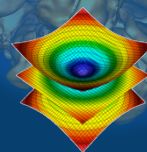


Visualization and Analysis of Massive Data with VisIt



ATPESC 2014

Argonne Training Program on Extreme-Scale Computing
Monday August 11, 2014

Cyrus Harrison

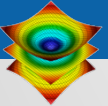
Lawrence Livermore National Laboratory

cyrush@llnl.gov



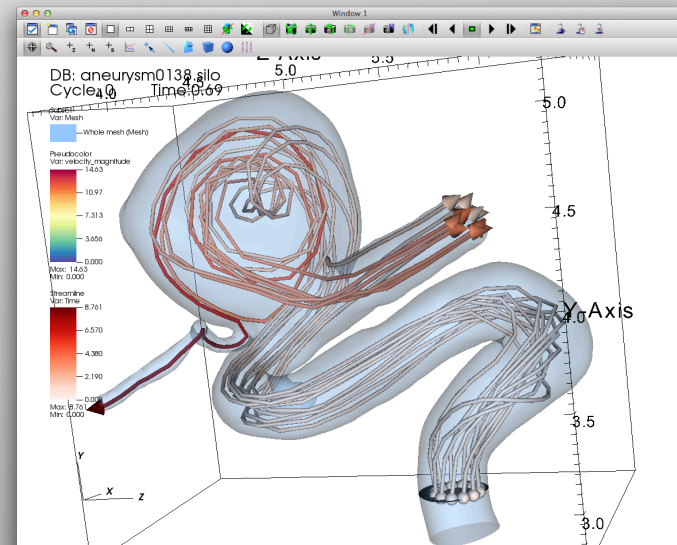
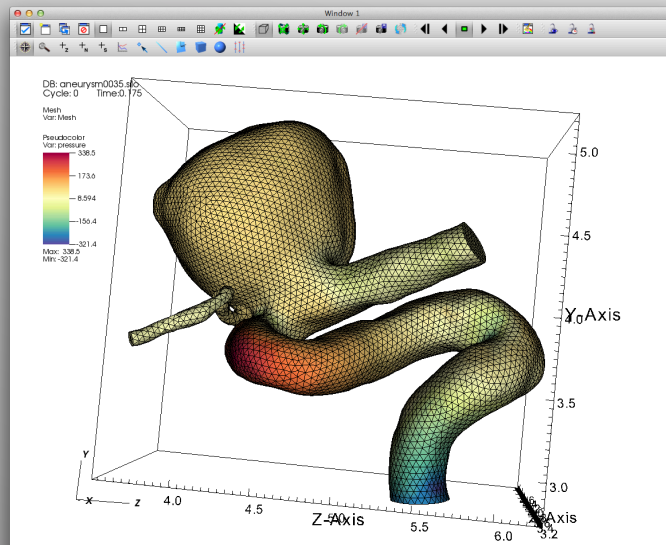
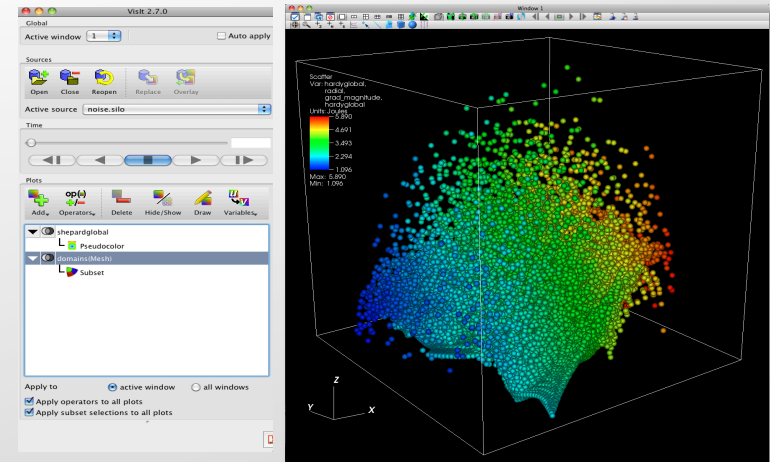
LLNL-PRES-658008

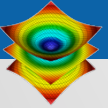
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC



Tutorial Outline

- VisIt Project Intro [5 min]
- Guided tour of VisIt [25 min]
- Hands on with an Aneurysm Simulation [30 min]





Tutorial Resources

- **Tutorial Prep:**

http://visitusers.org/index.php?title=Tutorial_Preparation

- **Example Datasets:**

http://visitusers.org/index.php?title=Tutorial_Data

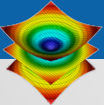
- **Blood Flow Hands-on:**

http://visitusers.org/index.php?title=Blood_Flow_Aneurysm_Tutorial

- **More Tutorial Materials (From past SC Tutorials):**

http://visitusers.org/index.php?title=Visit_Tutorial

- **Cyrus' Email: cyrush@llnl.gov**



Tutorial Data Acknowledgements

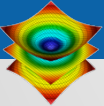
Aneurysm Simulation Data

Simulated using the LifeV (<http://www.lifev.org/>)
finite element solver.

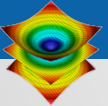
Available thanks to:

Gilles Fourestey and Jean Favre
Swiss National Supercomputing Centre

<http://www.cscs.ch/>

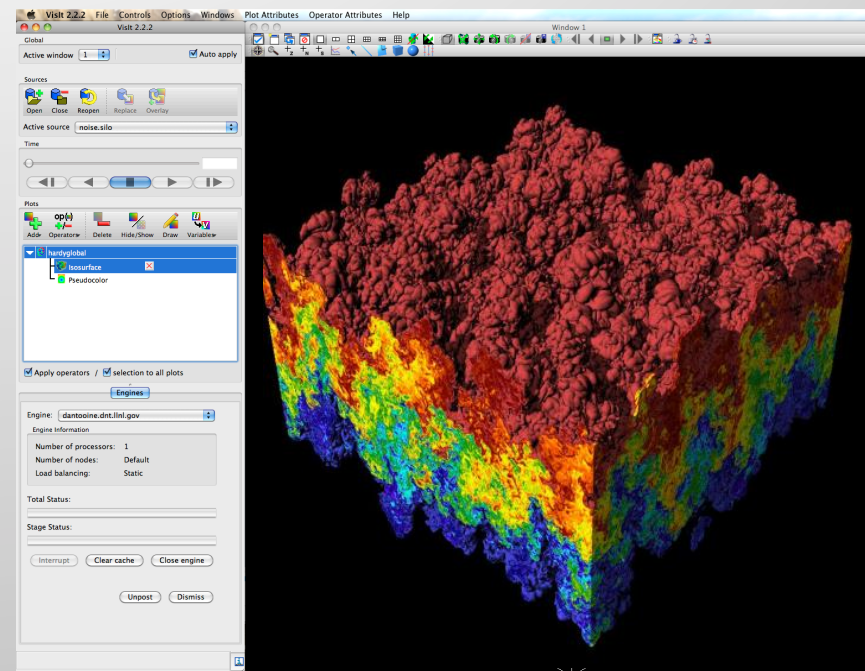


VisIt Project Introduction

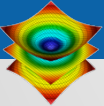


VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data.

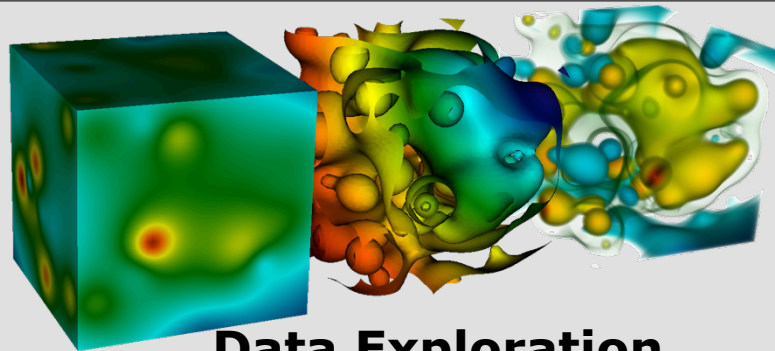
- Production end-user tool supporting scientific and engineering applications.
- Provides an infrastructure for parallel post-processing that scales from desktops to massive HPC clusters.
- Source released under a BSD style license.



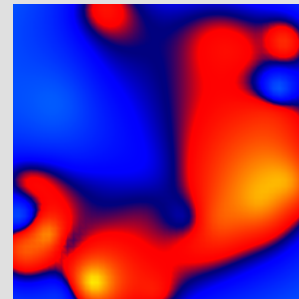
Density Isovolume of a $3K^3$ (27 billion cell) dataset



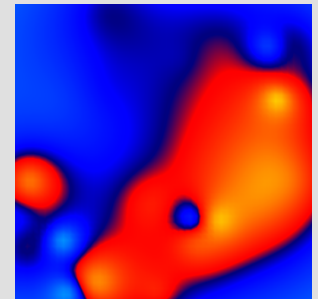
VisIt supports a wide range of use cases.



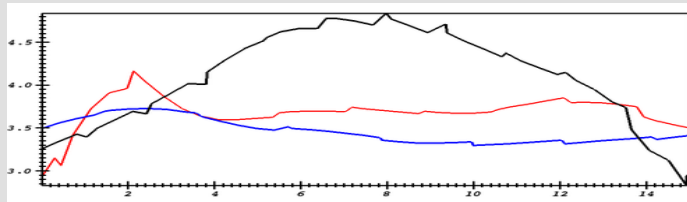
Data Exploration



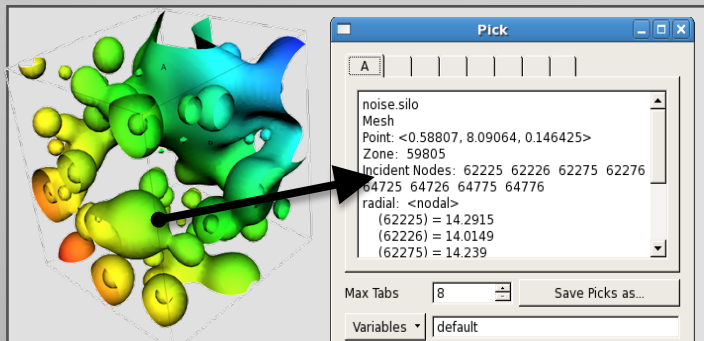
?



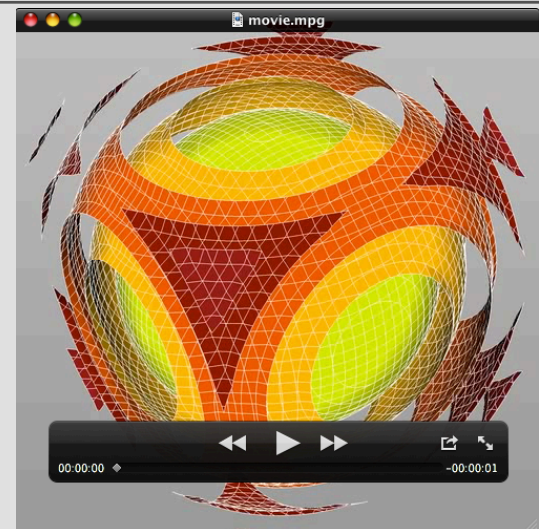
Comparative Analysis



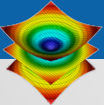
Quantitative Analysis



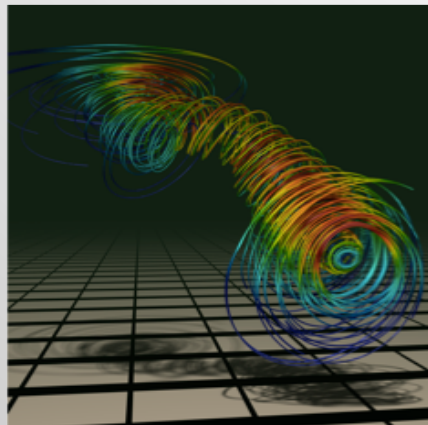
Visual Debugging



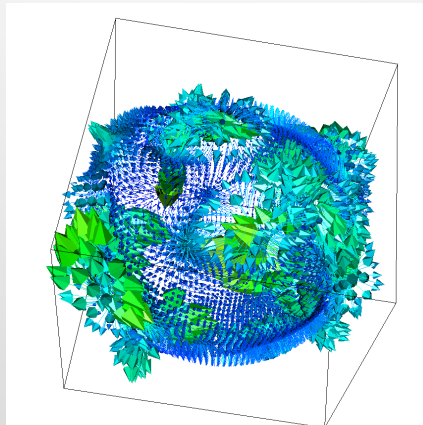
Presentation Graphics



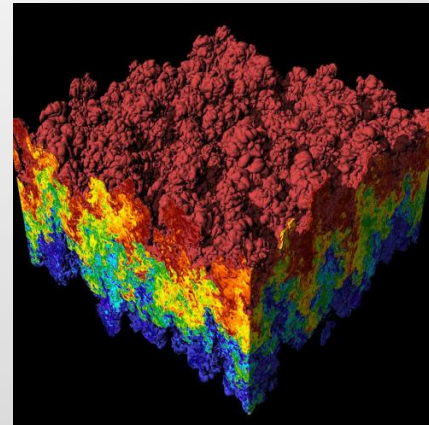
Examples of VisIt's visualization capabilities.



