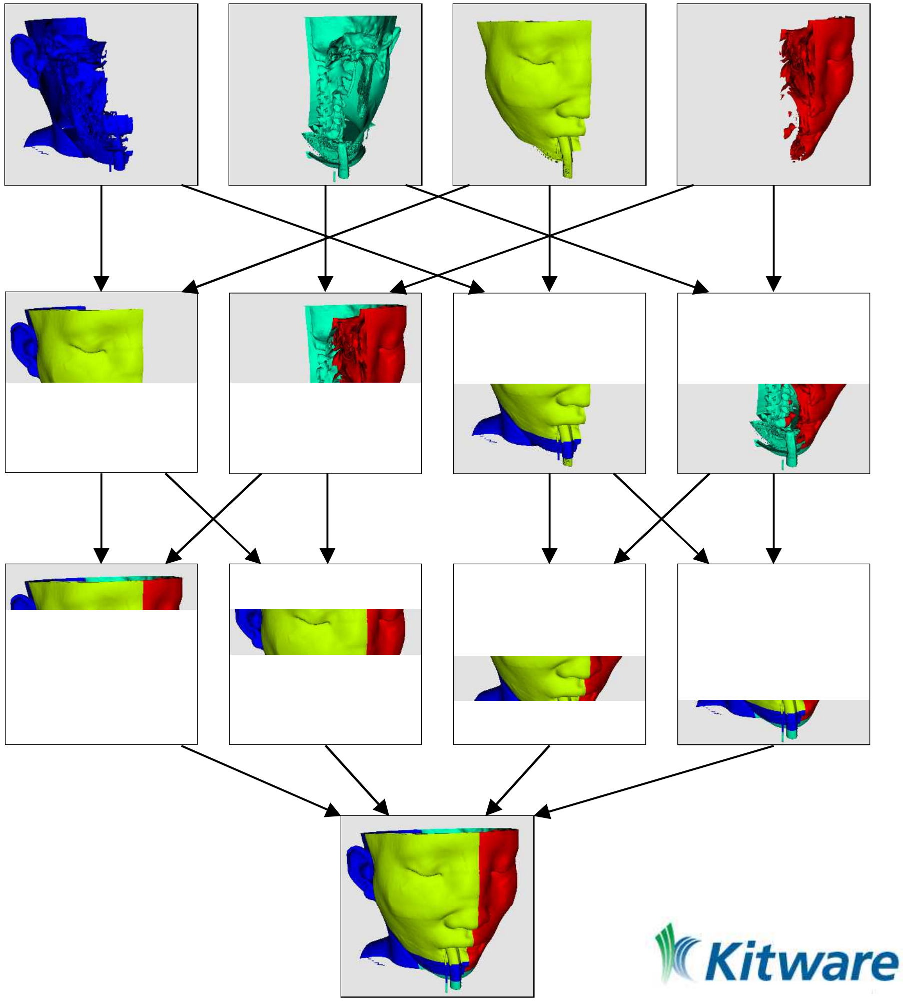
□ Allow Rendering Interrupts

Parallel Rendering

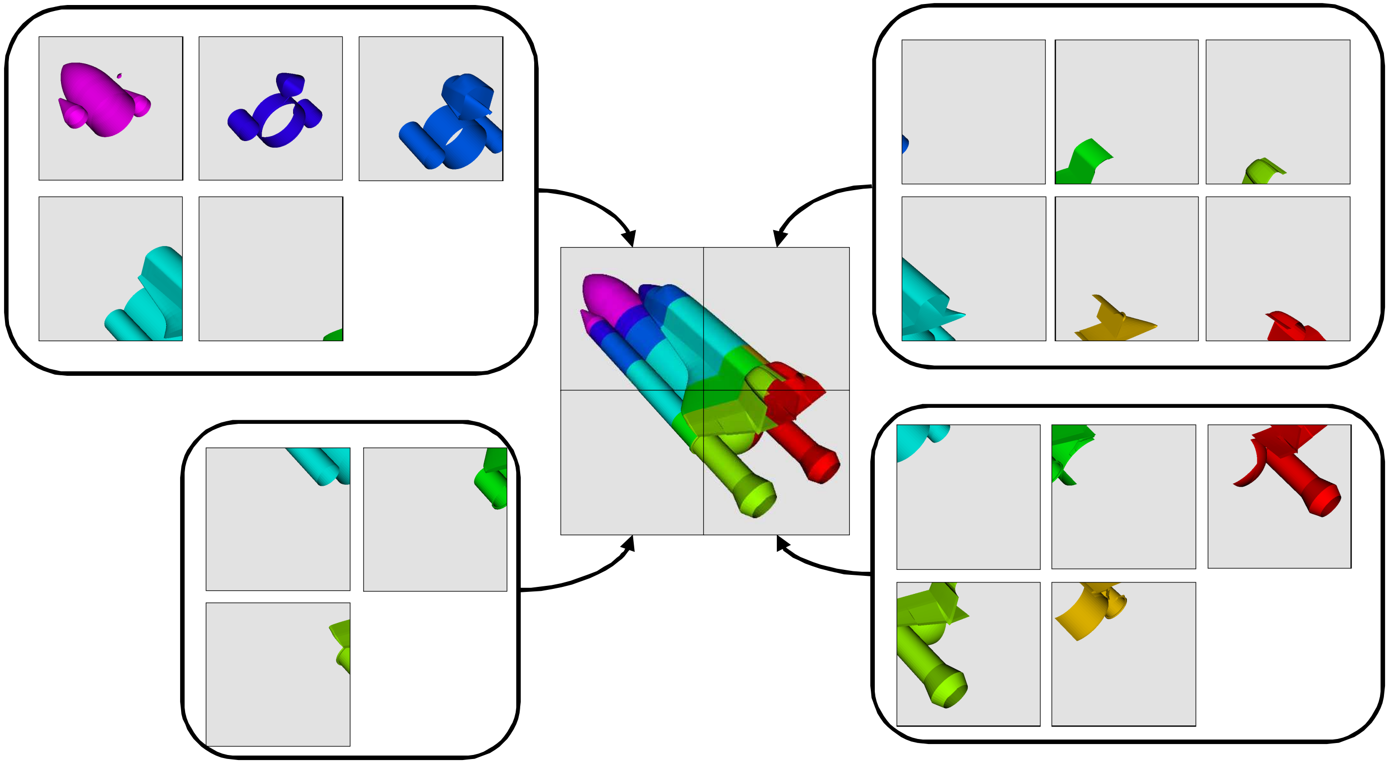


Parallel Rendering



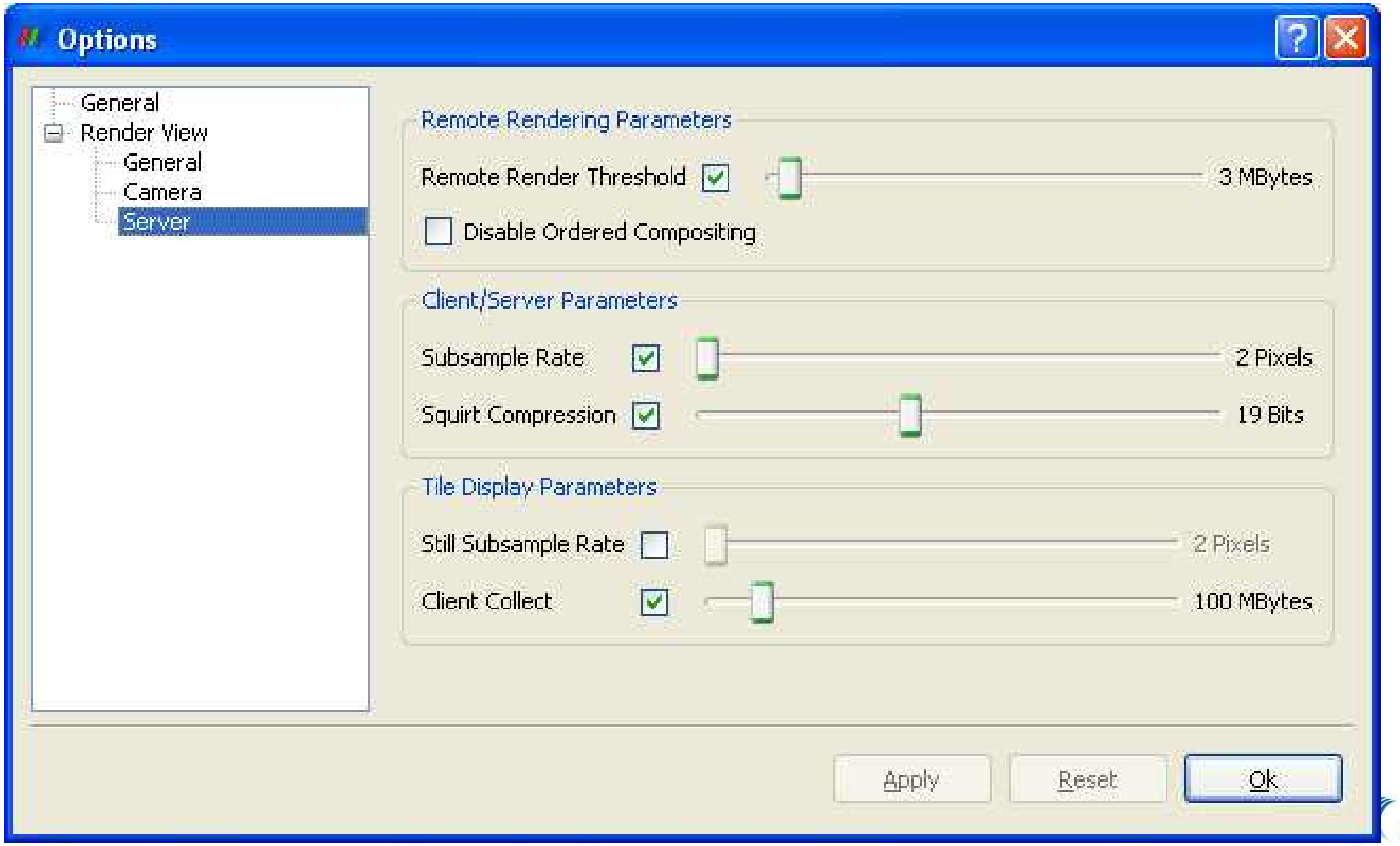


Tiled Displays



Parallel Rendering Parameters

Edit → Settings, Render View → Server



Remote Rendering Parameters

Remote Render Threshold ☑ 3 MBytes

□ Disable Ordered Compositing

Client/Server Parameters

Subsample Rate ☑ 2 Pixels

Squirt Compression ☑ 19 Bits

Tile Display Parameters

Still Subsample Rate □ 2 Pixels

Client Collect ☑ 100 MBytes

Parameters for Large Data

- Use Immediate Mode Rendering on.
- Use Triangle Strips off.
- Try LOD Threshold *off*.
