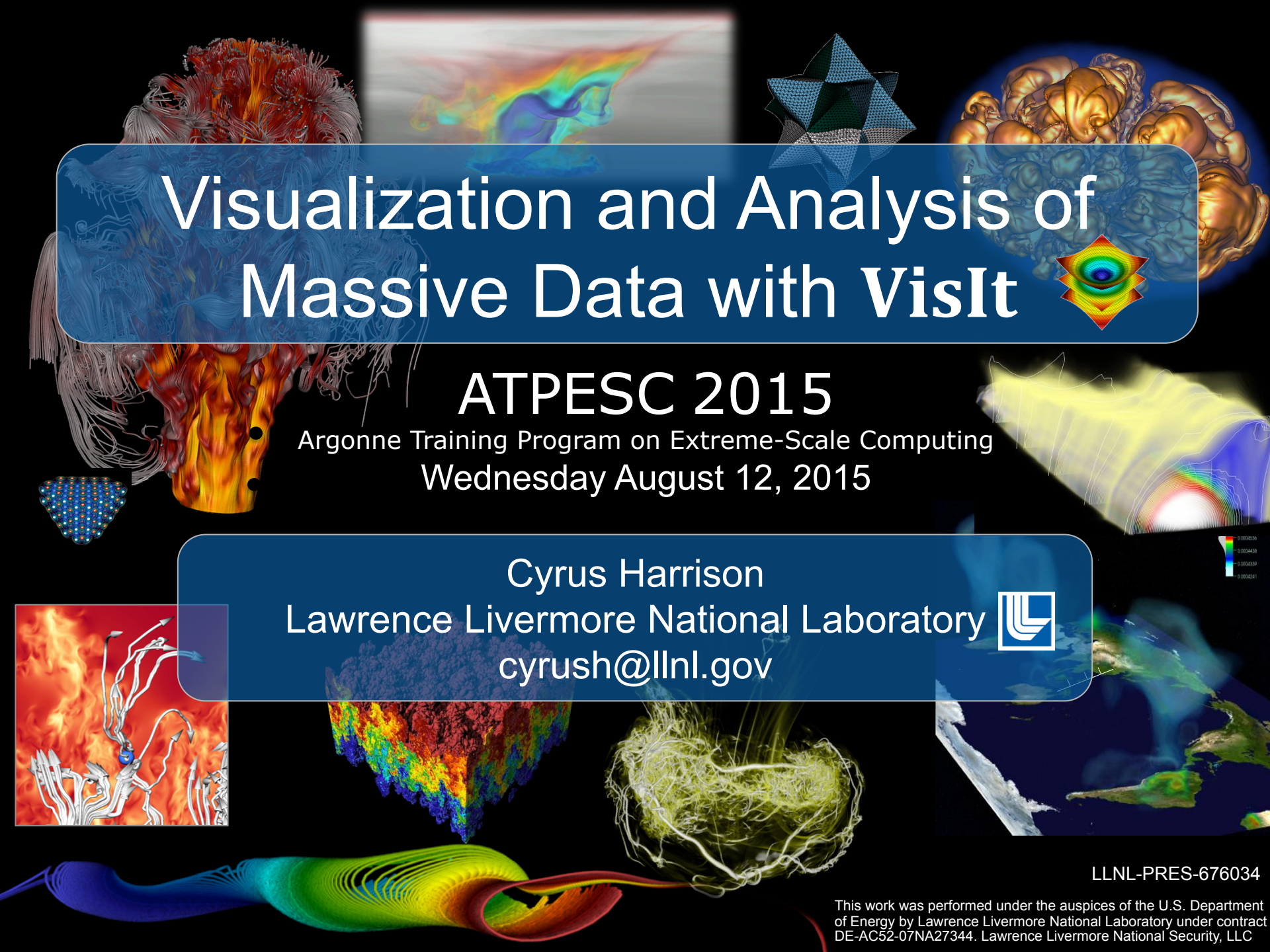


Visualization and Analysis of Massive Data with VisIt

ATPESC 2015

Argonne Training Program on Extreme-Scale Computing

Wednesday August 12, 2015

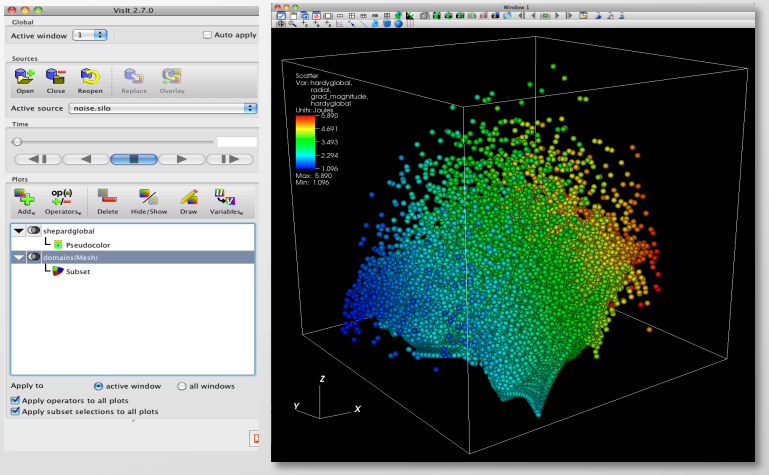
Cyrus Harrison
Lawrence Livermore National Laboratory
cyrush@llnl.gov

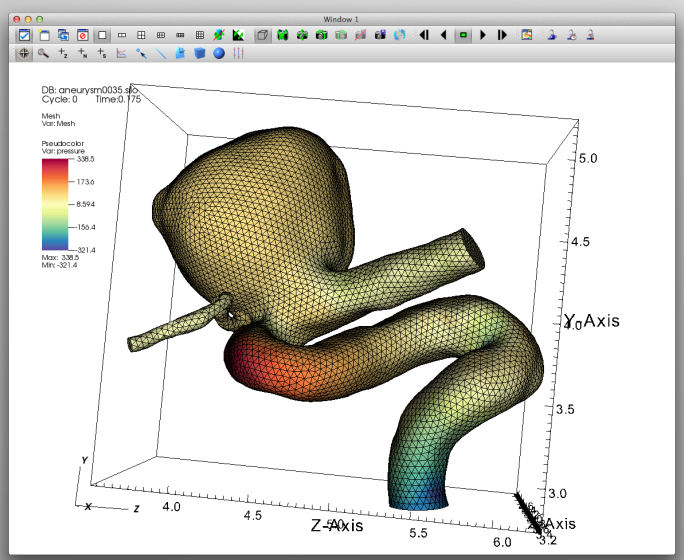
LLNL-PRES-676034

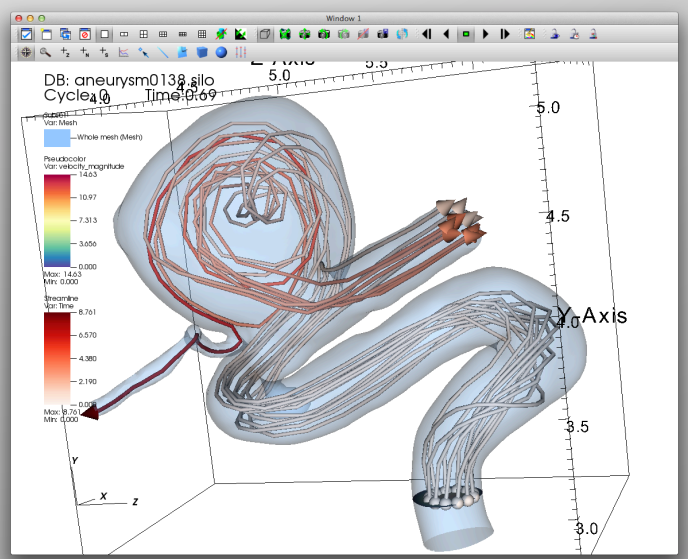
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Tutorial Outline

- **VisIt Project Intro [10 min]**
- **Guided tour of VisIt [20 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 Tutorials):**

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

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

Tutorial Data Acknowledgements

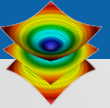
Aneurysm Simulation Data

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

Available thanks to:

Gilles Fourestey and Jean Favre
Swiss National Supercomputing Centre

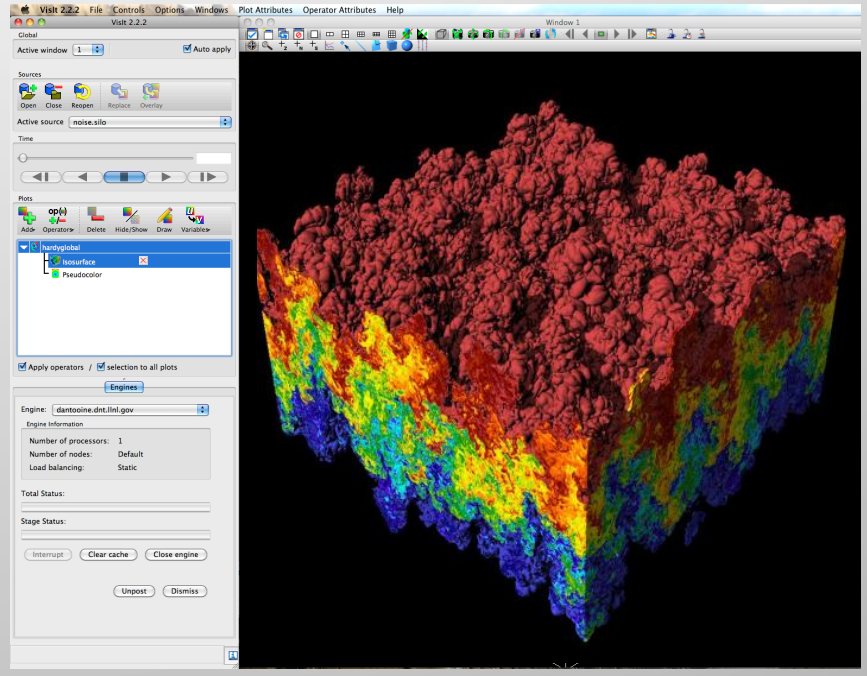
http://www.cscs.ch/



VisIt Project Introduction

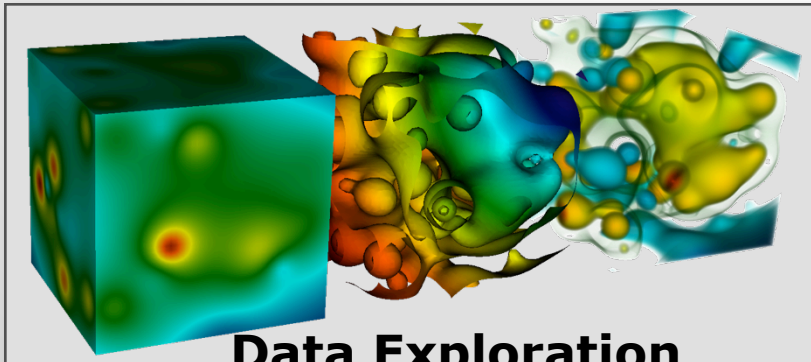
VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data.

- 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.

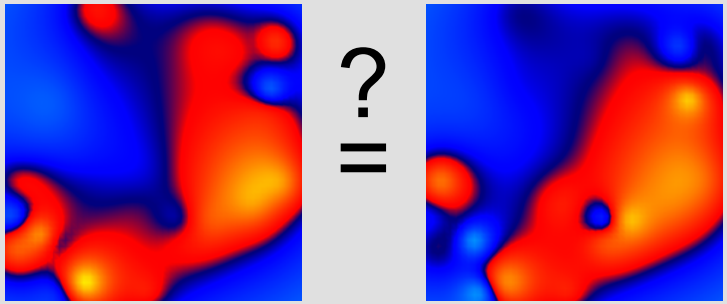


Density Isovolume of a 3K^3 (27 billion cell) dataset

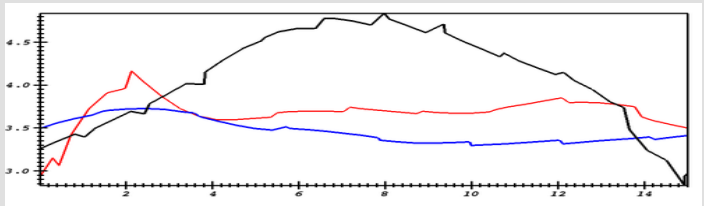
VisIt supports a wide range of use cases.



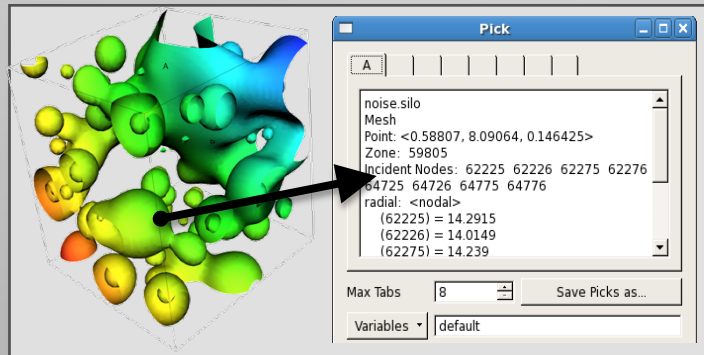
Data Exploration



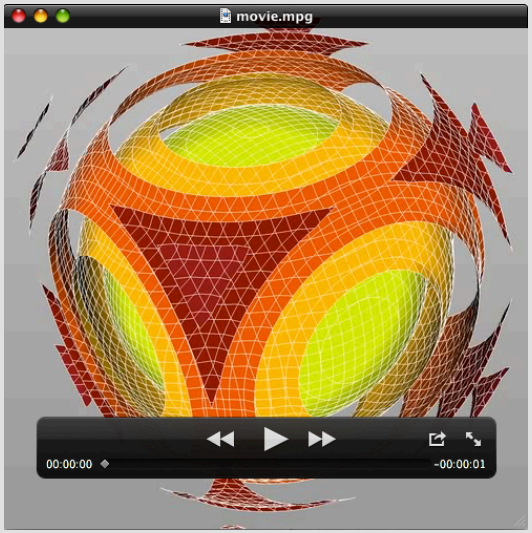
Comparative Analysis



Quantitative Analysis

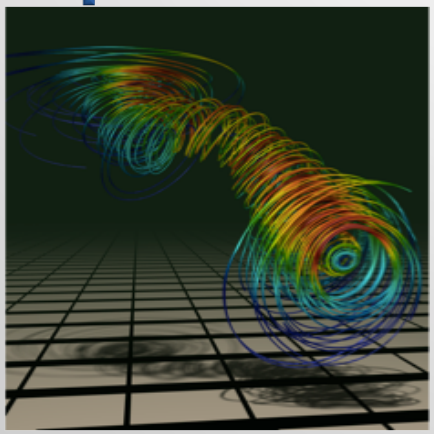


Visual Debugging

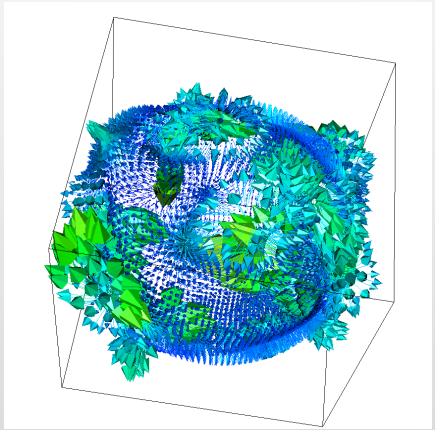


Presentation Graphics

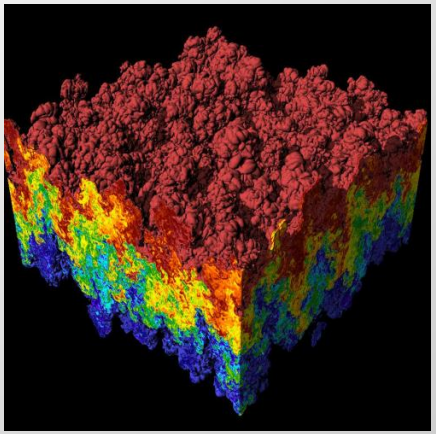
Examples of VisIt's visualization capabilities.