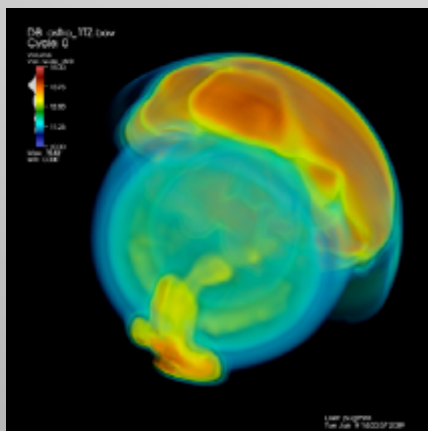
Streamlines



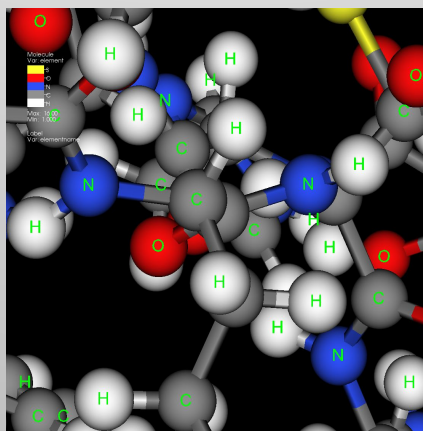
Vector / Tensor Glyphs



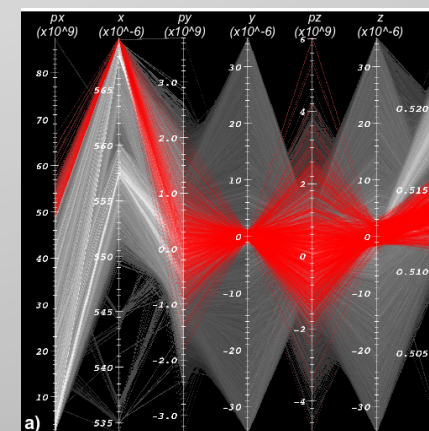
Pseudocolor Rendering



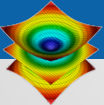
Volume Rendering



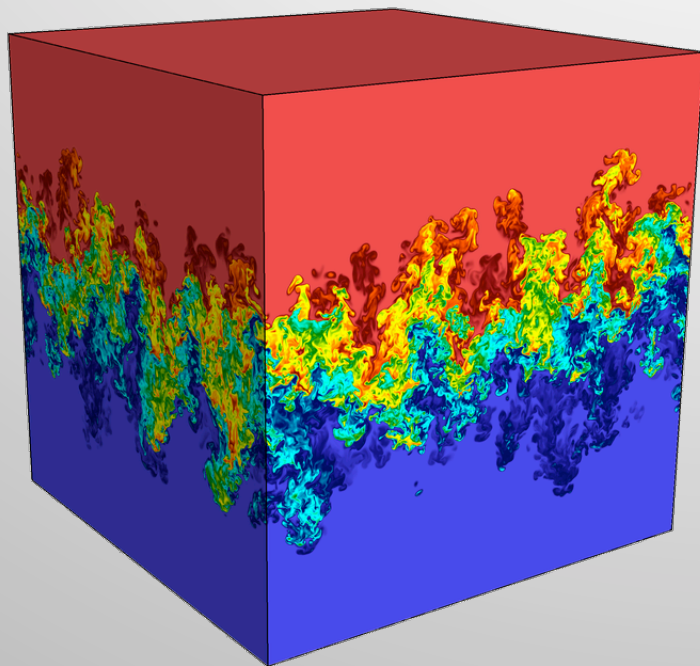
Molecular Visualization



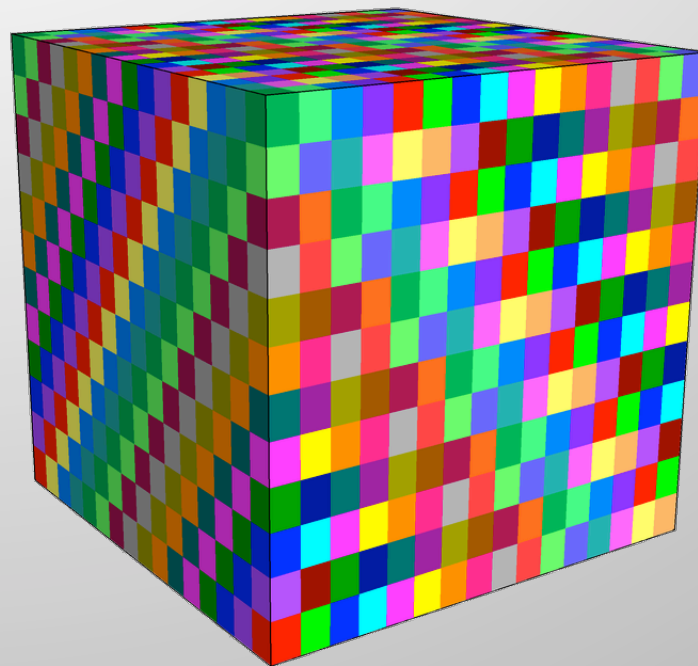
Parallel Coordinates



VisIt uses MPI for distributed-memory parallelism on HPC clusters.

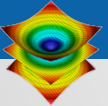


Full Dataset
(27 billion total cells)



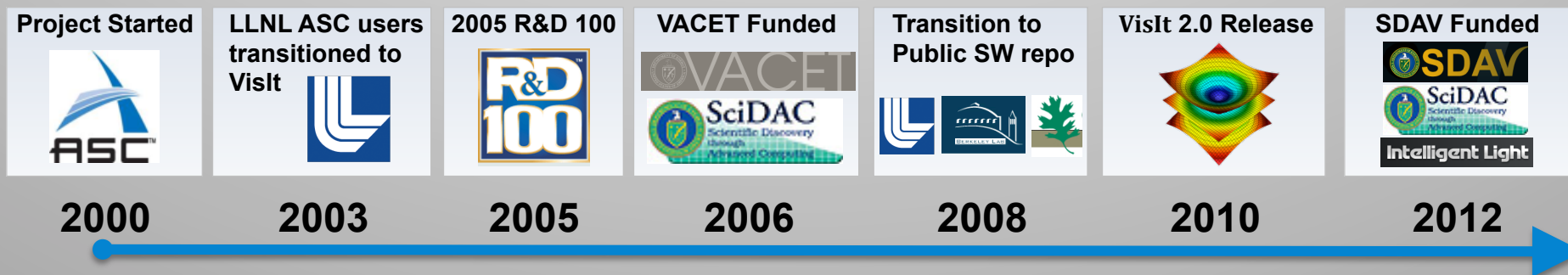
3072 sub-grids
(each 192x129x256 cells)

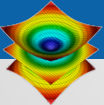
We are enhancing VisIt's pipeline infrastructure to also support threaded processing.



VisIt is a vibrant project with many participants.

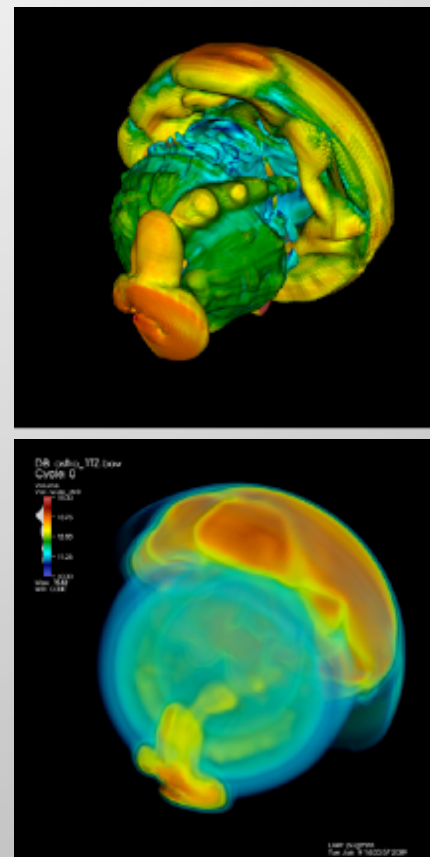
- The VisIt project started in 2000 to support LLNL's large scale ASC physics codes.
- The project grew beyond LLNL and ASC with research and development from DOE SciDAC and other efforts.
- VisIt is now supported by multiple organizations:
 - LLNL, LBNL, ORNL, Univ of Oregon, Univ of Utah, Intelligent Light, ...
- Over 75 person years of effort, 1.5+ million lines of code.





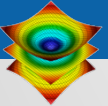
VisIt scales well on current HPC platforms.

Machine	Architecture	Problem Size	# of Cores
<i>Graph</i>	<i>X86_64</i>	20,001³ (8 T cells)	12K
Dawn	BG/P	15,871 ³ (4 T cells)	64K
Franklin	Cray XT4	12,596 ³ (2 T cells)	32K
JaguarPF	Cray XT5	12,596 ³ (2 T cells)	32K
Juno	X86_64	10,000 ³ (1 T cells)	16K
Franklin	Cray XT4	10,000 ³ (1 T cells)	16K
Ranger	Sun	10,000 ³ (1 T cells)	16K
Purple	IBM P5	8,000 ³ (0.5 T cells)	8K



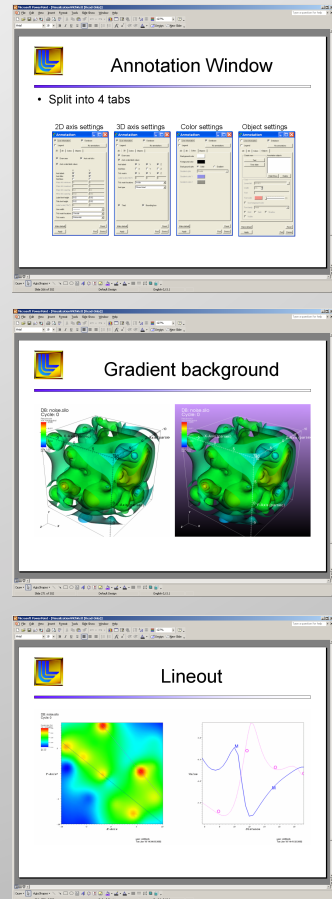
Scaling Studies of Isosurface Extraction and Volume Rendering (2009)

VisIt is also used daily by domain scientists.

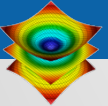


The VisIt team focuses on making a robust, usable product for end users.

- Regular releases (~ 6 / year)
 - Executables for all major platforms
 - End-to-end build process script ``build_visit``
- Customer Support and Training
 - visitusers.org, wiki for users and developers
 - Email lists: visit-users, visit-developers
 - Beginner and advanced tutorials
 - VisIt class with detailed exercises
- Documentation
 - “Getting data into VisIt” manual
 - Python interface manual
 - Users reference manual



Slides from the VisIt class



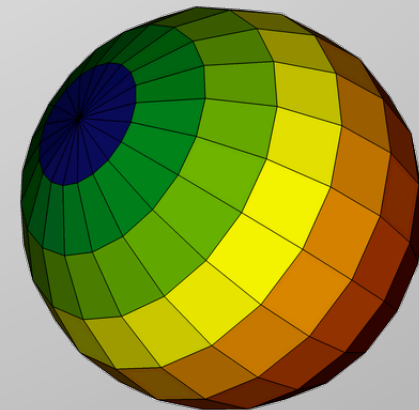
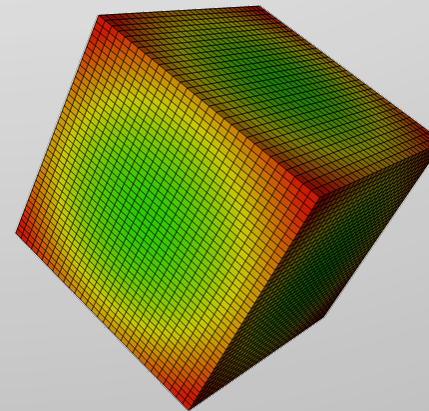
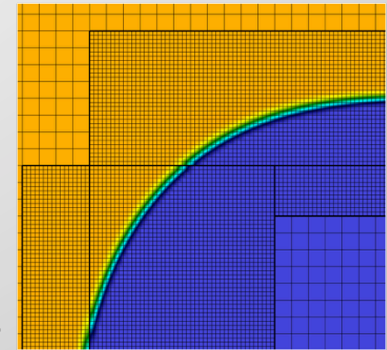
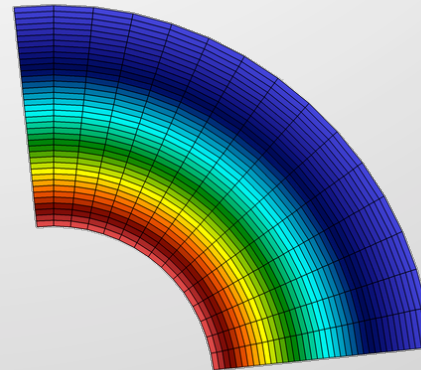
VisIt provides a flexible data model, suitable for many application domains.

■ Mesh Types:

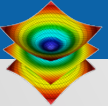
- Point, Curve, 2D/3D
Rectilinear, Curvilinear,
Unstructured
- Domain Decomposed, AMR
- Time Varying

■ Fields:

- Scalar, Vector, Tensor,
Material volume fractions,
Species

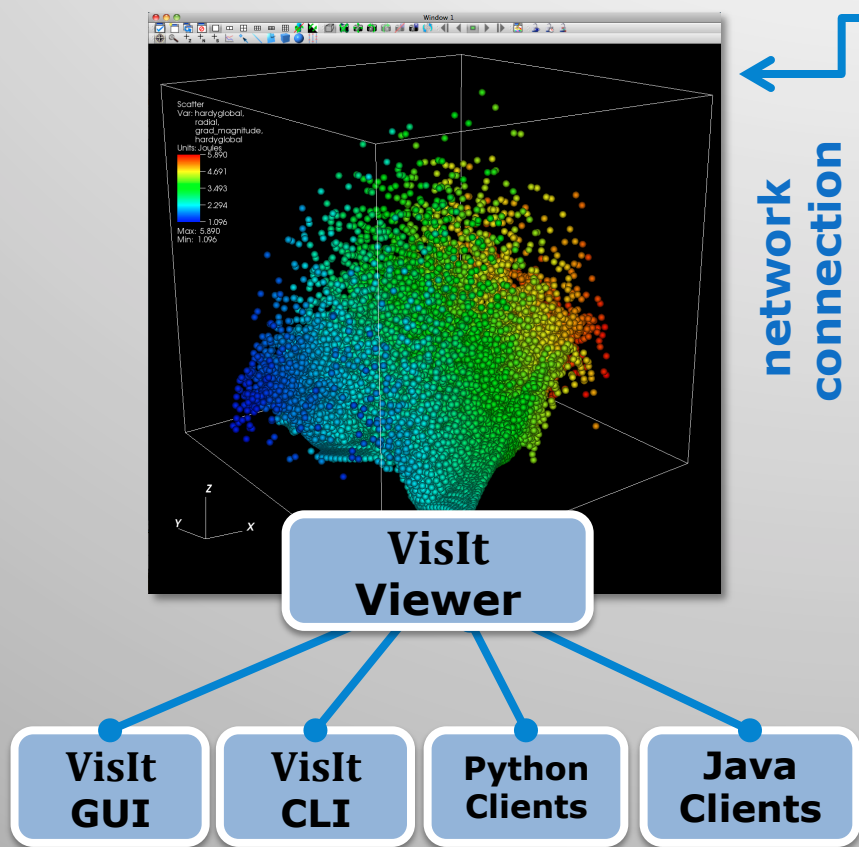


VisIt currently supports over 110 file formats.