  - Also try LOD Resolution 10x10x10.
- Always have remote rendering on.
- Turn on subsampling.
  - Try larger subsampling rates.
- Squirt Compression on.

Python Scripting for ParaView

Motivation

- Scripting
  - –Makes automation possible
  - –Used for batch processing
  - –Only means to visualize on supercomputers
- Python
  - –Cross-platform, easily extensible
  - –Object-oriented
  - –Supported by a huge community of avid developers and programmers

Compiling ParaView for Python support

- ***CMake*** Variables
  - –PARAVIEW_ENABLE_PYTHON: ON
    - Must be ON to enable python support
  - –PARAVIEW_USE_MPI: ON | OFF
    - Must be ON for MPI enabled server/pvbatch
  - –PARAVIEW_BUILD_QT_GUI: ON | OFF
    - Must be ON to build the ParaView’s Qt-based UI

Python in ParaView

- Standard python interpreter (***python***)
  - Set *PYTHON_PATH* to directory containing ParaView modules
  - Import relevant ParaView modules
- ParaView's python client (***pvpython***)
  - Python interpreter with ParaView initialization plus sets the path to ParaView modules
- ParaView's batch client (***pvbatch***)
  - Same as *pvpython* without remote server connection capabilities
  - Can be run in parallel (using *mpirun* etc.)

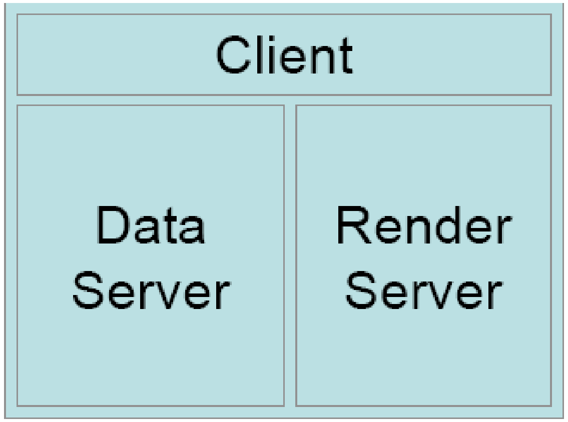
ParaView Configurations

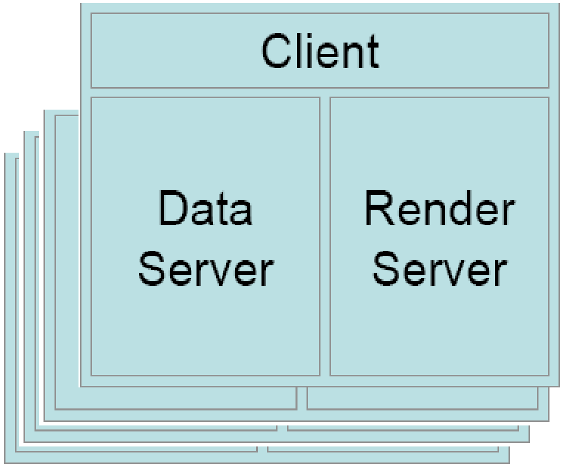
- Standalone

  (***pvpython***)