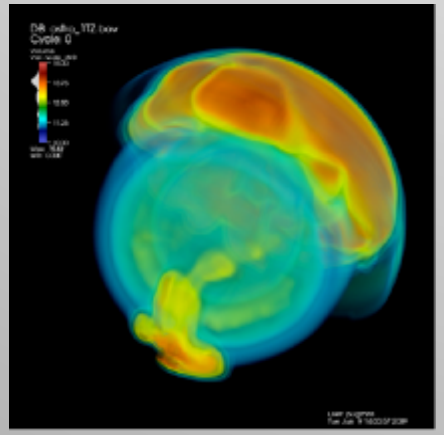
Streamlines



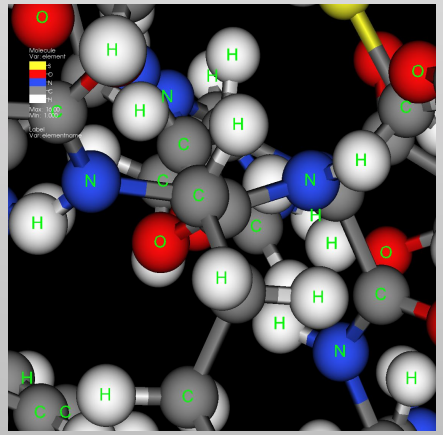
Vector / Tensor Glyphs



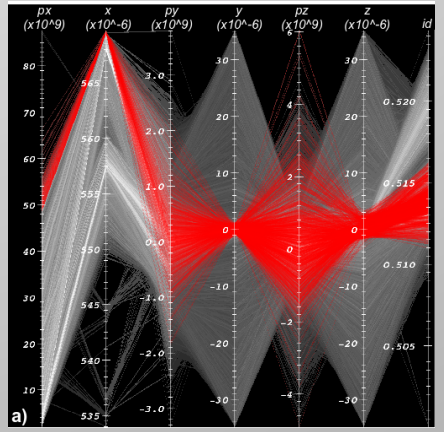
Pseudocolor Rendering



Volume Rendering

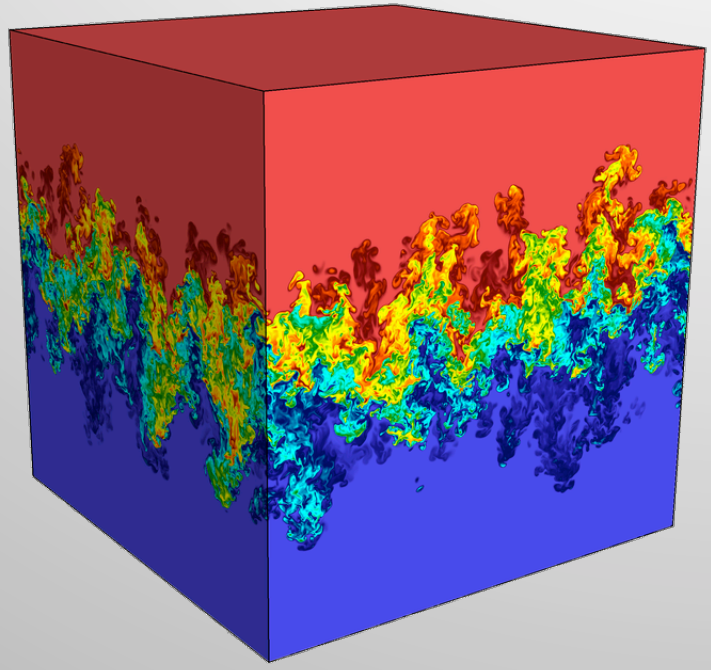


Molecular Visualization

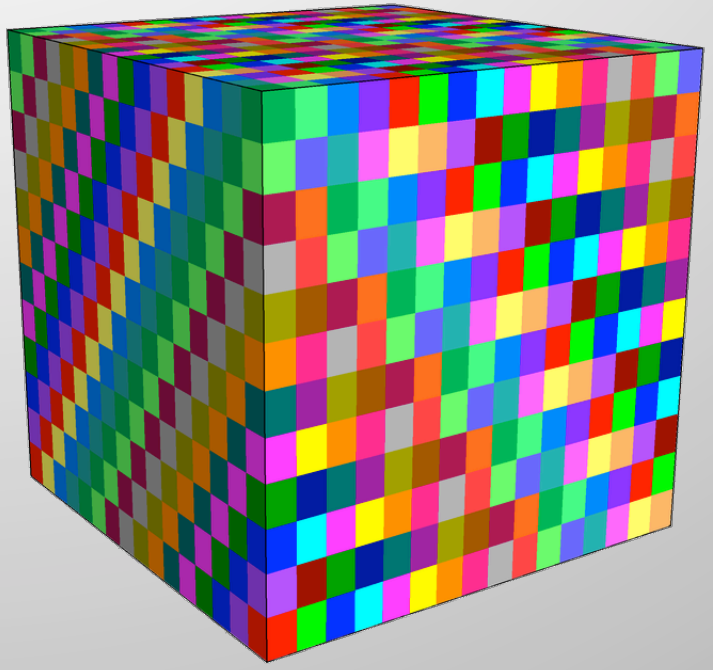


Parallel Coordinates

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



Full Dataset
(27 billion total cells)

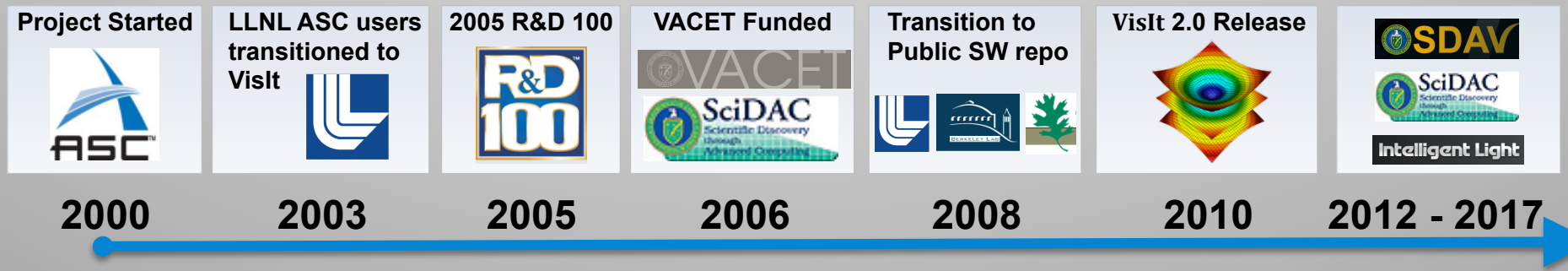


3072 sub-grids
(each 192x129x256 cells)

We are enhancing VisIt's pipeline infrastructure to support threaded processing and many-core architectures.

VisIt is a vibrant project with many participants.

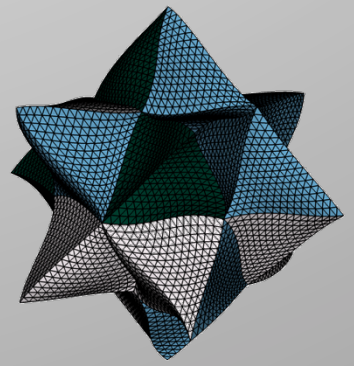
- The VisIt project started in 2000 to support LLNL's large scale ASC physics codes.
- The project grew beyond LLNL and ASC with research and development from DOE SciDAC and other efforts.
- VisIt is now supported by multiple organizations:
  - LLNL, LBNL, ORNL, UC Davis, Univ of Utah, Intelligent Light,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



VisIt's capabilities are constantly being expanded.

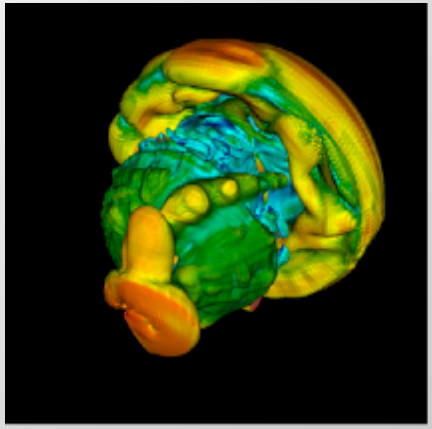
Ongoing + Recent Development Efforts:

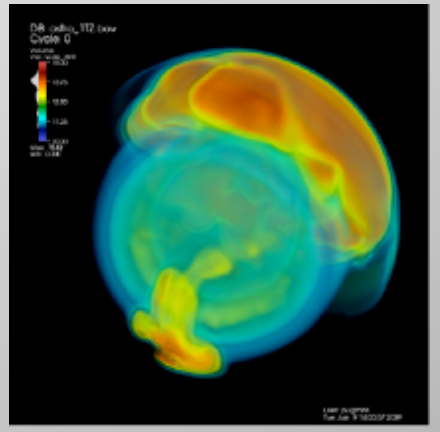
- Evaluation of EAVL / VTK-M for batch In-situ processing
  - http://m.vtk.org/index.php/Main_Page
  - http://ft.ornl.gov/eavl/
- Built-in SeedMe Python support for sharing visualizations
  - http://seedme.org/
- Support for High Order Finite Element Meshes via MFEM
  - https://code.google.com/p/mfem/



VisIt scales well on current HPC platforms.

Machine	Architecture	Problem Size	# of Cores
Graph	*X86_64*	***20,001³ (8 T cells)***	*12K*
Dawn	BG/P	15,871³ (4 T cells)	64K
Franklin	Cray XT4	12,596³ (2 T cells)	32K
JaguarPF	Cray XT5	12,596³ (2 T cells)	32K
Juno	X86_64	10,000³ (1 T cells)	16K
Franklin	Cray XT4	10,000³ (1 T cells)	16K
Ranger	Sun	10,000³ (1 T cells)	16K
Purple	IBM P5	8,000³ (0.5 T cells)	8K



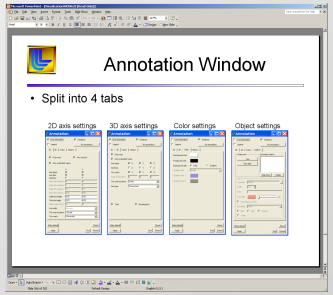


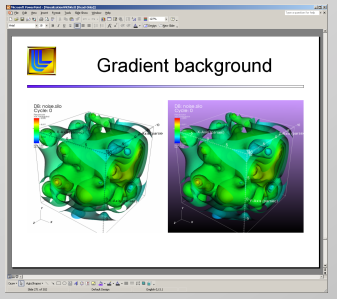
Scaling Studies of Isosurface Extraction and Volume Rendering (2009)

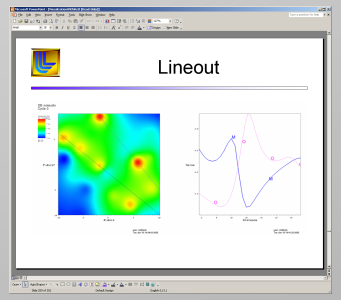
VisIt is also used daily by domain scientists.