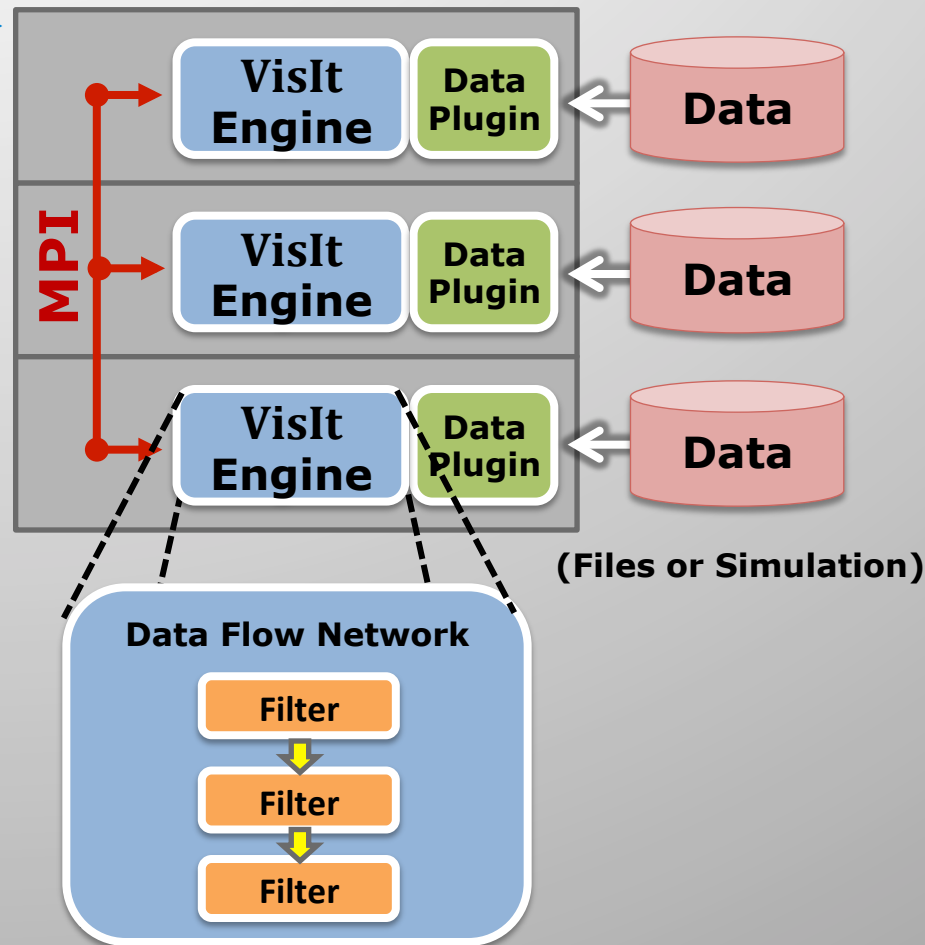


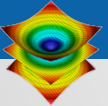
VisIt employs a parallelized client-server architecture.

Local Components

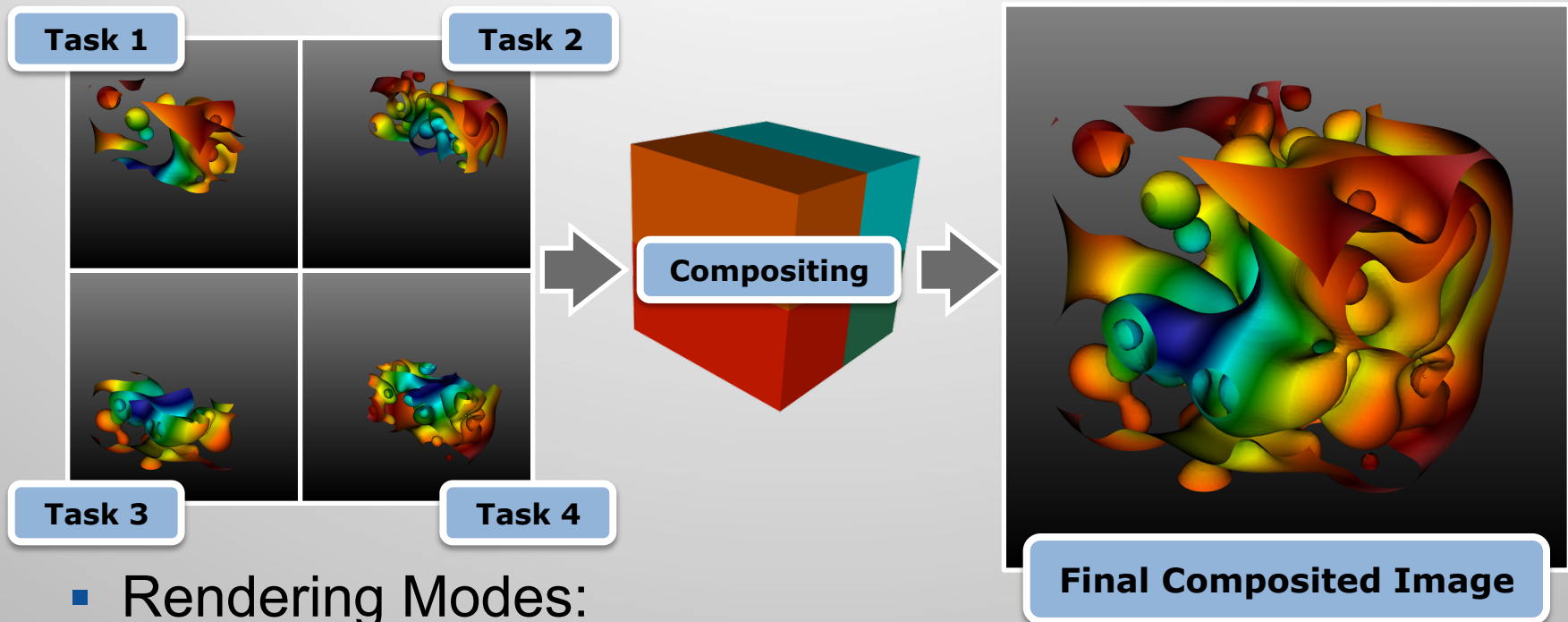


Parallel Cluster

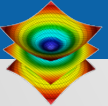




VisIt automatically switches to a scalable rendering mode for large data sets.

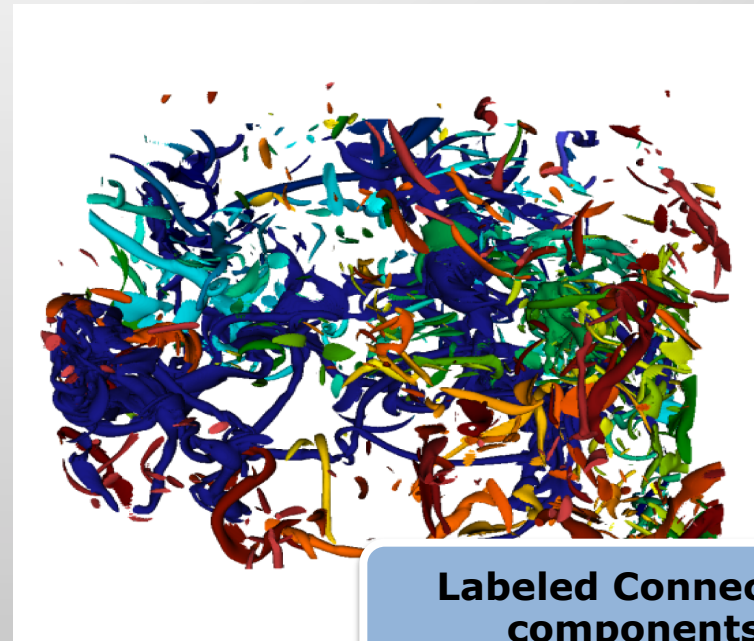
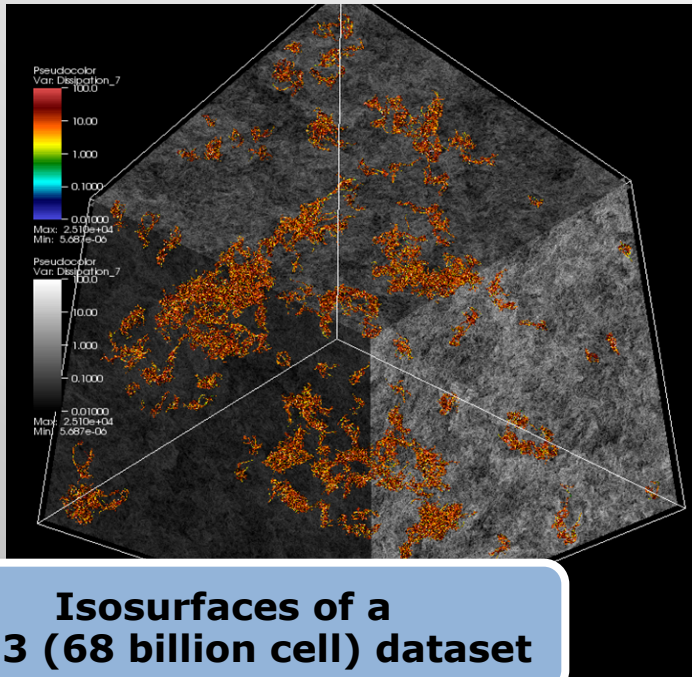


- Rendering Modes:
 - Local (hardware)
 - Remote (software or hardware)
- Beyond surfaces:
 - VisIt also provides scalable volume rendering.



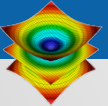
Analysis Example: Evolution of Vorticity

- **Goal:** Identify and track coherent vortical structures in turbulent flow as time evolves.



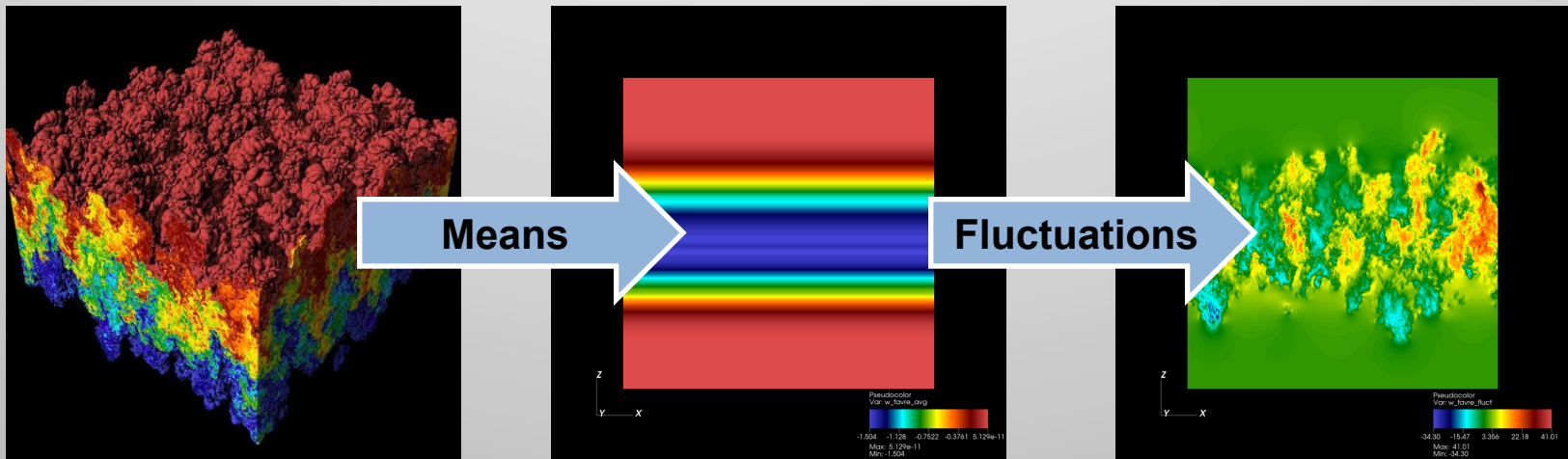
Collaboration with Kelly Gaither, TACC et al (IEEE CG&A July/August 2012)

VisIt was used to calculate isosurfaces, identify connected components, and extract component features.



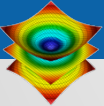
Analysis Example: Turbulence Operators

- **Goal:** Provide one set of turbulence tools that can be used across multiple codes.
- **Application:** Validate RANS model parameters from high fidelity DNS simulations.



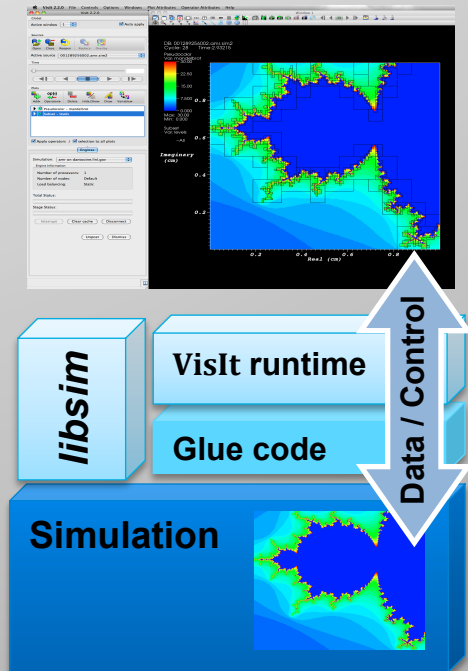
Joint work with Oleg Schilling and Britton Olson, LLNL

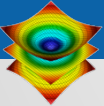
We are developing scripted building blocks for flow analysis, including field means and fluctuations.



VisIt's infrastructure provides a flexible platform for custom workflows.

- C++ Plugin Architecture
 - Custom File formats, Plots, Operators
 - Interface for custom GUIs in Python, C++ and Java
- Python Interfaces
 - Python scripting and batch processing
 - Data analysis via Python Expressions and Queries.
- *Libsim* library
 - Enables coupling of simulation codes to VisIt for in situ visualization.



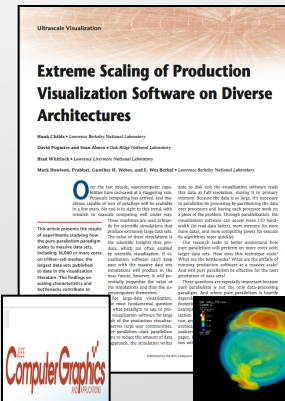


VisIt is used as a platform to deploy visualization research.

Research Collaborations:



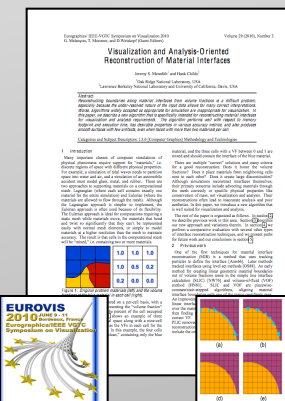
- Research Focus:
 - Next Generation Architectures
 - Parallel Algorithms
 - In-Situ Processing



Scaling research:
Scaling to 10Ks of cores and trillions of cells.



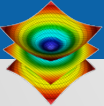
Algorithms research:
How to efficiently calculate particle paths in parallel.



Algorithms research:
Reconstructing material interfaces for visualization



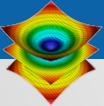
Methods research:
How to incorporate statistics into visualization.



VisIt: What's the Big Deal?