- Batch

  (***pvbatch***)





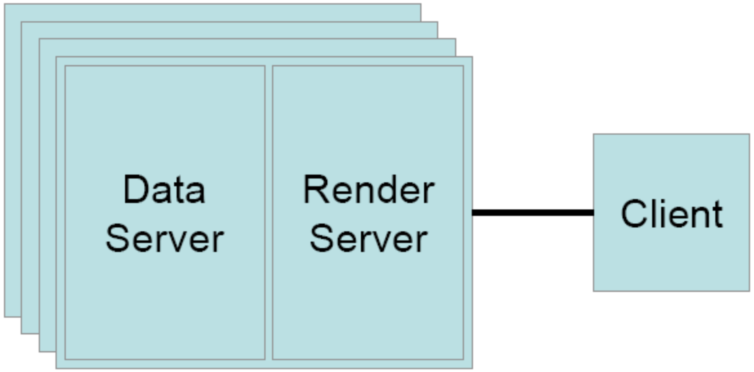
ParaView Configurations

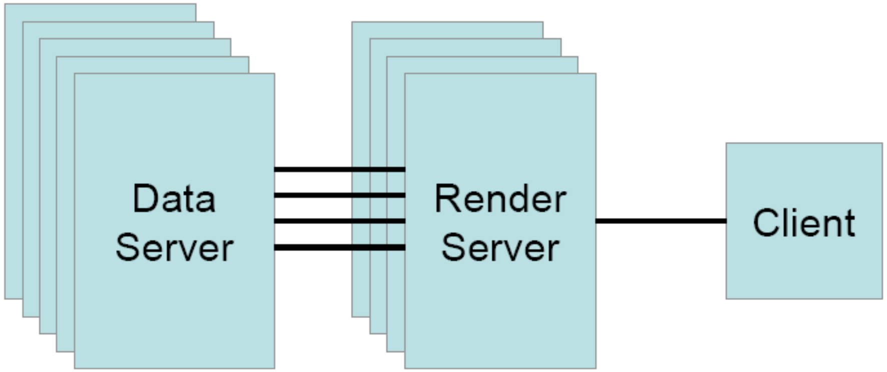
- Client – Server

  (***pvpython*** + ***pvserver***)

- Client – Render Server – Data Server

  (***pvpython*** + ***pvdataserver*** + ***pvrenderserver***)





Getting started with *pvpython*

- Import ParaView's python module

```
>>> from paraview import servermanager
```

- Connect to a server
  - For standalone operation (or batch mode)

```
>>> servermanager.Connect()
```

  - For additional help on Connect

```
>>> help(servermanager.Connect)
```

servermanager.Connect()

- Connect to *pvserver* running on *amber*

```
>>> connection = servermanager.Connect(“amber”)
```

- Connect to pvdataserver running on amber at port 10234 and pvrenderserver running on destiny at port 10235

```
>>> connection = servermanager.Connect(“amber”,10234,“destiny”,“10235”)
```

- Sets servermanager.ActiveConnection to the connection object

- To disconnect from the server

```
>>> servermanager.Disconnect()
```

Creating a simple visualization

- Create a cone

```
>>> cone = servermanager.sources.ConeSource()
```

- Create a view to show the cone

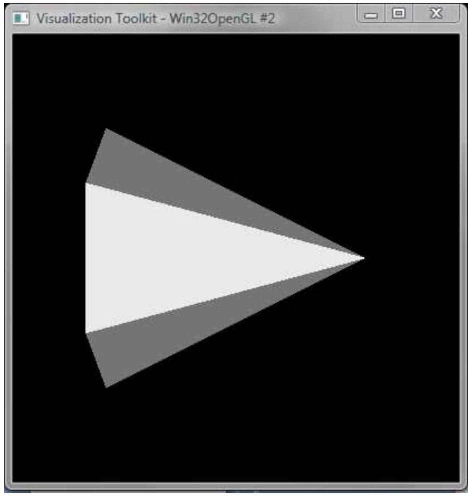
```
>>> view = servermanager.CreateRenderView()
```