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''
- User 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



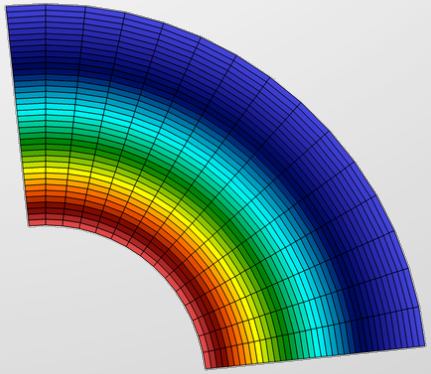
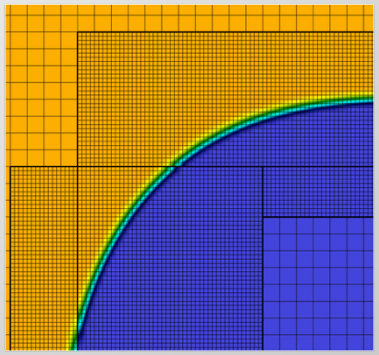
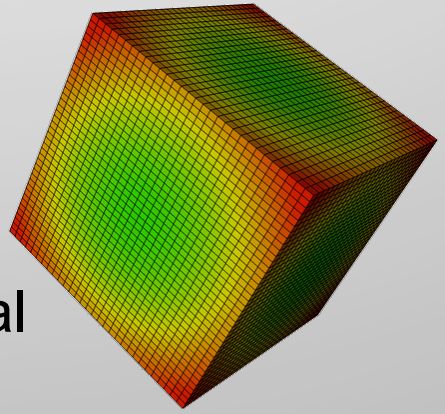
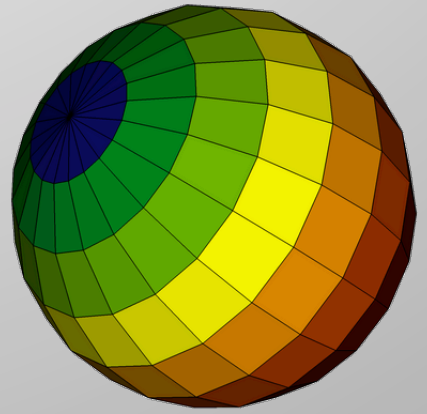




Slides from the VisIt class

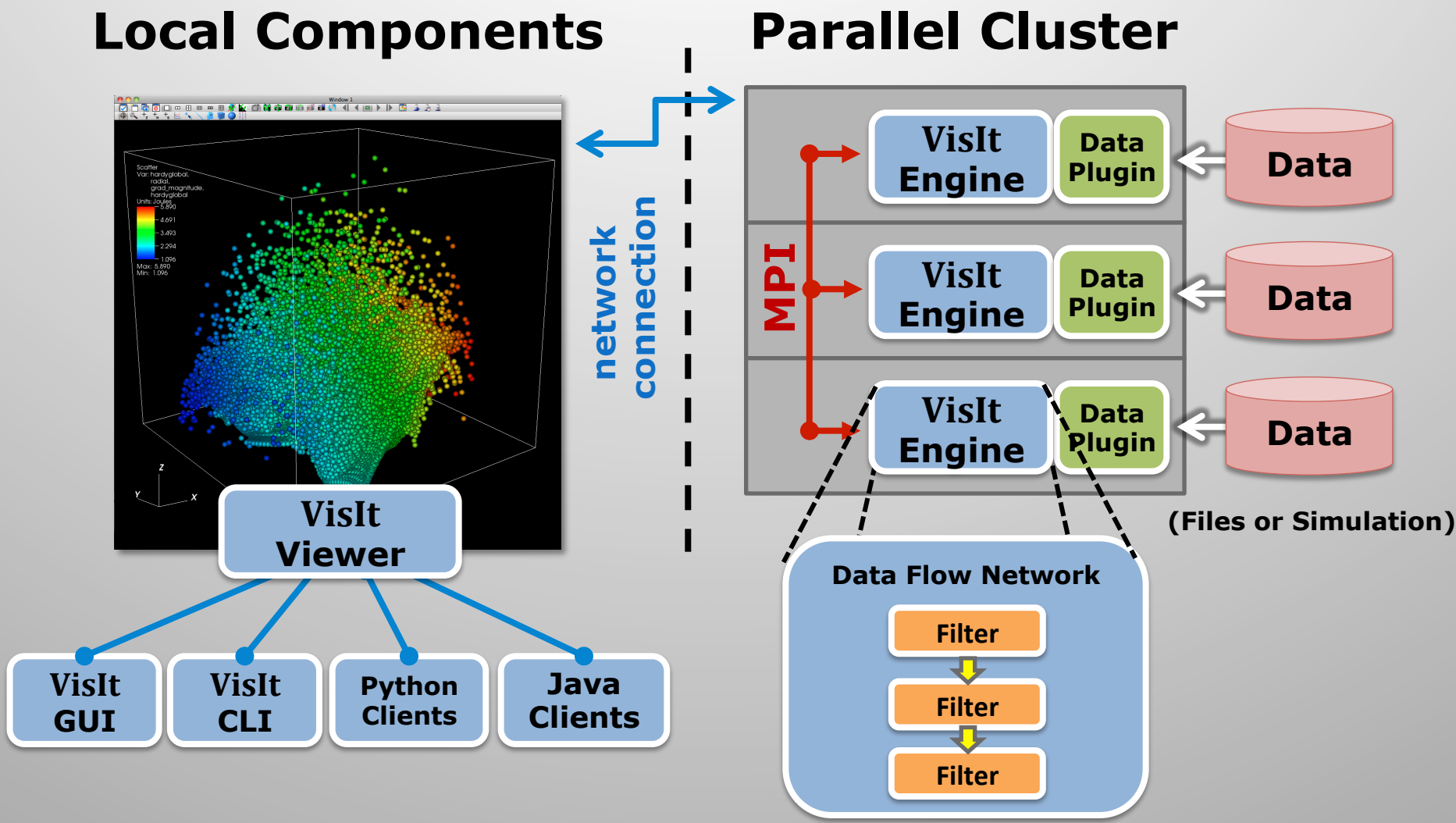
VisIt provides a flexible data model, suitable for many application domains.

- Mesh Types:
  - Point, Curve, 2D/3D Rectilinear, Curvilinear, Unstructured
  - Domain Decomposed, AMR
  - Time Varying
  - Primarily linear element support, limited quadratic element support
- Fields:
  - Scalar, Vector, Tensor, Material volume fractions, Species

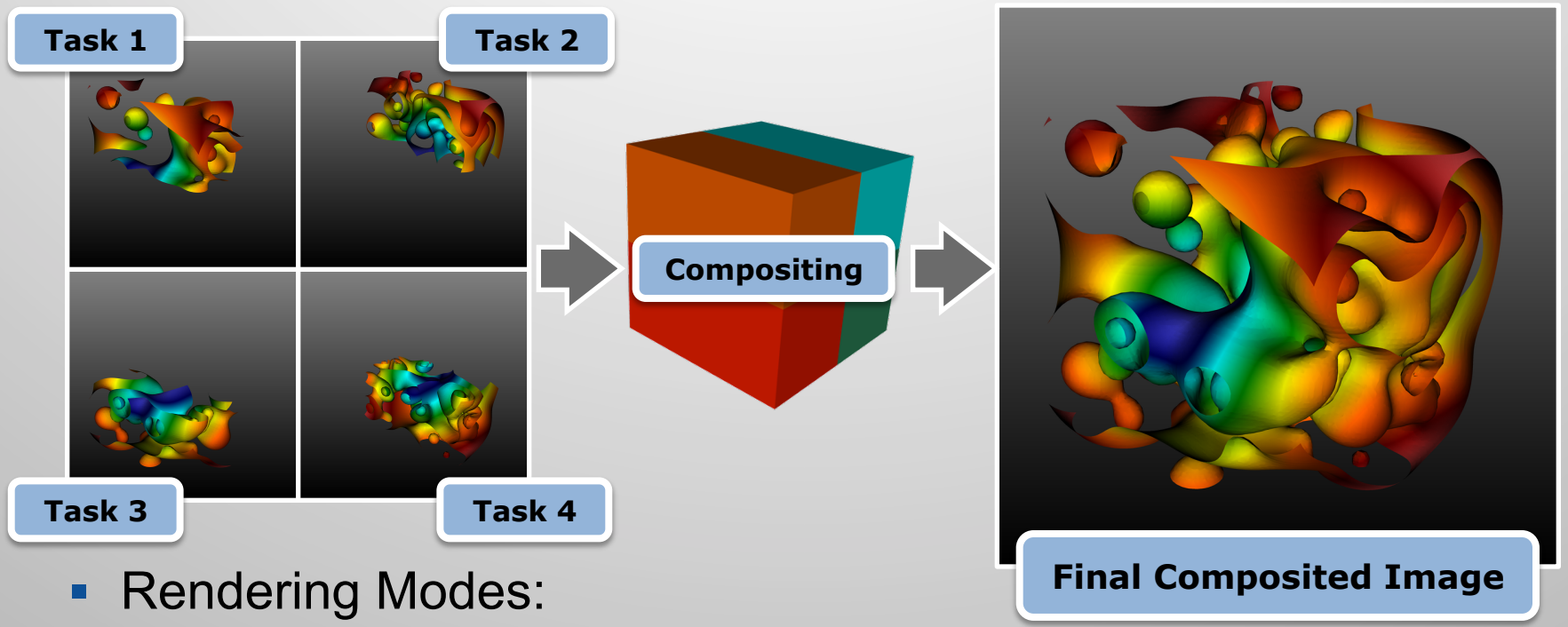





VisIt currently supports over 110 file formats.

VisIt employs a parallelized client-server architecture.