- Everything works at scale
- Robust, usable tool
- Features that span the “power of visualization”:
 - Data Exploration
 - Confirmation
 - Communication
- Features for different kinds of users:
 - Visualization Experts
 - Code Developers
 - Code Consumers

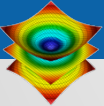
Healthy future: Vibrant Developer and User Communities



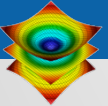
Resources

- **User resources:**
 - Main website: <http://www.llnl.gov/visit>
 - Wiki: <http://www.visitusers.org>
 - Email: visitusers@ornl.gov

- **Development resources:**
 - Email: visit-developers@ornl.gov
 - SVN: <http://portal.nersc.gov/svn/visit>



Hands On Visualizations



30 minute Hands on visualization of a Blood Flow Simulation.

- http://visitusers.org/index.php?title=Blood_Flow_Aneurysm_Tutorial#Tutorial_Setup

Blood Flow Aneurysm Tutorial

visitusers.org/index.php?title=Blood_Flow_Aneurysm_Tutorial#Tutorial_Setup

Cyrus my talk my preferences my watchlist my contributions log out

article discussion edit history delete move watch

Blood Flow Aneurysm Tutorial

This tutorial provides a short introduction to Visit's features while exploring a finite element blood flow simulation of an aneurysm. The simulation was run using the [LifeV] finite element solver and made available for this tutorial thanks to Gilles Fourestey and Jean Favre, [Swiss National Supercomputing Centre]

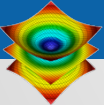
Tutorial Setup

- Visit Installation Instructions
- Tutorial Datasets

Blood Flow Aneurysm Tutorial

- Initial Dataset Exploration
- Visualizing the Velocity Vector Field
- Calculating the Flux Through a Surface

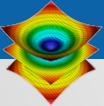
navigation



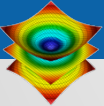
In-depth hands on visualization of a Water Flow Simulation.

Materials for Hand-on session after dinner:

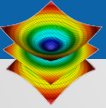
- http://visitusers.org/index.php?title=Water_Flow_Tutorial



[Supporting Slides]

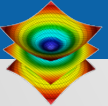


VisIt's Core Abstractions



VisIt's core abstractions

- **Databases:** How datasets are read
- **Plots:** How you render data
- **Operators:** How you manipulate data
- **Expressions:** Mechanism for generating derived quantities
- **Queries:** How to access quantitative information



Examples of VisIt Pipelines

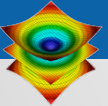
- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

Database

Open a database, which reads from a file (example: open file1.hdf5)

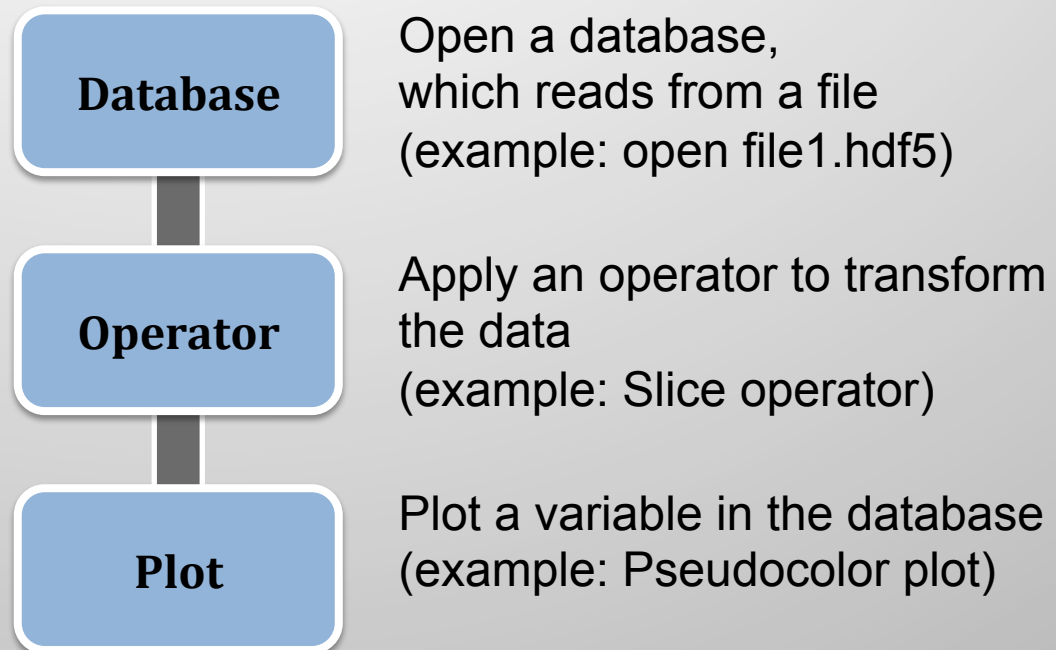
Plot

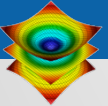
Make a plot of a variable in the database (example: Volume plot)



Examples of VisIt Pipelines

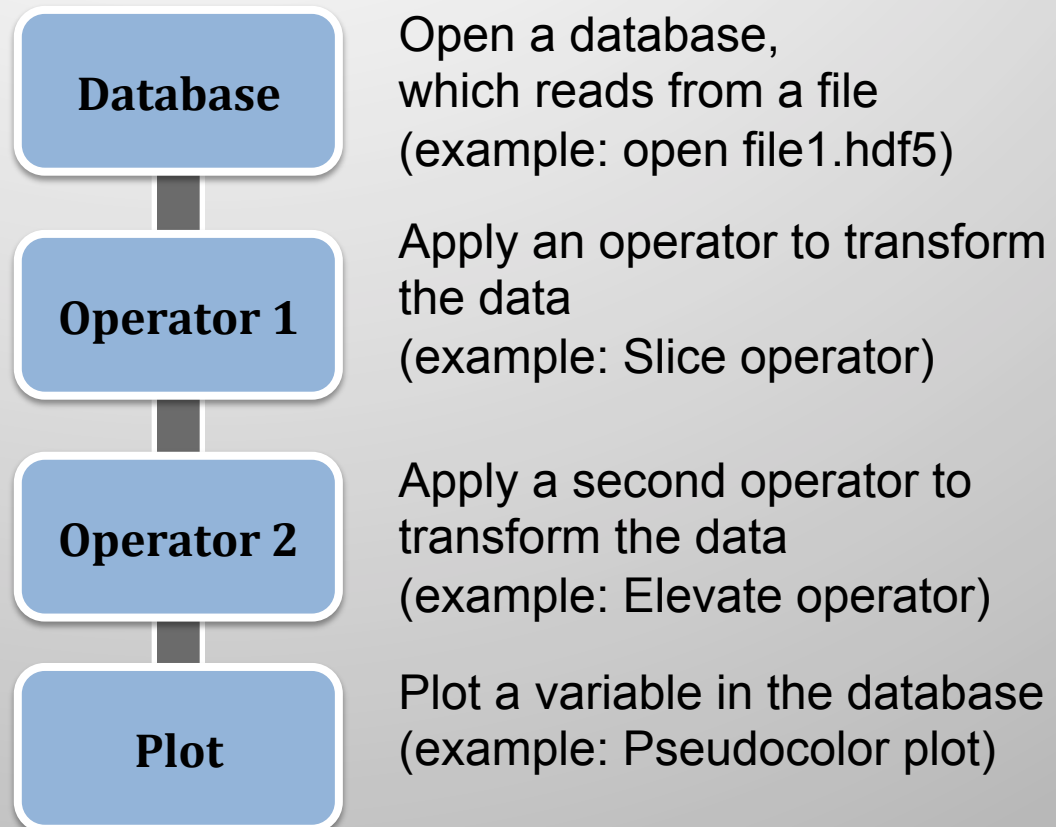
- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

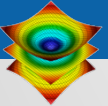




Examples of VisIt Pipelines

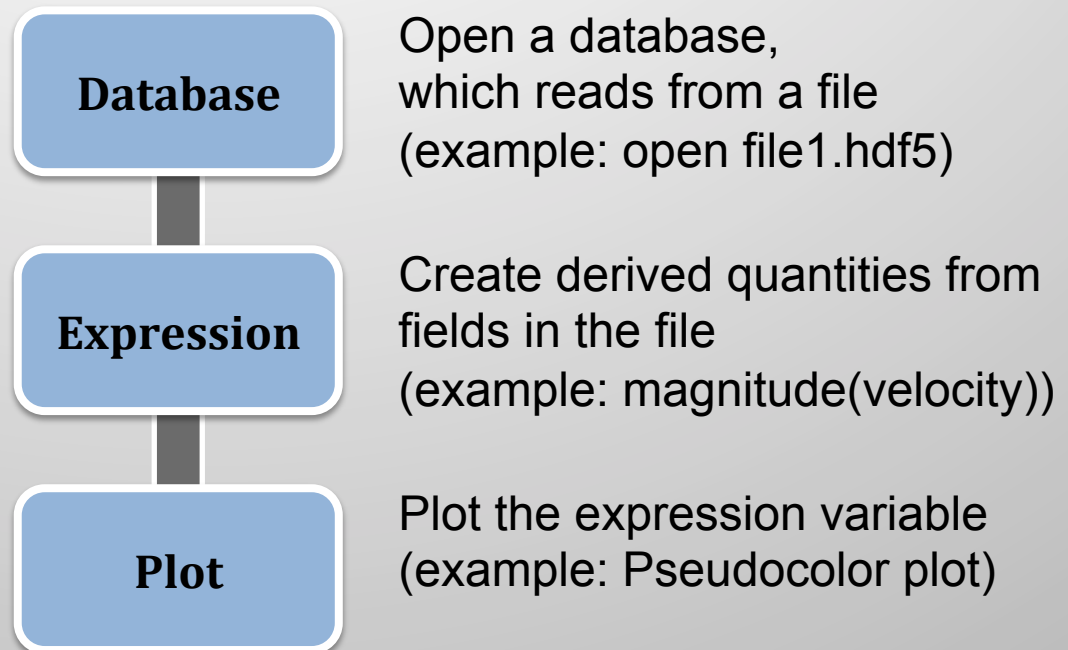
- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

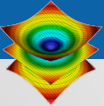




Examples of VisIt Pipelines

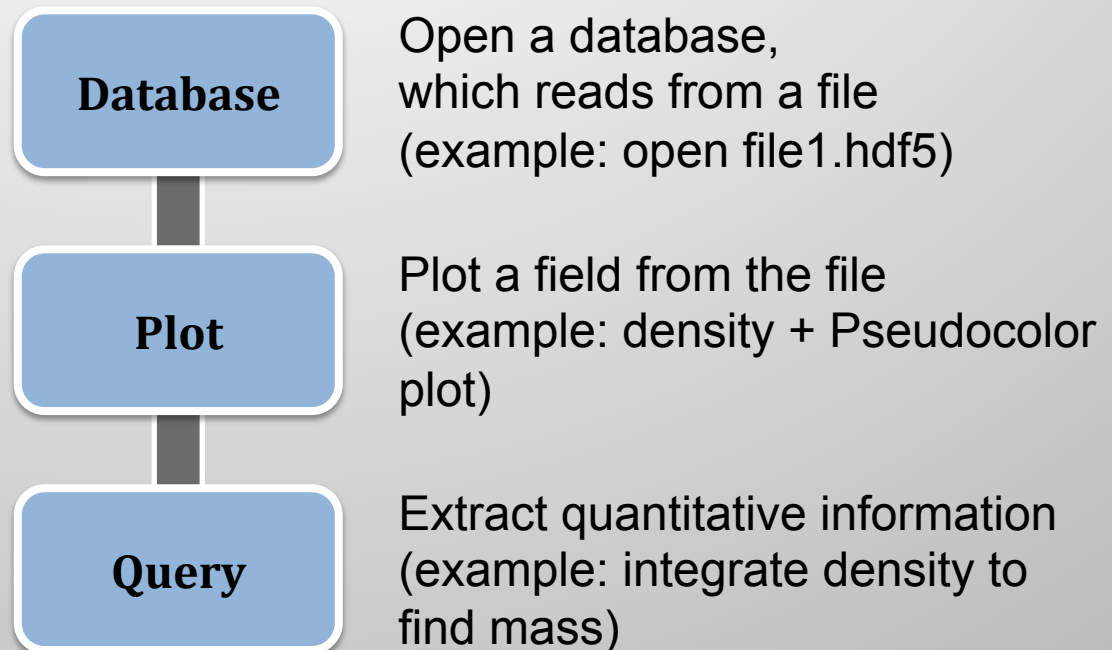
- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

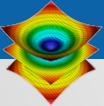




Examples of VisIt Pipelines

- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information





Examples of VisIt Pipelines

- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

Database

Open a database, which reads from a file (example: open file1.hdf5)

Expression

Create derived quantities from fields in the file (example: magnitude(velocity))

Operator 1

Apply an operator to transform the data (example: Slice operator)

Operator 2

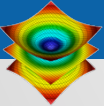
Apply a second operator to transform the data (example: Elevate operator)

Plot

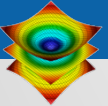
Plot a field (example: speed + Pseudocolor plot)

Query

Extract quantitative information (example: maximum speed over cross-section)

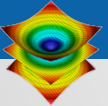


Visualization Techniques for Mesh-based Simulations



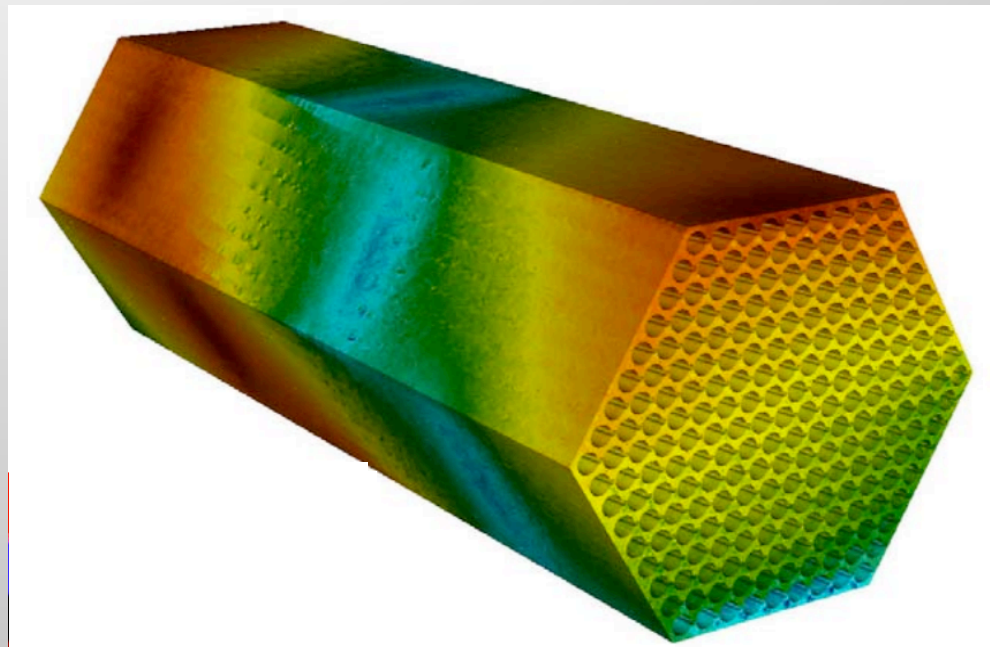
Terminology

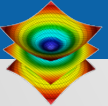
- Meshes: discretization of physical space
 - Contains “zones” / “cells” / “elements”
 - Contains “nodes” / “points” / “vertices”
 - VisIt speak: zone & node
- Fields: variables stored on a mesh
 - Scalar: 1 value per zone/node
 - Example: pressure, density, temperature
 - Vector: 3 values per zone/node (direction)
 - Example: velocity
 - Note: 2 values for 2D, 3 values for 3D
 - More fields discussed later...