- Show the cone in the view

```
>>> repCone = servermanager.CreateRepresentation(cone, view)
```

- Render

```
>>> view.ResetCamera()
>>> view.Render()
```



servermanager sub-modules

- ***servermanager*** module sub-modules:
  - ***sources*** – collection of data sources/readers eg. *ConeSource*, *SphereSource*, *ExodusIIReader* etc.
  - ***filters*** – collection of data processors eg. *Cut*, *Clip*, *Contour*, *ShrinkFilter* etc.
  - ***rendering*** – collection of rendering items eg. *RenderView*, *PVLookupTable* etc.
  - ***animation*** – collection of animation components eg. *AnimationScene* etc.
- List all available classes:

```
>>> dir(servermanager.sources)
```

- Documentation for each class:

```
>>> help(servermanager.sources.ConeSource)
```

help(servermanager.sources.ConeSource)

```
class ConeSource(SourceProxy)
  The Cone source can be used to add a polygonal cone to the 3D scene. The output of the
    Cone source is polygonal data.
  Data descriptors defined here:
  Capping
       If this property is set to 1, the base of the cone will be capped with a filled polygon.
    Otherwise, the base of the cone will be open.
  Center
     This property specifies the center of the cone.
  Height
     This property specifies the height of the cone.
  Radius
     This property specifies the radius of the base of the cone.
  Resolution
     This property indicates the number of divisions around the cone. The higher this number,
    the closer the polygonal approximation will come to representing a cone, and the more
    polygons it will contain.
```

Changing Parameters

- ***help(obj or class)*** can be used to obtain the list of available parameters
- Getting the current value

```
>>> param = cone.Radius
>>> center0 = cone.Center[0]
```

- Setting a new value

```
>>> cone.Radius = 10
>>> cone.Center[0] = 120.0333
>>> cone.Center = [120, 130, 121.09]
>>> repCone.Representation = “Wireframe”
```

- Parameter values can be specified when instantiating

```
>>> cone = servermanager.sources.ConeSource(Radius=10.5,
        Center=[10, 10, 10] )
```

Using Filters

- Filters are available in servermanager.filters sub-module

- Similar to creating sources, with required Input

```
>>> shrink = servermanager.filters.ShrinkFilter(Input=cone)
>>> shrink.ShrinkFactor = 0.8
>>> repShrink = servermanager.CreateRepresentation(shrink, view)
>>> repCone.Visibility = False
>>> view.StillRender()
```

More about filters

- List available filters

```
>>> dir(servermanager.filters)
```

- Help about any filter

```
>>> help(servermanager.filters.Calculator)
```

Connecting to a particular output port

- Connecting to the first output port

```
>>> shrink.Input = inFilter
>>> shrink.Input = servermanager.OutputPort(inFilter, 0)
```

- Connecting to any other output port

```
>>> shrink.Input = servermanager.OutputPort(inFilter, 1)
```

Rendering

- Views
  - Pane to display data in
  - servermanager.CreateRenderView() creates a render view suitable for the active connection type
- Representations
  - Maps the data from a source to a view
  - Has parameters that control the appearance eg. *LookupTable*, *Color* etc.
  - servermanager.CreateRepresentation(source, view) creates representation suitable to show the data produced by the source in the given view

Color by an array

- To color by an array one must do the following:
  - – Specify the array to color with

```
>>> repShrink.ColorAttributeType = “POINT_DATA”
>>> repShrink.ColorArrayName = “Normals”
```

  - – Specify the scalar lookup table to map data to colors

```
>>> lut = servermanager.rendering.PVLookupTable()
>>> repShrink.LookupTable = lut
```

- Setup data-to-color mapping
  - RGBPoints is a list of 4-tuples (scalar value, red, green, blue)

```
>>> lut.RGBPoints = [0.0, 0, 0, 1,    1.0, 1, 0, 0]
```

Script with scalar coloring

```
>>> from paraview import servermanager as sm
>>> sm.Connect()
>>> cone = sm.sources.ConeSource(Radius=0.9)
>>> shrink = sm.filters.ShrinkFilter(Input=cone)
>>> shrink.ShrinkFactor = 0.8
>>> view = sm.CreateRenderView()
>>> repShrink = sm.CreateRepresentation(shrink, view)
>>> repShrink.ColorAttributeType = “POINT_DATA”
>>> repShrink.ColorAttributeName = “Normals”
>>> lut = sm.rendering.PVLookupTable()
>>> lut.RGBPoints = [0, 0,0, 1,  1, 1, 0, 0]
>>> repShrink.LookupTable = lut
>>> view.ResetCamera()
>>> view.StillRender()
```