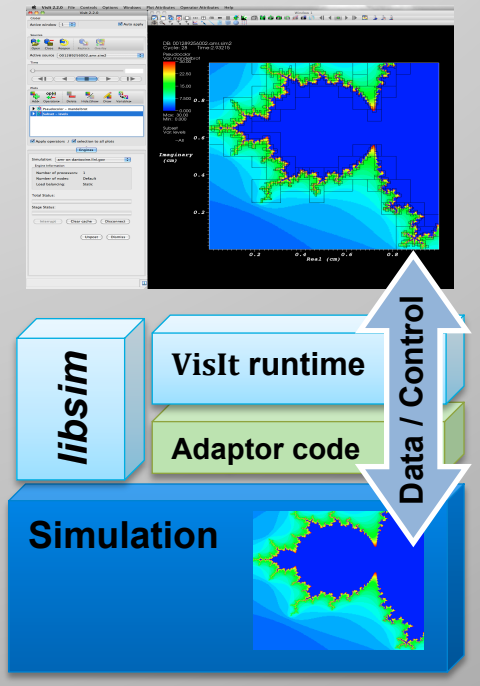
VisIt automatically switches to a scalable rendering mode for large data sets.



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

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.

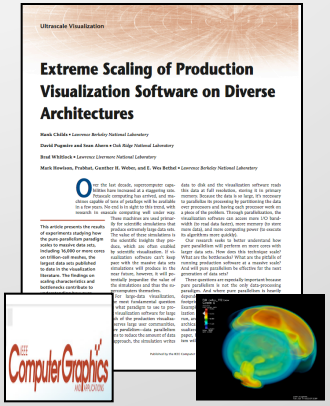
- Research Collaborations:

2006 – 2011

SDAV 2012 – 2017

Scalable Data Management, Analysis, and Visualization

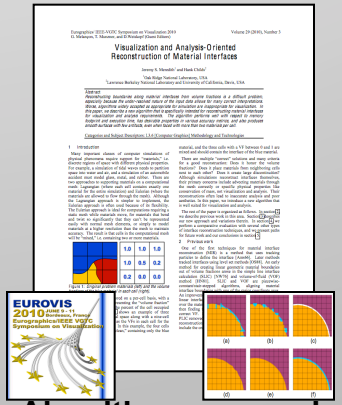
- Research Focus:
  - Next Generation Architectures
  - Parallel Algorithms
  - In-Situ Processing



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.

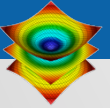
VisIt: What's the Big Deal?

- 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

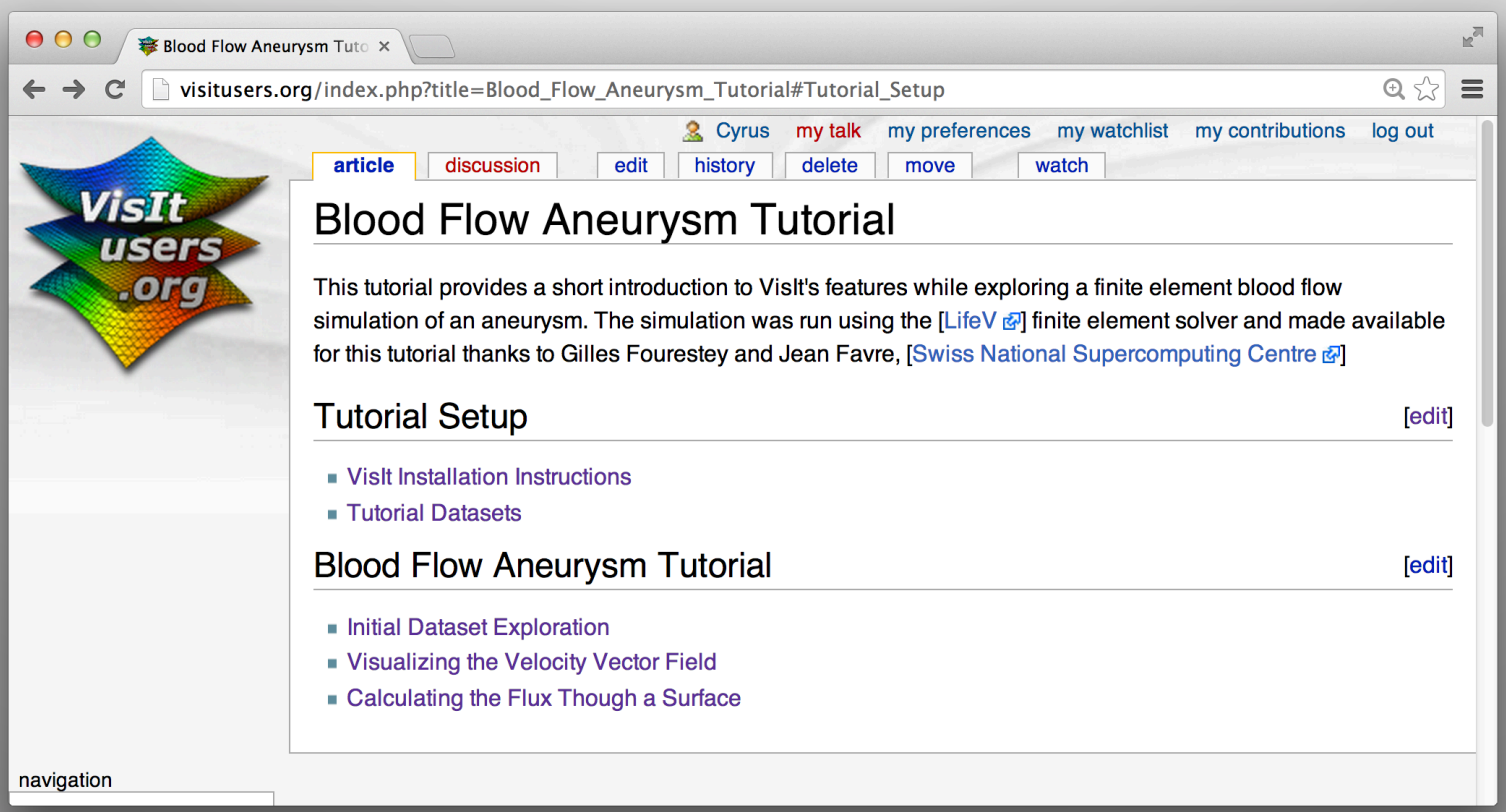
- **User resources:**
  - Main website: http://www.llnl.gov/visit
  - Wiki: http://www.visitusers.org
  - Email: visitusers@ornl.gov
- **Development resources:**
  - Email: visit-developers@ornl.gov
  - SVN: http://portal.nersc.gov/svn/visit



Hands On Visualizations

30 minute Hands on visualization of a Blood Flow Simulation.

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

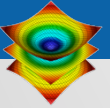


In-depth hands on visualization of a Water Flow Simulation.

Additional Material for Lunch or Dinner Hands-on sessions:

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

[Supporting Slides]