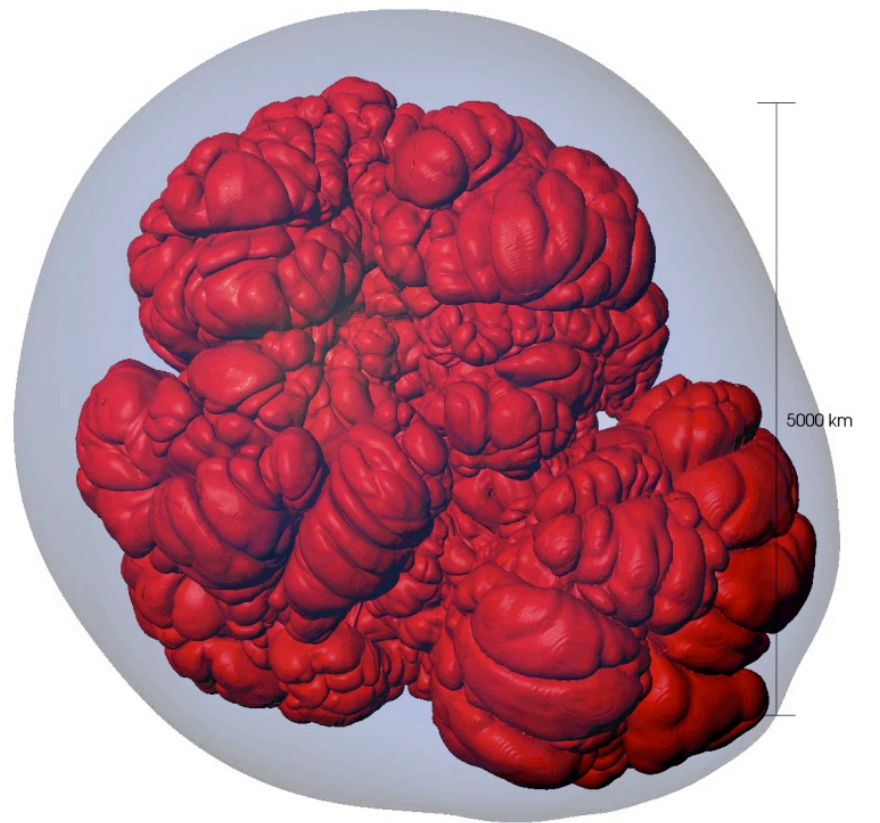
Pseudocolor

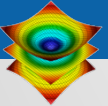
- Maps scalar fields (e.g., density, pressure, temperature) to colors.



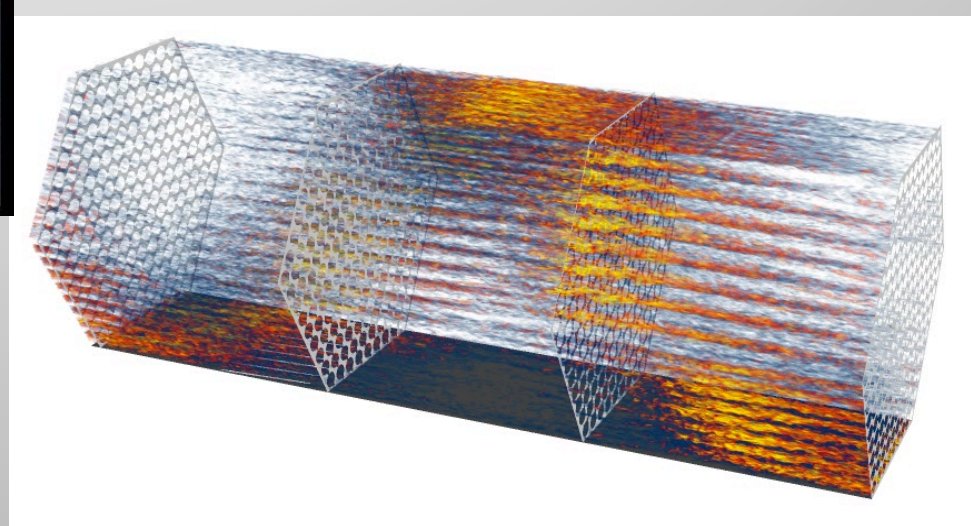
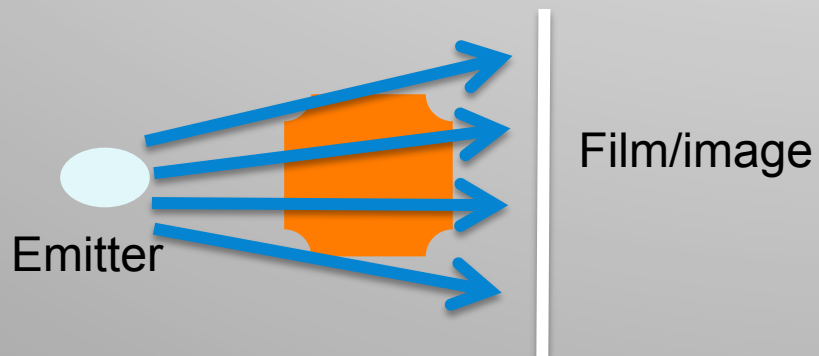
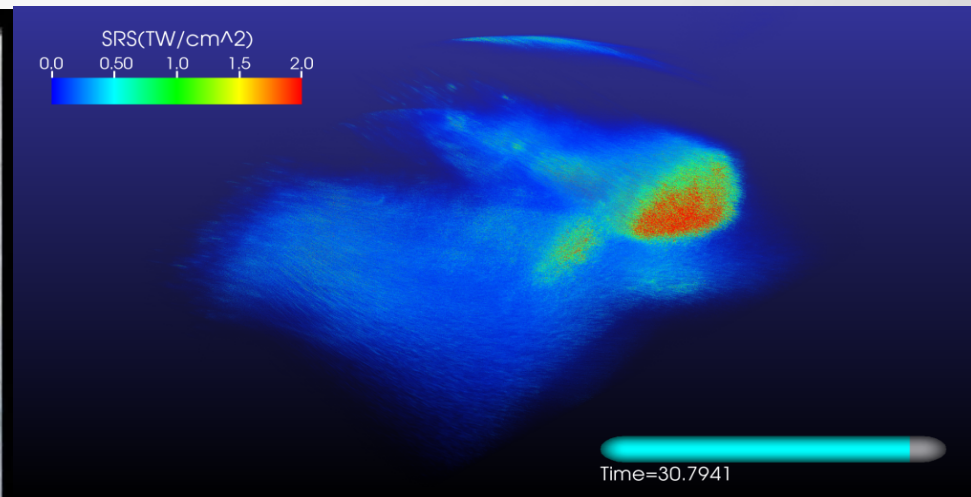


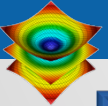
Contour / Isosurface





Volume rendering

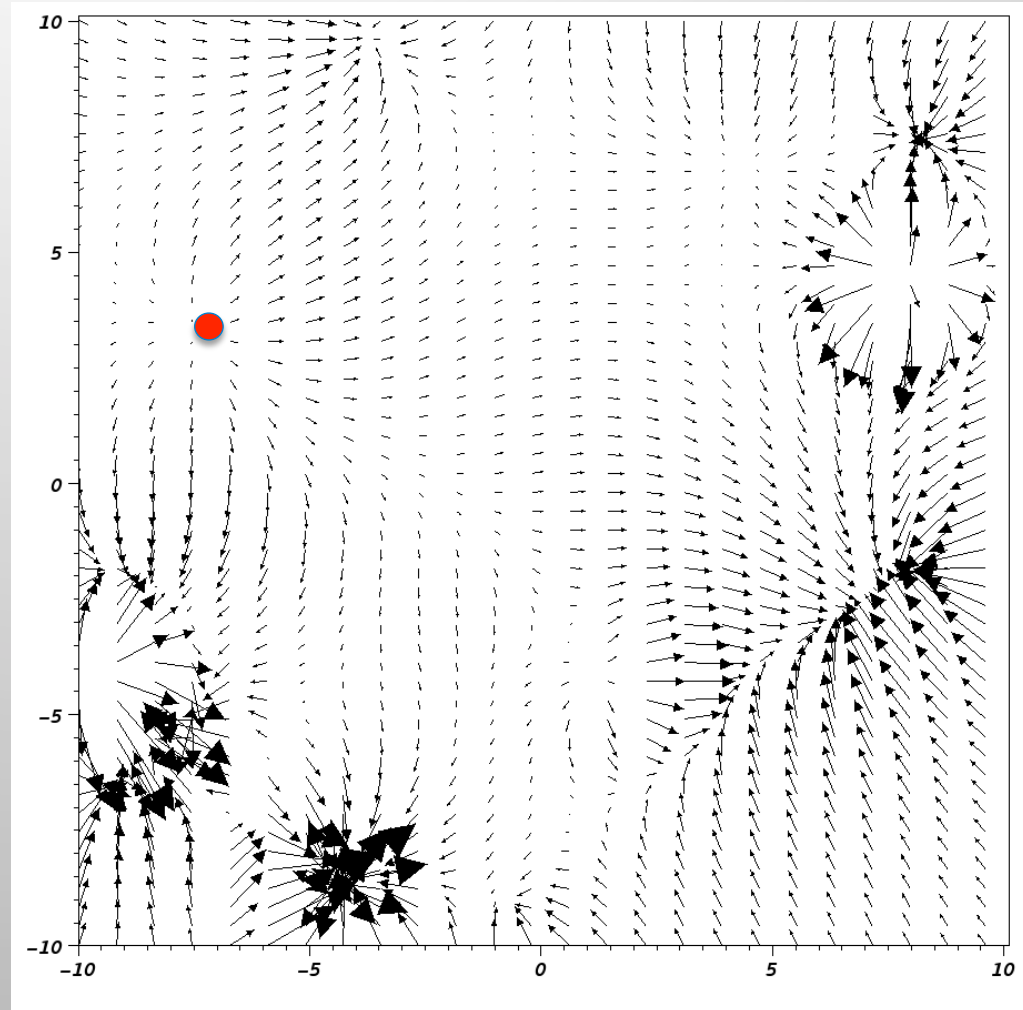


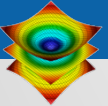


Particle advection: the foundation of flow visualization

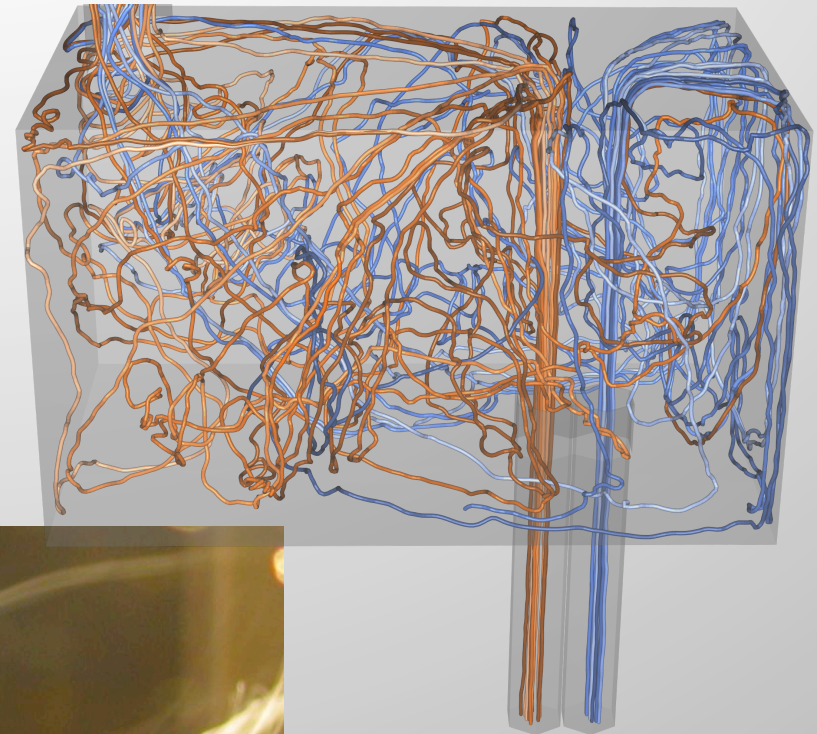
- Displace massless particle based on velocity field
- $S(t)$ = position of curve at time t
 - $S(t_0) = p_0$
 - t_0 : initial time
 - p_0 : initial position
 - $S'(t) = v(t, S(t))$
 - $v(t, p)$: velocity at time t and position p
 - $S'(t)$: derivative of the integral curve at time t

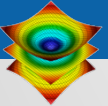
This is an ordinary differential equation