Clipping a dataset

```
# Create the clip filter
>>> clipper = servermanager.filters.Clip()
# Assign input
>>> clipper.Input = sphere
# Create the implicit plane that is used to define the clip function
>>> plane = servermanager.implicit_functions.Plane()
>>> plane.Normal = [0.5, 0.5, 0.0]
# Assign the clip function
>>> clipper.ClipFunction = plane
>>> repClip = servermanager.CreateRepresentation(clipper, view)
# Reset camera and render
>>> view.ResetCamera()
>>> view.StillRender()
```

Data Information

- Classes in sources/filters are merely proxies for server-side VTK objects
- Data processing is done on the server, hence data is available on the server alone
- DataInformation provides an API to obtain information about the data on the client without fetching the entire data to the client

Fetching Data Information

- Update the source/filter

```
>>> shrink.UpdatePipeline()
```

- Need to update the source after any changes in parameters
- Rendering (view.StillRender()) automatically updates all sources in visible piplines
- Fetch the data information

```
>>> di = shrink.GetDataInformation()
```

Accessing Data Information

- Data-type information

```
>>> di.GetDataSetTypeAsString()
```

- Accessing point attributes

```
>>> pdi = di.GetPointDataInformation()
```

- Accessing arrays available as point attributes (***vtkPVDataSetAttributesInformation***)

```
>>> num_arrays = pdi.GetNumberOfArrays()
```

- Accessing information about an array (***vtkPVArrayInformation***)

```
>>> pa = pdi.GetArrayInformation(“Normals”)
>>> pa = pdi.GetArrayInformation(0)
>>> name = pa.GetName()
>>> num_comps = pa.GetNumberOfComponents()
>>> num_tuples = pa.GetNumberOfTuples()
>>> range = pa.GetComponentRange(0) ;# -1 for vector magnitude
```

Readers

- Available in the servermanager.sources sub-module
- Specify the filename (or filenames) using the FileName or FileNames parameter
- Readers may provides options to select which attribute arrays to load
- Soon a method will be available to create choose a reader give the filename. Currently, user has to select what reader to create to read the file

Reading a *.vtk file

```
>>> from paraview import servermanager
>>> servermanager.Connect()
>>> reader = servermanager.sources.
  LegacyVTKFileReader(FileNames="…/data.vtk")
>>> view = servermanager.CreateRenderView()
>>> repr = servermanager.CreateRepresentation(reader, view)
>>> view.ResetCamera()
>>> view.StillRender()
```

Read an Exodus file

- Create reader

```
>>> reader = servermanager.sources.ExodusIIReader()
>>> reader.FileName = “…/can.ex2”
```

- Update the reader information: causes the reader to read meta-data from the file

```
>>> reader.UpdatePipelineInformation()
```

- List available point arrays

```
>>> reader.PointResultArrayInfo
 Property name= PointResultArrayInfo
 value=[‘DISPL’, ‘0’, ‘VEL’, ‘0’, ‘ACCL’, ‘0’]
```

* Turn on a few arrays

```
>>> reader.PointResultArrayStatus = [‘DISPL’, ‘1’, ‘VEL’, ‘1’]
```

Datasets with Time

- Readers which provide time-support have TimestepValues parameter

```
>>> reader.TimestepValues
Property name= TimestepValues value = [0.0, 0.00010007373930420727,
0.00019990510190837085, ...]
```

- To request a particular time
  - UpdatePipeline(time) can be used to force the pipeline to update with the given time
  - View has a ViewTime parameter which is the time the view will request from all visible pipelines

Animating through available time steps

```
>>> reader = servermanager.ExodusIIReader(…)
…
>>> tsteps = reader.TimestepValues
>>> for cur_time in tsteps:
>>>      view.ViewTime = cur_time
>>>      view.ResetCamera()
>>>      view.StillRender()
```

Accessing data directly

- In client-server configurations, the python script is run on the client while the data processing and hence the data is on the server
- Generally use DataInformation to obtain information about the data on the client
- servermanager.Fetch() can be used to deliver data to the client
- Modes of operation for Fetch()
    - Append all of the data together and bring it to the client (only available for polygonal and unstructured datasets). Note: Do not do this if data is large otherwise the client will run out of memory.
    - Bring data from a given process to the client.
    - Use a reduction algorithm and bring its output to the client. For example, find the minimum value of an attribute.

servermanager.Fetch

- To fetch the appended data on the client:

```
>>> data = servermanager.Fetch(source)
```

- To fetch data from a particular processes

```
>>> dataP1 = servermanager.Fetch(source, 1)
```

- To fetch the sum of values for each attribute arrays

```
>>> mm =servermanager.sources.MinMax()
>>> mm.Operation = “SUM”
>>> sumdata = servermanager.Fetch(source, mm, mm)
```

  - sumdata is polydata with 1 point (and 1 cell) with cell/point arrays containing the sum of all the values in that array across all the processess.