Visualization Techniques for Mesh-based Simulations

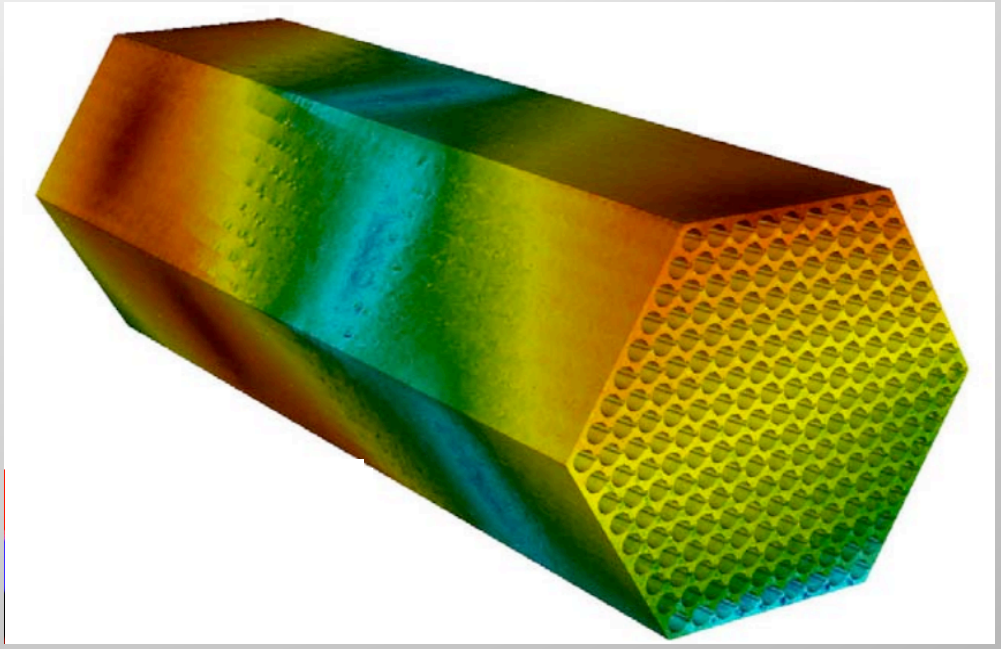
Terminology

- 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

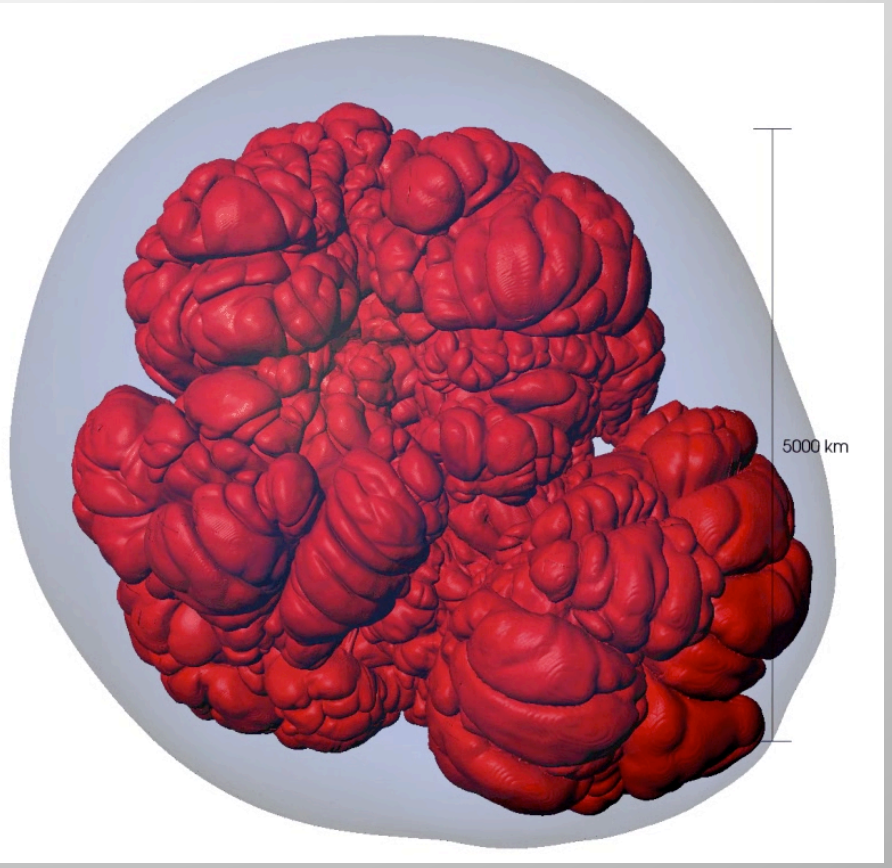
- Maps scalar fields (e.g., density, pressure, temperature) to colors.