Streamlines

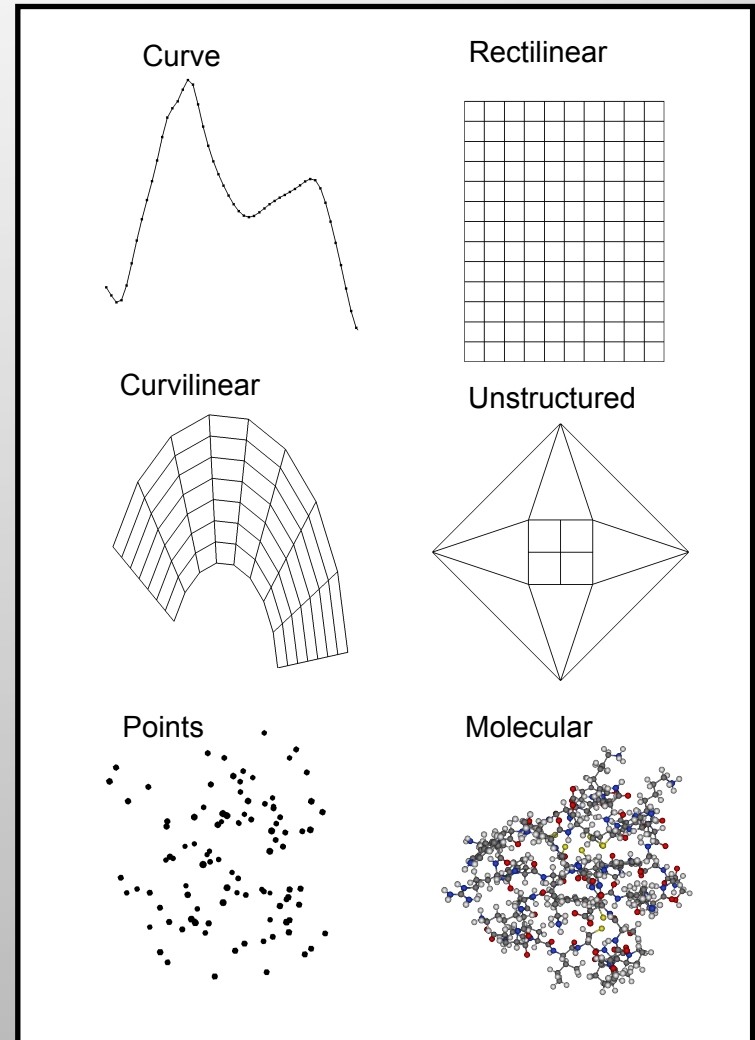


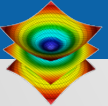


Meshes

- All data in VisIt lives on a mesh
- Discretizes space into points and cells
 - (1D, 2D, 3D) + time
 - Mesh dimension need not match spatial dimension (*e.g. 2D surface in 3D space*)
- Provides a place for data to be located
- Defines how data is interpolated

Mesh Types

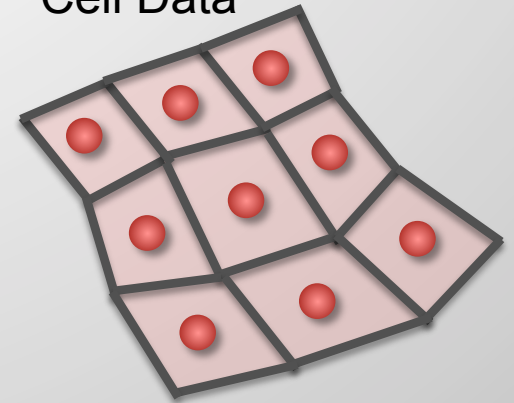




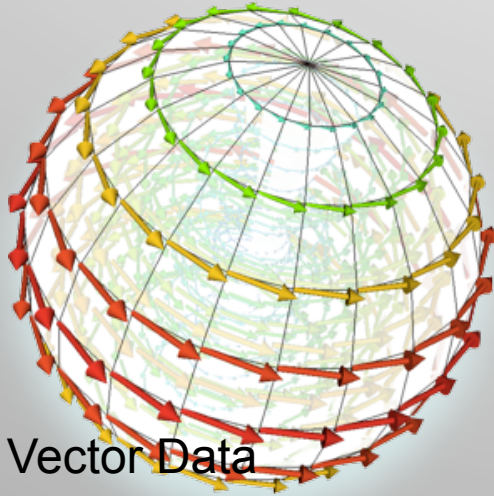
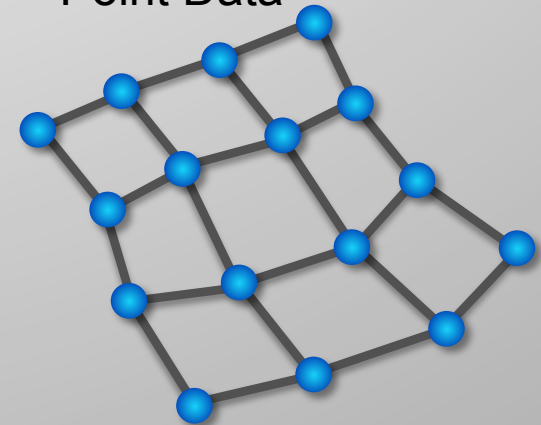
Variables

- Scalars, Vectors, Tensors
- Associated with points or cells of a mesh
 - Points: linear interpolation
 - Cells: piecewise constant
- Can have different dimensionality than the mesh (e.g. 3D vector data on a 2D mesh)

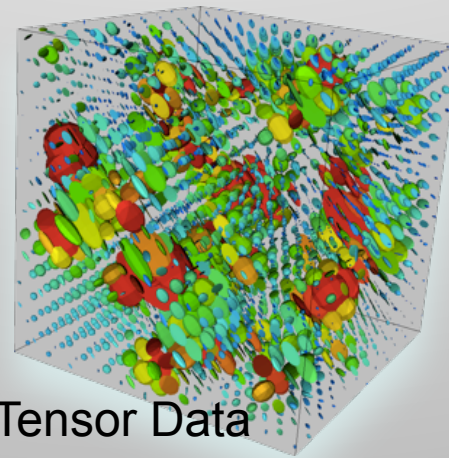
Cell Data



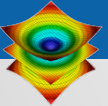
Point Data



Vector Data

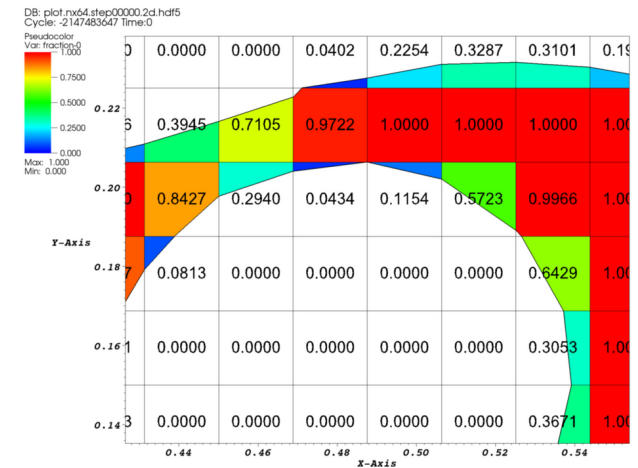


Tensor Data

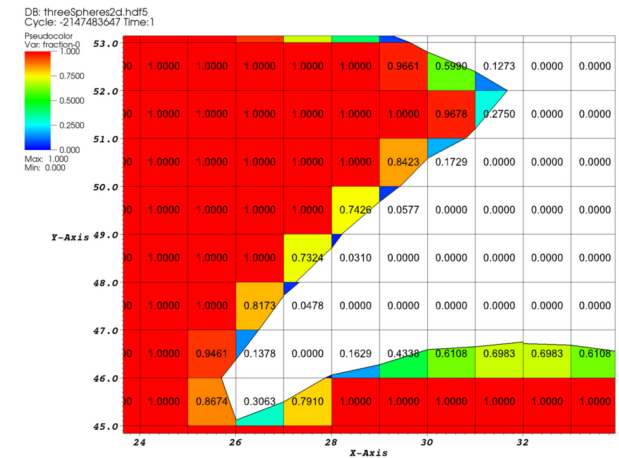


Materials

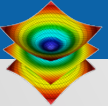
- Describes disjoint spatial regions at a sub-grid level
- Volume/area fractions
- VisIt will do high-quality sub-grid material interface reconstruction



user: ligocki
Thu Apr 23 00:10:11 2009

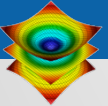


user: ligocki
Thu Apr 23 00:17:29 2009



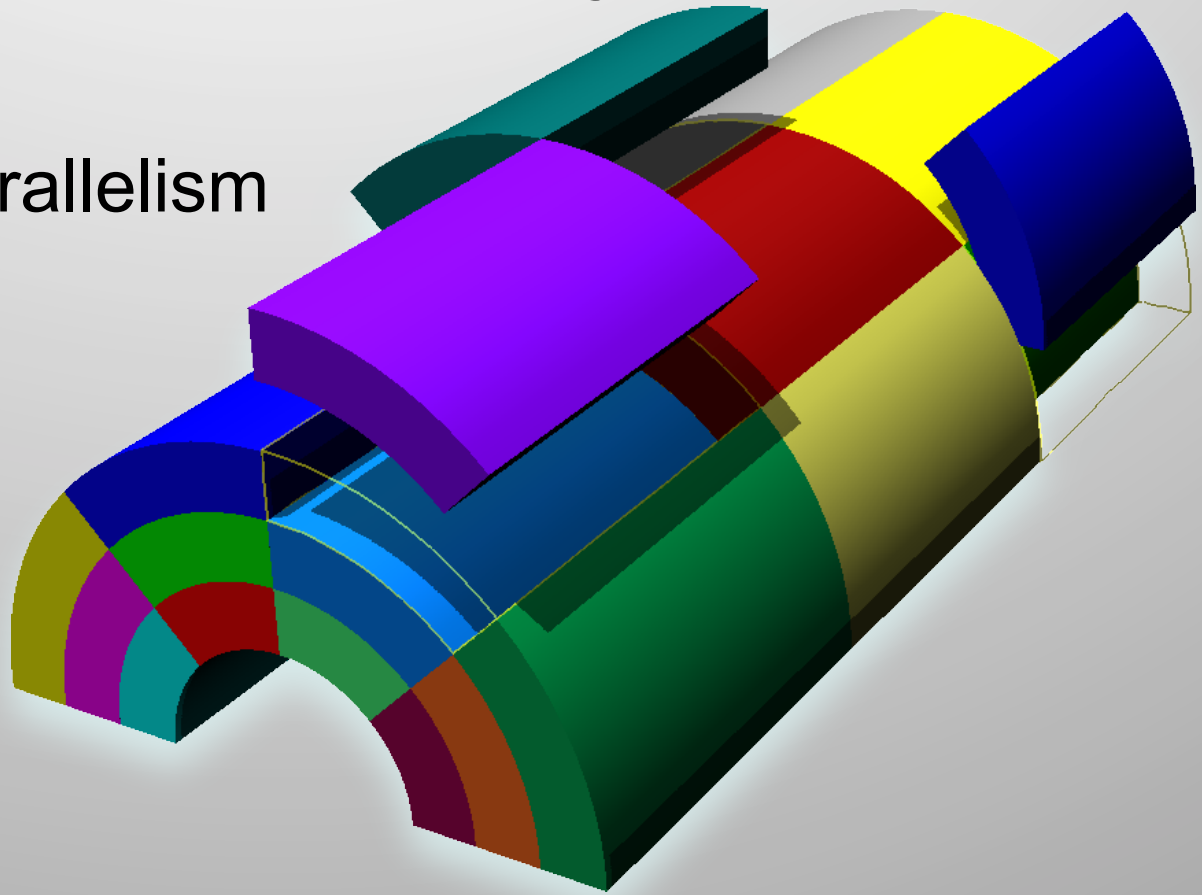
Species

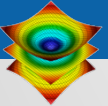
- Similar to materials, describes sub-grid variable composition
 - Example: *Material “Air” is made of species “N₂”, “O₂”, “Ar”, “CO₂”, etc.*
- Used for mass fractions
- Generally used to weight other scalars (e.g. partial pressure)



Parallel Meshes

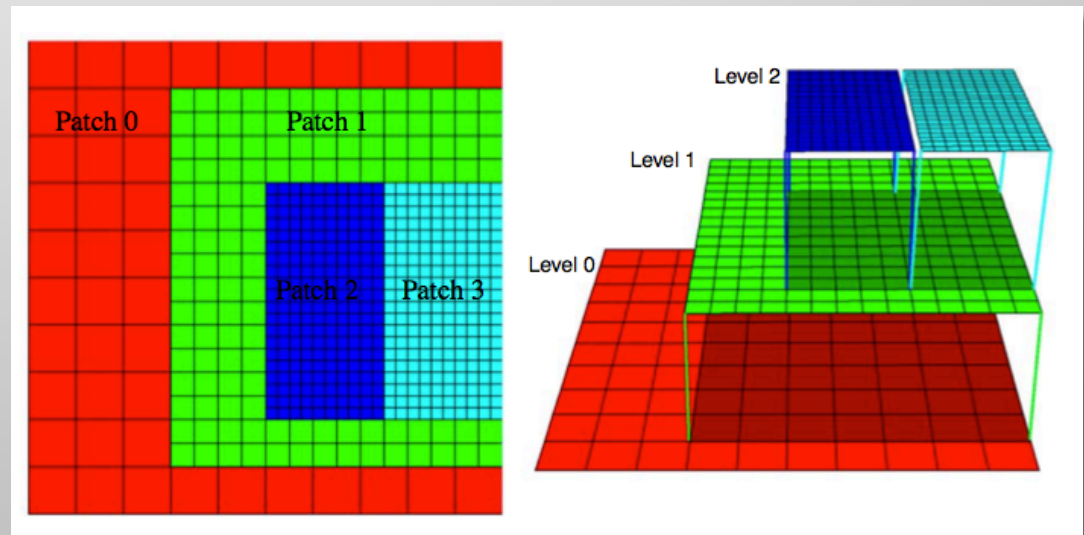
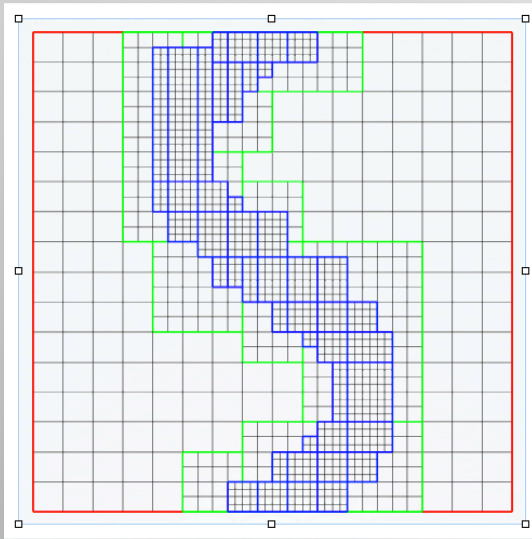
- Provides aggregation for meshes
- A mesh may be composed of large numbers of mesh “blocks”
- Allows data parallelism

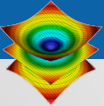




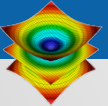
AMR meshes

- Mesh blocks can be associated with patches and levels
- Allows for aggregation of meshes into AMR hierarchy levels



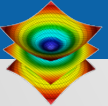


Practical Tips for Using VisIt



Practical Tips for Using VisIt

- **How to get VisIt to read your data**
- How to get help when you run into trouble

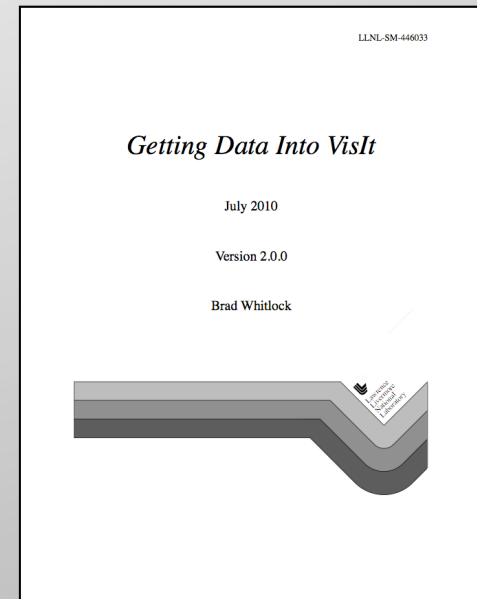


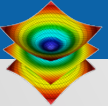
How to get VisIt to read your data.