Loading state saved from GUI

```
>>> from paraview import servermanager
# Connect to a server
>>> servermanager.Connect()
# Load the state
>>> servermanager.LoadState(“…/state.pvsm”)
# Obtain the first render view (for multiple views use GetRenderViews())
>>> view = servermanager.GetRenderView()
>>> view.StillRender()
```

Scripting for pvbatch

- Same as pvpython except
  - It can be run in parallel
  - One cannot connect to remote server (only servermanager.Connect() calls are supported)
- The python script is run on the 0th node

Scripting from within GUI

- Things to remember
  - Don't use servermanager.Connect() or servermanager.Disconnect(). All connection related operation have to be done using the GUI
  - servermanager.ActiveConnection is automatically setup to refer to the connection made by the GUI
  - The python shell and the GUI are both working on the same engine hence changes in one will have effects on the other

Scripting from within GUI

```
>>> sphere = servermanager.sources.SphereSource()
# Make the GUI aware of this sphere
>>> servermanager.Register(sphere, “SphereFromPython”)
>>> view = servermanager.GetRenderView()
# Create a new representation in a existsing view
>>> repSphere = servermanager.CreateRepresentation(sphere, view)
>>> servermanager.Register(repSphere)
# Create a new view
>>> view2 = servermanager.CreateRenderView()
>>> servermanager.Register(view2)
# Create a new representation for the new view
>>> rep2 = servermanager.CreateRepresentation(sphere, view2)
>>> servermanager.Register(rep2)
```

Register/UnRegister

- Register(object, [registrationGroup="..", registrationName="…")
  - Registers an object so that the GUI becomes aware of it
  - registrationName is the name with which the object appears in the GUI
  - If registrationGroup is not specified, the group is inferred from the type of the object
  - Returns a tuple (group, name) which which the object is registered on success
- UnRegister(object, [registrationGroup="...", registrationName="…")
  - Unregisters a previously registered object
  - The GUI treats the object as if it were deleted

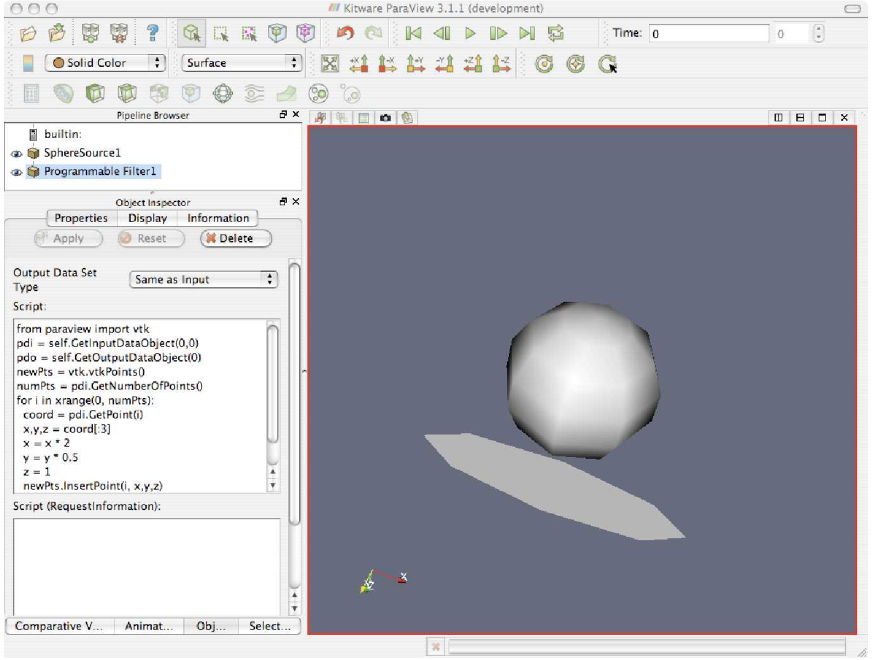
Saving state from python

- State of all those objects that are registered can be saved
- Objects not registered are ignored
- To be able to load saved state from GUI it is essential that default explicit registrationGroup is not specified.

```
>>> servermanager.Connect()
# Set up pipeline, views etc.
….
# Register all objects
>>> servermanager.Register(source, ..)
>>> servermanager.Register(view, …)
# Save state
>>> servermanager.SaveState("…/state.pvsm")
```

Python Programmable Filter

- Used to write custom filters using Python
- Python is used to data processing
- servermanager module is not accesible either use paraview.vtk or vtk for creating VTK filters etc.