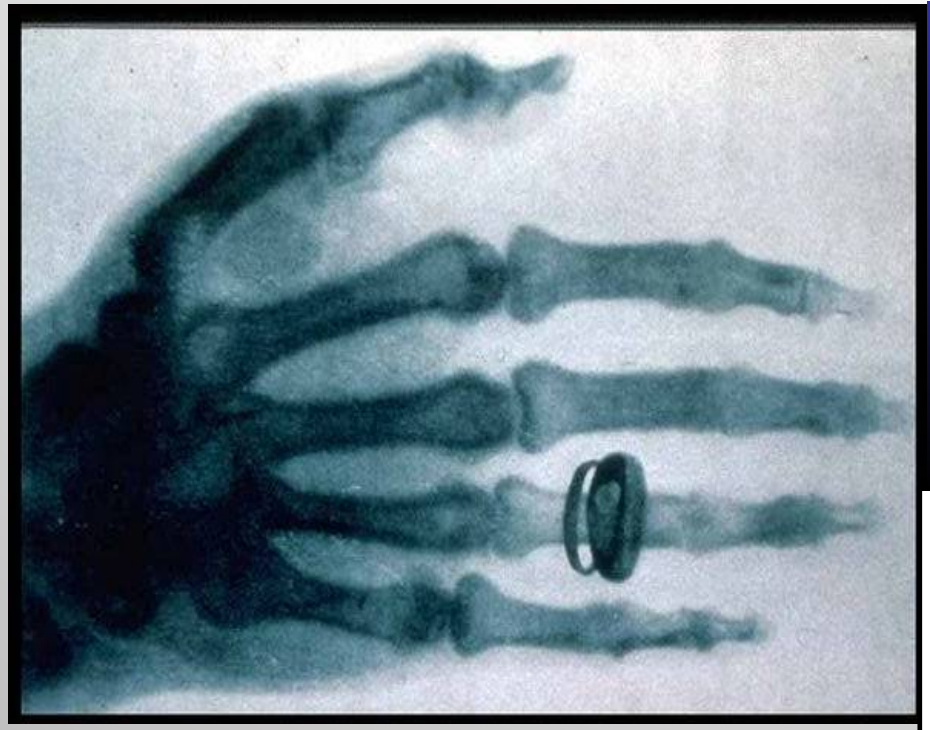


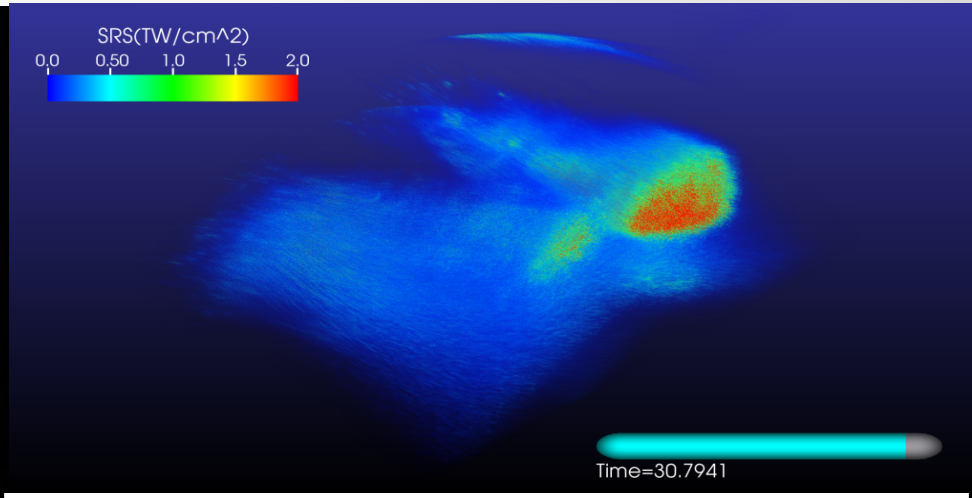
Contour / Isosurface

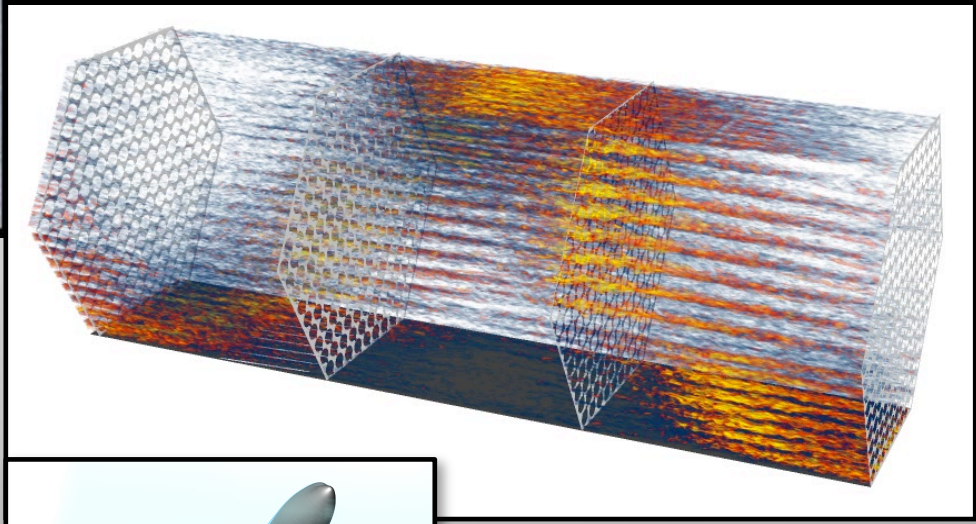


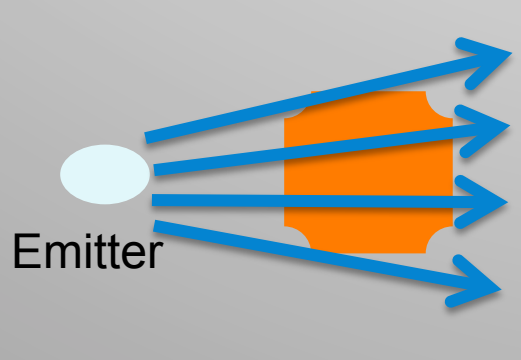


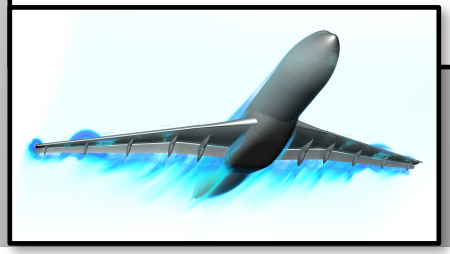
Volume rendering









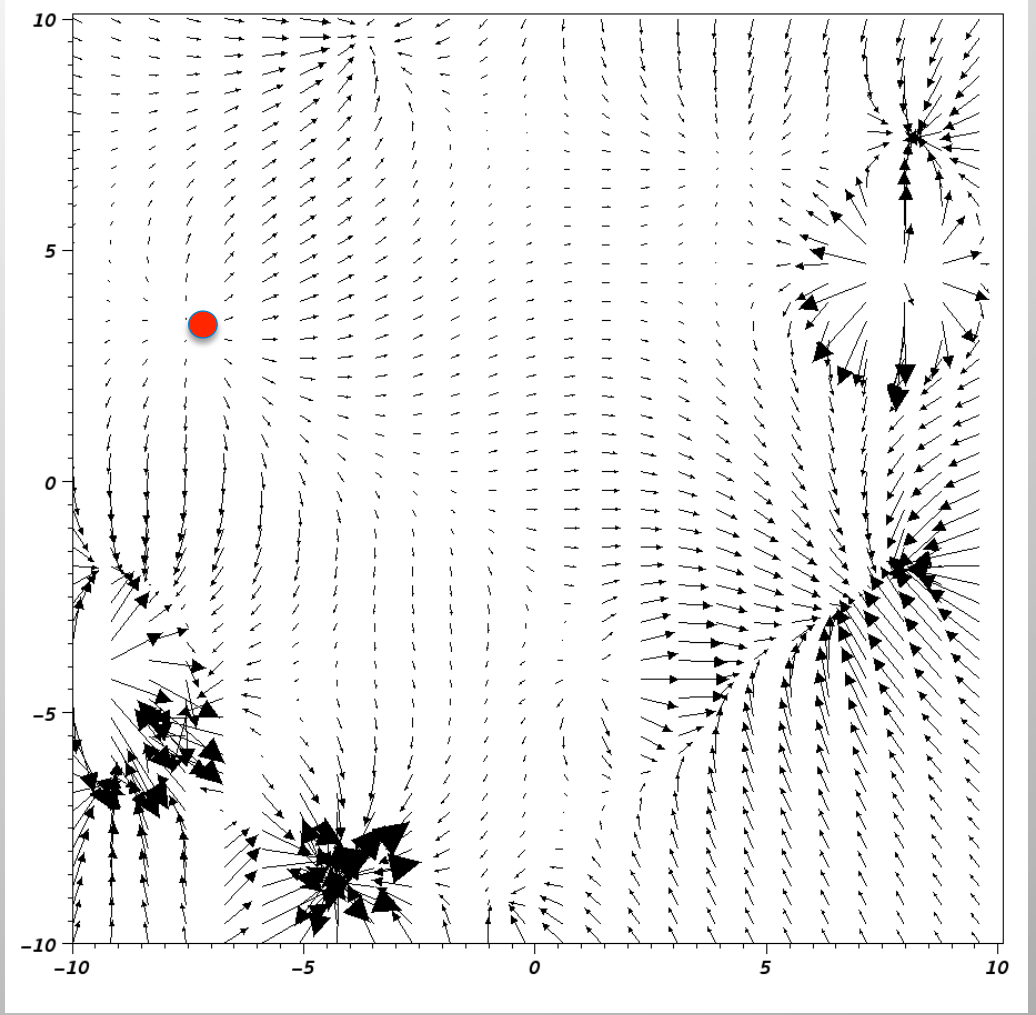


VisIt can combine volume rendering and opaque geometry

Particle advection: the foundation of flow visualization

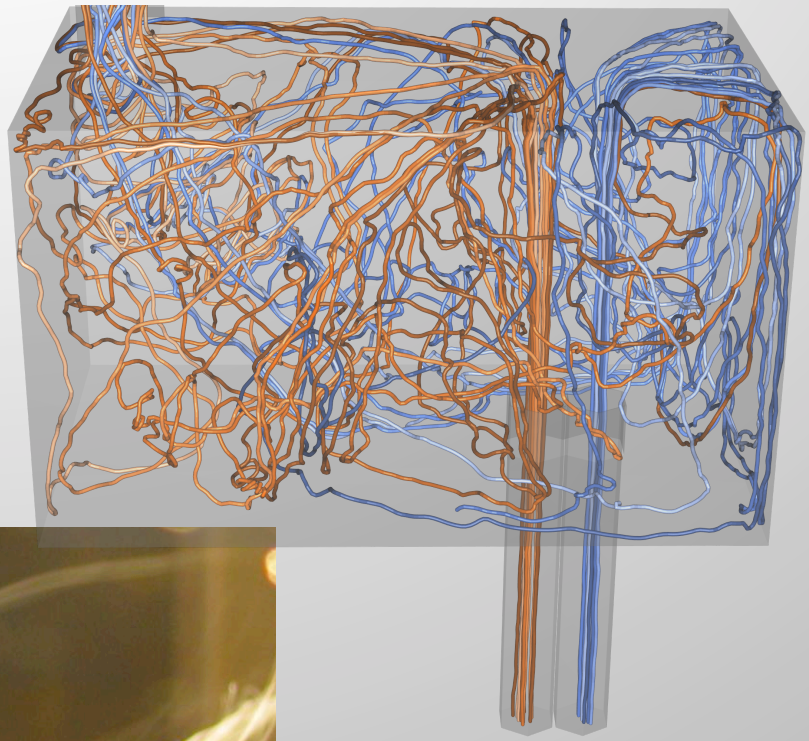
- Displace massless particle based on velocity field
- S(t) = position of curve at time t
  - $S(t_0) = p_0$
    - t_0: initial time
    - p_0: initial position
  - $S'(t) = v(t, S(t))$
    - v(t, p): velocity at time t and position p
    - S'(t): derivative of the integral curve at time t

This is an ordinary differential equation



Streamlines

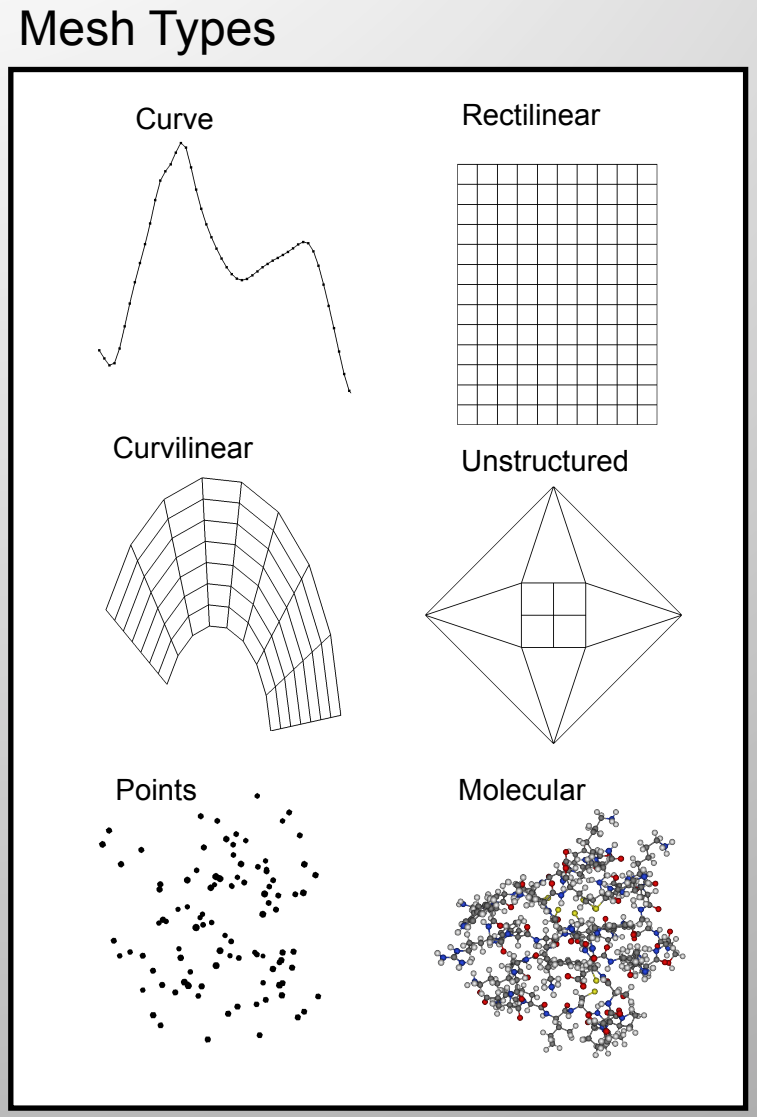
- Streamlines – instantaneous paths
- Pathlines – time dependent paths





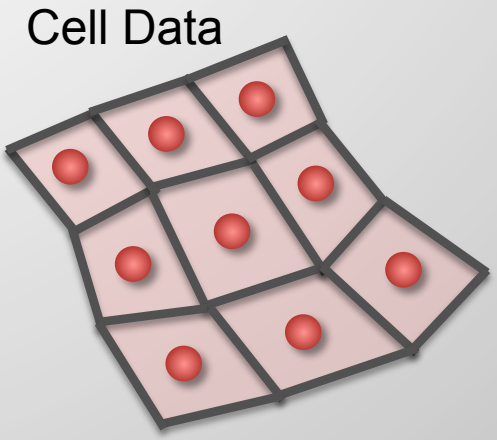
Meshes

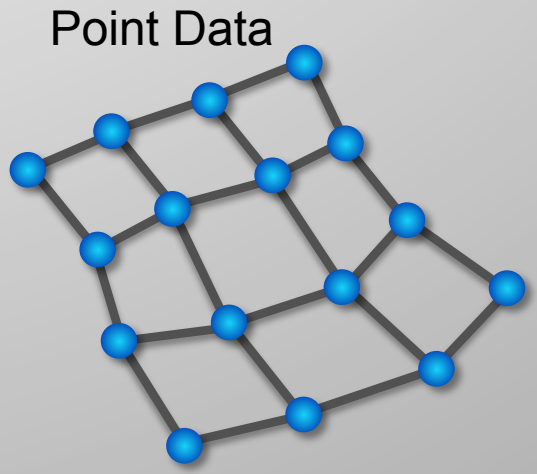
- All data in VisIt lives on a mesh
- Discretizes space into points and cells
  - (1D, 2D, 3D) + time
  - Topological dimension need not match spatial dimension *(e.g. 2D surface in 3D space)*
- Provides a place for data to be located
- Defines how data is interpolated