- There is an extensive manual on this topic: “Getting Data Into VisIt”

<https://wci.llnl.gov/codes/visit/manuals.html>

- Three ways:
 - Use a known format
 - Write a file format reader
 - In situ processing





File formats that VisIt supports

- **110+ Total Readers:** ADIOS, **BOV**, Boxlib, CCM, CGNS, Chombo, CLAW, EnSight, ENZO, Exodus, FLASH, Fluent, GDAL, Gadget, Images (TIFF, PNG, etc), ITAPS/MOAB, LAMMPS, NASTRAN, **NETCDF**, Nek5000, OpenFOAM, PLOT3D, **PlainText**, **Pixie**, Shapefile, **Silo**, Tecplot, **VTK**, **Xdmf**, **Vs**, and many more

[http://www.visitusers.org/index.php?title=Detailed list of file formats VisIt supports](http://www.visitusers.org/index.php?title=Detailed%20list%20of%20file%20formats%20VisIt%20supports)

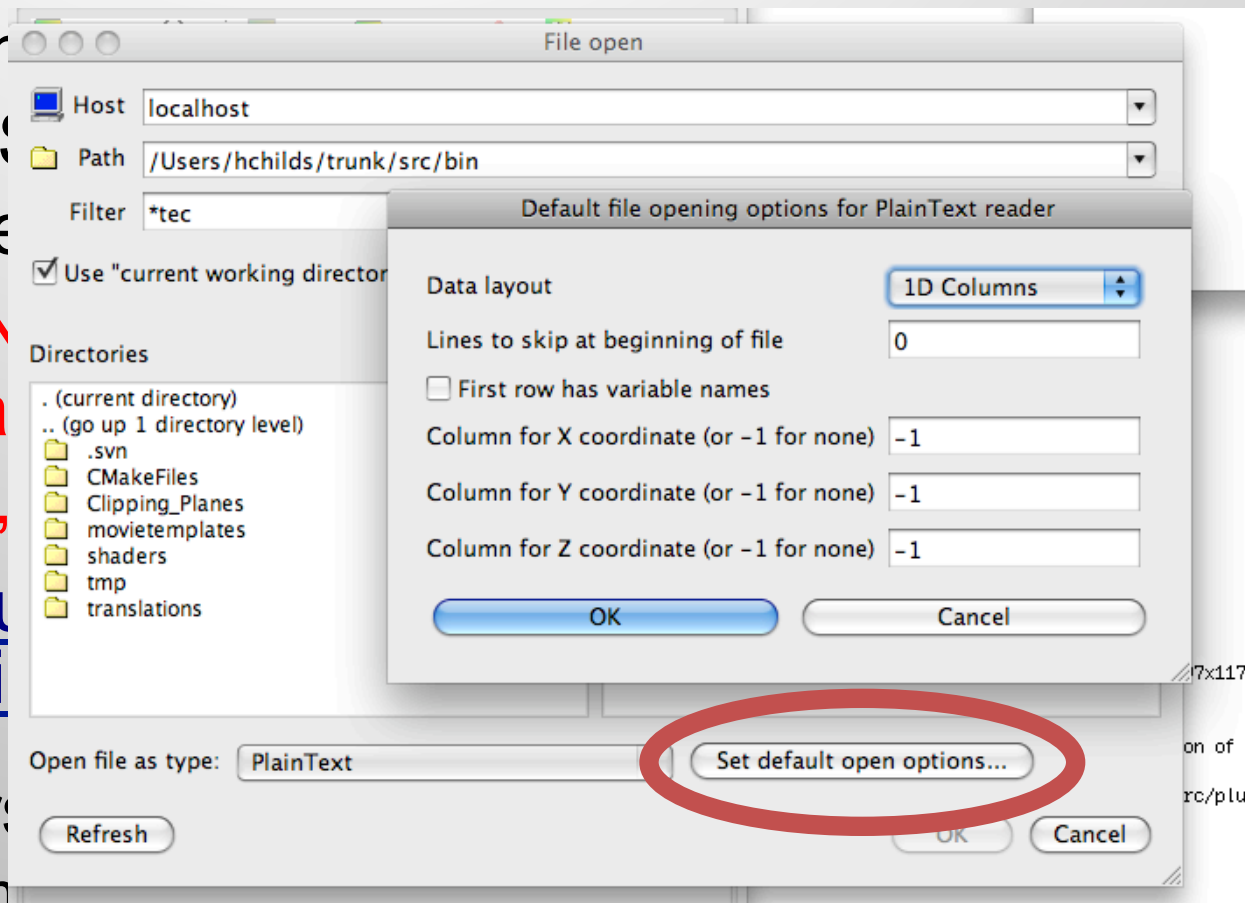
- Some readers are more robust than others.
 - For some formats, support is limited to flavors of a file a VisIt developer has encountered previously (e.g. Tecplot).

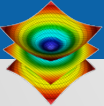
File formats that VisIt supports

- **110+ Total Readers:** ADIOS, **BOV**, Boxlib, CCM, CGNS, Chombo, Exodus, FLACS, Gmsh, Hdf5, Icy, NetCDF, NASTRAN, NEMO, PLOT3D, **Pla**, Tecplot, **VTK**, ...

http://www.visitall.org/wiki/index.php/Supported_File_Formats
 title=Detailed list of supported file formats

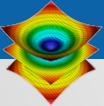
- Some readers may not be supported out of the box, but a VisIt developer has encountered previously (e.g. Tecplot).





File formats that VisIt supports

- Common array writing libraries:
 - NETCDF
 - VisIt reader understands many (but not all) conventions
 - HDF5
 - Pixie is most general HDF5 reader
 - Many other HDF5 readers
- Xdmf: specify an XML file that describes semantics of arrays in HDF5 file
- VizSchema (Vs): add attributes to your HDF5 file that describes semantics of the arrays.



VTK File Format

- The VTK file format has both ASCII and binary variants.
 - Great documentation at:
<http://www.vtk.org/VTK/img/file-formats.pdf>
- Easiest way to write VTK files: use VTK modules
 - ... but this creates a dependence on the VTK library
- You can also try to write them yourself, but this is an error prone process.
- Third option: visit_writer



File Formats

for VTK Version 4.2

(Taken from The VTK User's Guide
Contact Kitware www.kitware.com to purchase)

VTK File Formats

The Visualization Toolkit provides a number of source and writer objects to read and write popular data file formats. The Visualization Toolkit also provides some of its own file formats. The main reason for creating yet another data file format is to offer a consistent data representation scheme for a variety of dataset types, and to provide a simple method to communicate data between software. Whenever possible, we recommend that you use formats that are more widely used. But if this is not possible, the Visualization Toolkit formats described here can be used instead. Note that these formats may not be supported by many other tools.

There are two different styles of file formats available in VTK. The simplest are the legacy, serial formats that are easy to read and write either by hand or programmatically. However, these formats are less flexible than the XML based file formats described later in this section. The XML formats support random access, parallel I/O, and portable data compression and are preferred to the serial VTK file formats whenever possible.

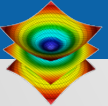
Simple Legacy Formats

The legacy VTK file formats consist of five basic parts.

1. The first part is the file version and identifier. This part contains the single line: `# vtkDataFile Version x.x`. This line must be exactly as shown with the exception of the version number `x.x`, which will vary with different releases of VTK. (Note: the current version number is 3.0. Version 1.0 and 2.0 files are compatible with version 3.0 files.)
2. The second part is the header. The header consists of a character string terminated by end-of-line character `\n`. The header is 256 characters maximum. The header can be used to describe the data and include any other pertinent information.
3. The next part is the file format. The file format describes the type of file, either ASCII or binary. On this line the single word `ASCII` or `BINARY` must appear.
4. The fourth part is the dataset structure. The geometry part describes the geometry and topology of the dataset. This part begins with a line containing the keyword `DATASET` followed by a keyword describing the type of dataset. Then, depending upon the type of dataset, other keyword/data combinations define the actual data.
5. The final part describes the dataset attributes. This part begins with the keywords `POINT_DATA` or `CELL_DATA`, followed by an integer number specifying the number of points or cells, respectively. (It doesn't matter whether `POINT_DATA` or `CELL_DATA` comes first.) Other keyword/data combinations then define the actual dataset attribute values (i.e., scalars, vectors, tensors, normals, texture coordinates, or field data).

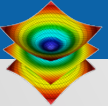
An overview of the file format is shown in Figure 1. The first three parts are mandatory, but the other two are optional. Thus you have the flexibility of mixing and matching dataset attributes and geometry, either by opening system file manipulation or using VTK filters to merge data. Keywords are case insensitive, and may be separated by whitespace. Before describing the data file formats please note the following.

- `dataType` is one of the types `bit`, `unsigned_char`, `char`, `unsigned_short`, `short`, `unsigned_int`, `int`, `unsigned_long`, `long`, `float`, or `double`. These keywords are used to describe the form of the data, both for reading from file, as well as constructing the appropriate internal objects. Not all data types are supported for all classes.



VisIt Writer writes VTK files

- It is a “library” (actually a single C file) that writes VTK-compliant files.
 - The typical path is to link `visit_writer` into your code and write VTK files
- There is also Python binding for `visit_writer`.
 - The typical path is to write a Python program that converts from your format to VTK
- Both options are short term: they allow you to play with VisIt on your data. If you like VisIt, then you typically formulate a long term file format strategy.
- More information on `visit_writer`:
 - <http://visitusers.org/index.php?title=VisItWriter>



Python VisIt Writer in action

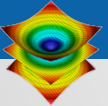
```
import visit_writer
import math
import sys

nX = 20
nY = 20
conn = []
for i in range(nX-1):
    for j in range(nY-1):
        pt1 = j*(nX) + i;
        pt2 = j*(nX) + i+1;
        pt3 = (j+1)*(nX) + i+1;
        pt4 = (j+1)*(nX) + i;
        conn.append([ "quad", pt1, pt2, pt3, pt4 ])

pts = []
rad = []
for i in range(nX):
    for j in range(nY):
        pts.extend([ float(i), float(j), 0 ])
        rad.append( math.sqrt(i*i + j*j) )

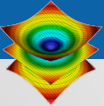
var_datum = [ "radius", 1, 1, rad ]
vars = [ var_datum ]
visit_writer.WriteUnstructuredMesh("ugrid.vtk", 0, pts, conn, vars)

sys.exit()
```



Silo file format

- Silo is a mature, self-describing file format that deals with multi-block data.
- It has drivers on top of HDF5 and “PDB”.
- Fairly rich data model
- More information:
 - <https://wci.llnl.gov/codes/silo/>



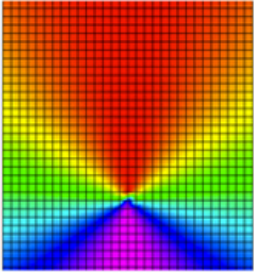
Silo features

WCI | B Codes - SILO

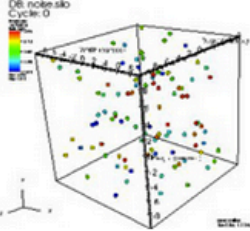
<https://wci.llnl.gov/codes/silo/>

Welcome to Silo

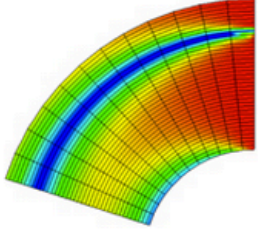
A mesh and field I/O library and scientific database



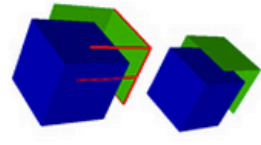
Structured Rectilinear Mesh



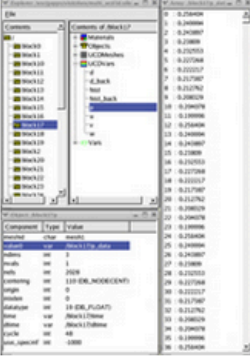
Gridless Point Mesh




Structured (Curvilinear) Mesh



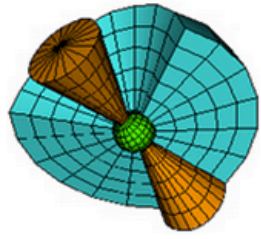
Arbitrary Subsets



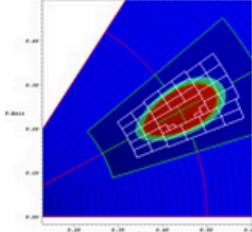
Silex browser for Silo files



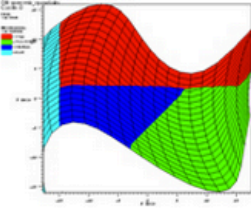
Constructive Solid Geometry (CSG) Mesh



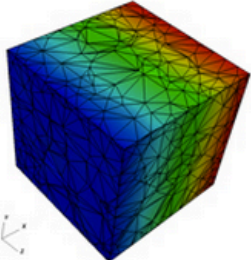
Unstructured Zoo (UCD) Mesh



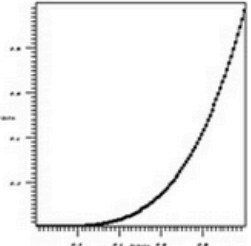
Adaptive Mesh Refinement (AMR) Mesh



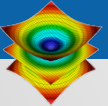
Mixing Materials



Arbitrary Polyhedral Mesh



XY Curve

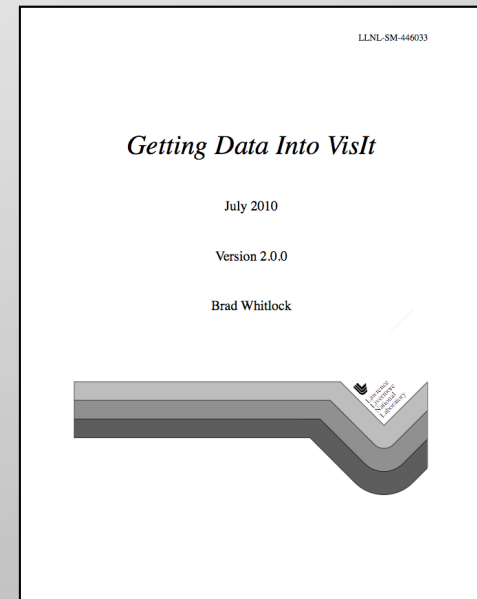


How to get VisIt to read your data.