Output Data Set Type: Same as Input

Script:

```
from paraview import vtk
pdi = self.GetInputDataObject(0,0)
pdo = self.GetOutputDataObject(0)
newPts = vtk.vtkPoints()
numPts = pdi.GetNumberOfPoints()
for i in xrange(0, numPts):
  coord = pdi.GetPoint(i)
  x,y,z = coord[:3]
  x = x * 2
  y = y * 0.5
  z = 1
  newPts.InsertPoint(i, x,y,z)
```

Script (RequestInformation):

Python Programmable Filter

```
>>> from paraview import vtk
>>> #reads a poly data and modifies the geometry
>>> pdi = self.GetInputDataObject(0,0)
>>> pdo = self.GetOutputDataObject(0)
>>> newPts = vtk.vtkPoints()
>>> numPts = pdi.GetNumberOfPoints()
>>> for i in xrange(0, numPts):
>>>     coord = pdi.GetPoint(i)
>>>     x,y,z = coord[:3]
>>>     x = x * 2
>>>     y = y * 0.5
>>>     z = 1
>>>     newPts.InsertPoint(i, x,y,z)
>>> pdo.SetPoints(newPts)
```

Conclusion

- Python is the main scripting language for ParaView
- Python can be used to write pure client side code as well as for server side data processing (using programmable filter)
- paraview.servermanager module provides components used for client-side programming. It also has several demo*() functions which can be used as guidelines for writing custom scripts
- paraview.vtk or simply vtk modules are provided for server side python programming. These provide access to VTK classes through python
- We are constantly working on improving the scripting API to make is easier to use and more python friendly

Further Reading

- Amy Henderson Squillacote. *The ParaView Guide*. Kitware, Inc., 2006.
- http://www.paraview.org/Wiki/ParaView
- http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server

Further Reading
Visualization and Customization

- Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit*. Kitware, Inc., fourth edition, 2006.
- Kitware Inc. *The VTK User's Guide*. Kitware, Inc., 2006.
- Jasmin Blanchette and Mark Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall, 2006.

Further Reading
Parallel VTK Topics

- James Ahrens, Charles Law, Will Schroeder, Ken Martin, and Michael Papka. “A Parallel Approach for Efficiently Visualizing Extremely Large, Time-Varying Datasets.” Technical Report #LAUR-00-1620, Los Alamos National Laboratory, 2000.
- James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, and Michael Papka. “Large-Scale Data Visualization Using Parallel Data Streaming.” *IEEE Computer Graphics and Applications*, 21(4): 34–41, July/August 2001.
- Andy Cedilnik, Berk Geveci, Kenneth Moreland, James Ahrens, and Jean Farve. “Remote Large Data Visualization in the ParaView Framework.” *Eurographics Parallel Graphics and Visualization 2006*, pg. 163–170, May 2006.

Further Reading
Advanced Pipeline Execution

- James P. Ahrens, Nehal Desai, Patrick S. McCormic, Ken Martin, and Jonathan Woodring. “A Modular, Extensible Visualization System Architecture for Culled, Prioritized Data Streaming.” *Visualization and Data Analysis 2007, Proceedings of SPIE-IS&T Electronic Imaging*, pg 64950I-1–12, January 2007.
- John Biddiscombe, Berk Geveci, Ken Martin, Kenneth Moreland, and David Thompson. “Time Dependent Processing in a Parallel Pipeline Architecture.” *IEEE Visualization 2007*. October 2007.

Further Reading
Parallel Rendering

- Kenneth Moreland, Brian Wylie, and Constantine Pavlakos. “Sort-Last Parallel Rendering for Viewing Extremely Large Data Sets on Tile Displays.” *Proceedings of IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, pg. 85–92, October 2001.
- Kenneth Moreland and David Thompson. “From Cluster to Wall with VTK.” *Proceddings of IEEE 2003 Symposium on Parallel and Large-Data Visualization and Graphics*, pg. 25–31, October 2003.
- Kenneth Moreland, Lisa Avila, and Lee Ann Fisk. “Parallel Unstructured Volume Rendering in ParaView.” *Visualization and Data Analysis 2007, Proceedings of SPIE-IS&T Electronic Imaging*, pg. 64950F-1–12, January 2007.