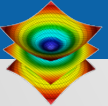
- There is an extensive manual on this topic: “Getting Data Into VisIt”

<https://wci.llnl.gov/codes/visit/manuals.html>

- Three ways:
 - Use a known format
 - Write a file format reader
 - In-situ processing

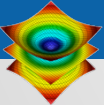


These topics are covered in the manual



Practical Tips for Using VisIt

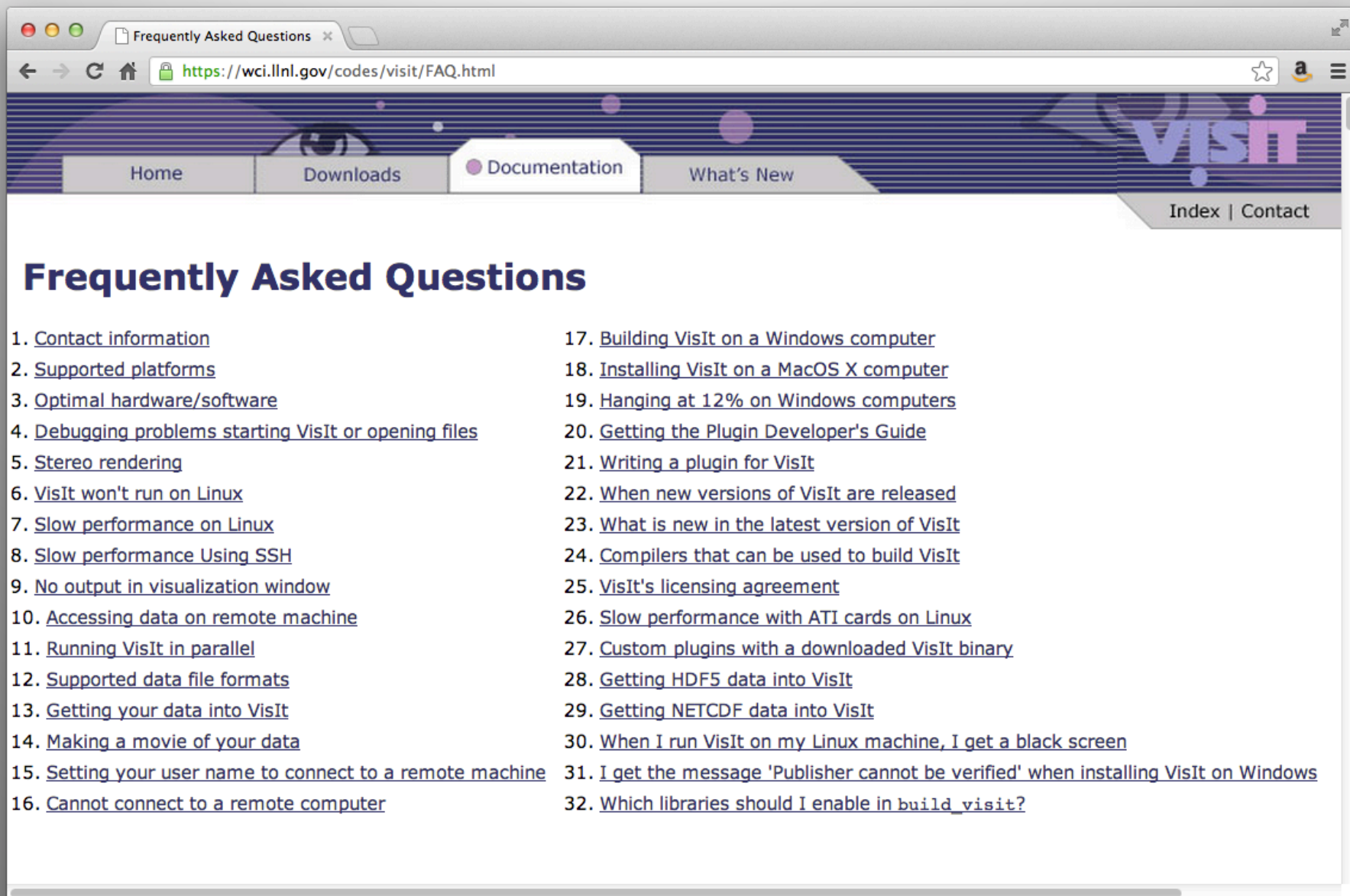
- How to get VisIt to read your data
- **How to get help when you run into trouble**



How to get help when you run into trouble

- FAQ
 - <http://visit.llnl.gov/FAQ.html>
- VisIt Users Mailing List
 - Address: visit-users@elist.ornl.gov
 - Info: <https://elist.ornl.gov/mailman/listinfo/visit-users>
 - Archive: <https://elist.ornl.gov/pipermail/visit-users/>
- VisIt Users Wiki
 - <http://www.visitusers.org>
- VisIt Users Forum
 - <http://visitusers.org/forum/YaBB.pl>
- Priority support for specific user groups:
 - VisIt-help-{XYZ} Mailing Lists
- Reference Manuals
 - <https://wci.llnl.gov/codes/visit/manuals.html>

FAQ: <http://visit.llnl.gov/FAQ.html>

A screenshot of a web browser displaying the VisIt Frequently Asked Questions page. The browser's address bar shows the URL https://wci.llnl.gov/codes/visit/FAQ.html. The page has a dark blue header with the VisIt logo on the right and navigation links (Home, Downloads, Documentation, What's New) in the center. Below the header, there are links for Index and Contact. The main content area is titled 'Frequently Asked Questions' and contains a list of 32 numbered links to various FAQ topics, arranged in two columns. The links cover topics such as contact information, supported platforms, hardware/software, debugging, rendering, performance, installation on different operating systems, building VisIt, and using data formats like HDF5 and NETCDF.

Frequently Asked Questions

1. [Contact information](#)

2. [Supported platforms](#)

3. [Optimal hardware/software](#)

4. [Debugging problems starting VisIt or opening files](#)

5. [Stereo rendering](#)

6. [VisIt won't run on Linux](#)

7. [Slow performance on Linux](#)

8. [Slow performance Using SSH](#)

9. [No output in visualization window](#)

10. [Accessing data on remote machine](#)

11. [Running VisIt in parallel](#)

12. [Supported data file formats](#)

13. [Getting your data into VisIt](#)

14. [Making a movie of your data](#)

15. [Setting your user name to connect to a remote machine](#)

16. [Cannot connect to a remote computer](#)

17. [Building VisIt on a Windows computer](#)

18. [Installing VisIt on a MacOS X computer](#)

19. [Hanging at 12% on Windows computers](#)

20. [Getting the Plugin Developer's Guide](#)

21. [Writing a plugin for VisIt](#)

22. [When new versions of VisIt are released](#)

23. [What is new in the latest version of VisIt](#)

24. [Compilers that can be used to build VisIt](#)

25. [VisIt's licensing agreement](#)

26. [Slow performance with ATI cards on Linux](#)

27. [Custom plugins with a downloaded VisIt binary](#)

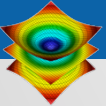
28. [Getting HDF5 data into VisIt](#)

29. [Getting NETCDF data into VisIt](#)

30. [When I run VisIt on my Linux machine, I get a black screen](#)

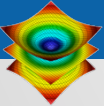
31. [I get the message 'Publisher cannot be verified' when installing VisIt on Windows](#)

32. [Which libraries should I enable in build_visit?](#)



Visit-users Mailing List

- You may only post to mailing list if you are also a subscriber.
- Approximately 400 recipients, approx. 300 posts per month.
- Developers monitor mailing list, strive for 100% response rate.
- Response time is typically excellent ($O(1)$ hour).
 - International community participates ... not unusual for a question from Australia to be answered by a European, while all US developers are asleep.
- List Address: visit-users@ornl.gov
- More information: <https://email.ornl.gov/mailman/listinfo/visit-users>
- Archive: <https://email.ornl.gov/pipermail/visit-users/>



VisItusers.org

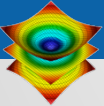
- Great source for VisIt tips and recipes.
- Users section has lots of practical advice:
 - “I solved this problem using this technique”
 - “Here’s my script to do this analysis”

Misc

[\[edit\]](#)

- [Using VisIt in an mxterm](#)
- [Using derived data functions \(DDFs\)](#)
- [Using the command line interface](#)
- [How volume rendering works in VisIt](#)
- [Using cross-mesh field evaluations ... how to do differences, access other time slices, etc](#)
- [Keyframing example](#)
- [Exporting databases](#)
- [Directions for specific machines](#)
- [Using the VisIt Python API with a standard Python interpreter](#)
- [Pages that contain instructions specific to certain user groups and needs](#)
- [Issues related to running VisIt on Windows under cygwin](#)
- [VisIt's Camera model](#)
- [Using VisIt's mpeg2encode](#)
- [Molecular data features](#)
- [Extracting alpha](#)
- [\(Very\) High resolution rendering](#)
- [Elevating shapefiles](#)
- [Raytracing your visualizations with POV-Ray and a tutorial POV-Ray exporting example](#)

VisItusers.org is the VisIt project's staging area for usage recipes and future formal documentation.



VisIt Users Forum

- <http://www.visitusers.org/forum>
- Increasingly popular option; you can post without receiving 300 emails a month
 - But it is viewed by less people and less well supported.
- Google indexes these pages.

Members viewing this topic (1): **Hank Childs**.

pseudocolor plot legend attributes in python (Read 18 times)

Jennifer
YaBB Newbies
★
Offline



Posts: 4
Fort Collins, CO

pseudocolor plot legend attributes in python
11/07/10 at 19:06:30

Hello. I want to set the attributes for a pseudocolor plot legend (turn off Let VisIt manage location of the legend, number of Tic Marks, and the label appearance (I set these properties in a python script? If so, how?

I tried to use the Command Control to record the
"# Logging for AddAnnotationObject is not implemented
Logging for SetAnnotationObjectOptions is not implemented

Thanks,
Jennifer

Back to top

Hank Childs
YaBB Moderator
★★★★★
Online



I use VisIt and I develop VisIt

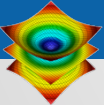
Posts: 135
Davis, CA

Re: pseudocolor plot legend attributes in python
Reply #1 - 11/07/10 at 19:47:03

Hello Jennifer,

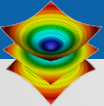
Each plot has an index and the plot's legend is re

```
>>> GetAnnotationObjectNames()
('Plot0003',)
>>> a = GetAnnotationObject("Plot0003")
>>> a
active = 1
managePosition = 1
position = (0.05, 0.9)
xScale = 1
yScale = 1
```

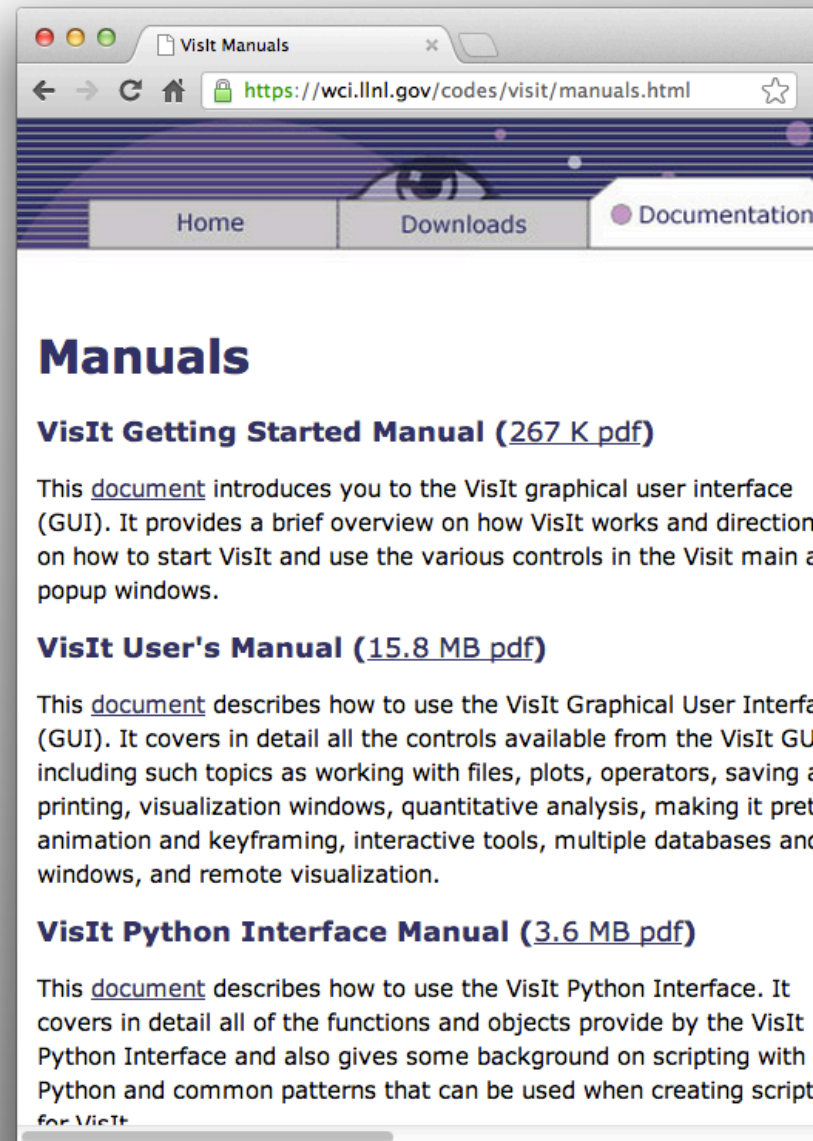
Visit-help-{XYZ}

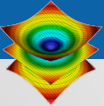
- Some customer groups pay for priority VisIt support:
 - These customers can post directly to specific visit-help-{XYZ} support lists without subscribing.
 - The messages are received by all VisIt developers and supported collectively.
- Current Lists:
 - visit-help-asc, visit-help-scidac, visit-help-gnep, visit-help-ascem



Manuals & Other Documentation

- Getting Started Manual
- Users Manual
- Python Interface
- Getting Data Into VisIt
- VisIt Class Slides
- VisIt Class Exercises
- {Tutorials}





Resources

- **Presenters:**

- Cyrus Harrison cyrush@llnl.gov

- **User resources:**

- Main website: <http://www.llnl.gov/visit>
- Wiki: <http://www.visitusers.org>
- Email: visitusers@ornl.gov

- **Development resources:**

- Email: visit-developers@ornl.gov
- SVN: <http://portal.nersc.gov/svn/visit>