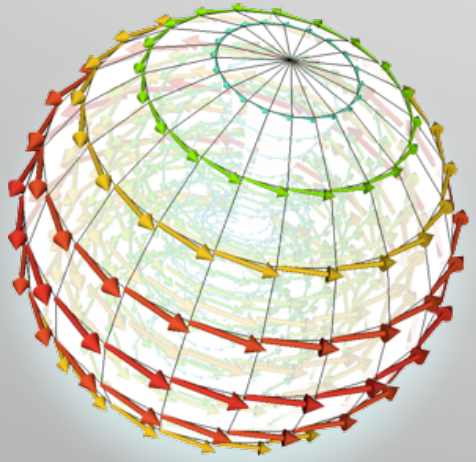


Variables

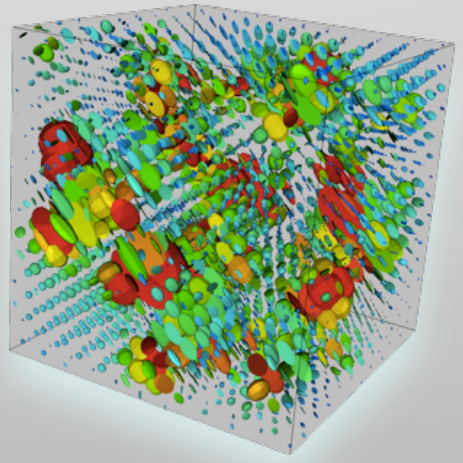
- Scalars, Vectors, Tensors
- Associated with points or cells of a mesh
  - Points: linear interpolation
  - Cells: piecewise constant
- Can have different dimensionality than the mesh (e.g. 3D vector data on a 2D mesh)







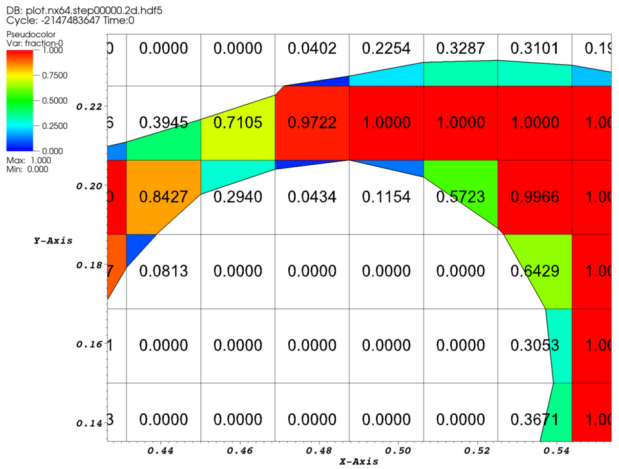
Vector Data

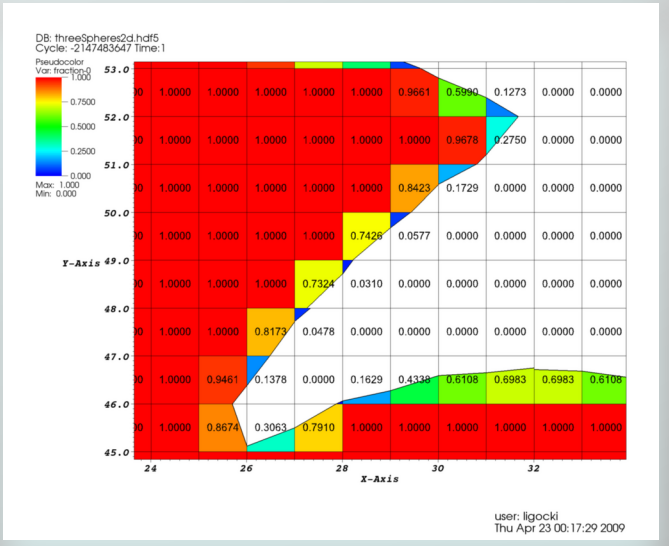


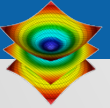
Tensor Data

Materials

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





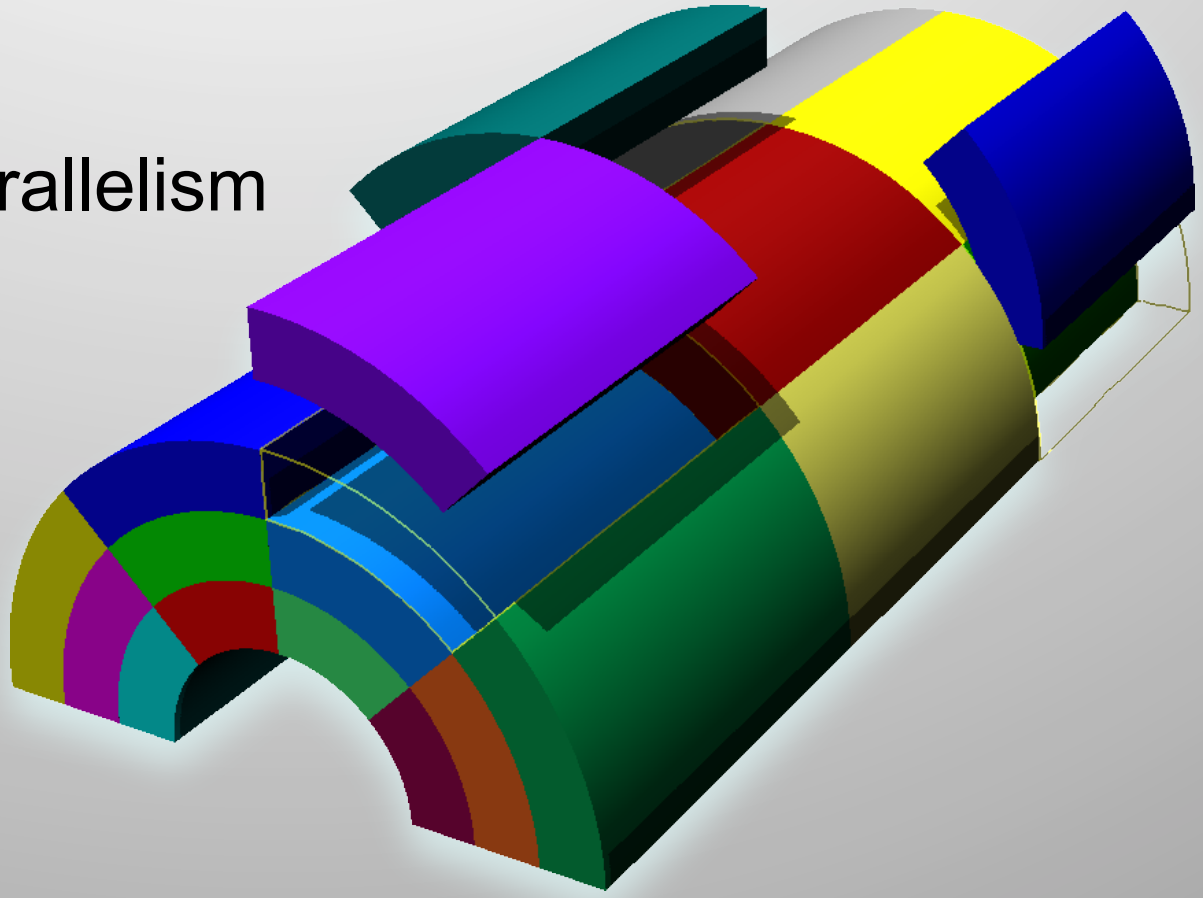


Species

- Similar to materials, describes sub-grid variable composition
  - Example: *Material "Air" is made of species "N_2" ,"O_2", "Ar", "CO_2", etc.*
- Used for mass fractions
- Generally used to weight other scalars (e.g. partial pressure)

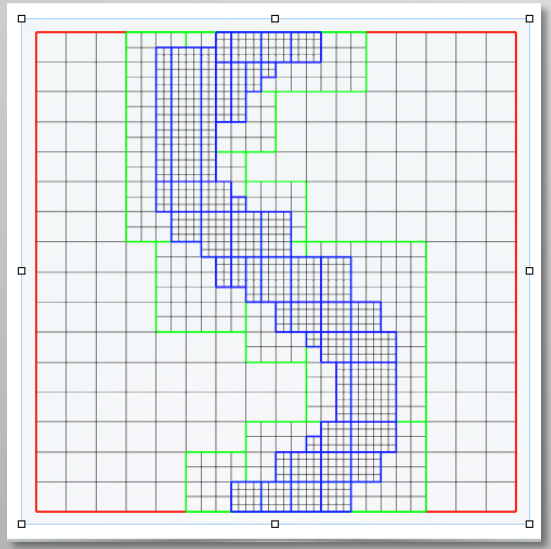
Parallel Meshes

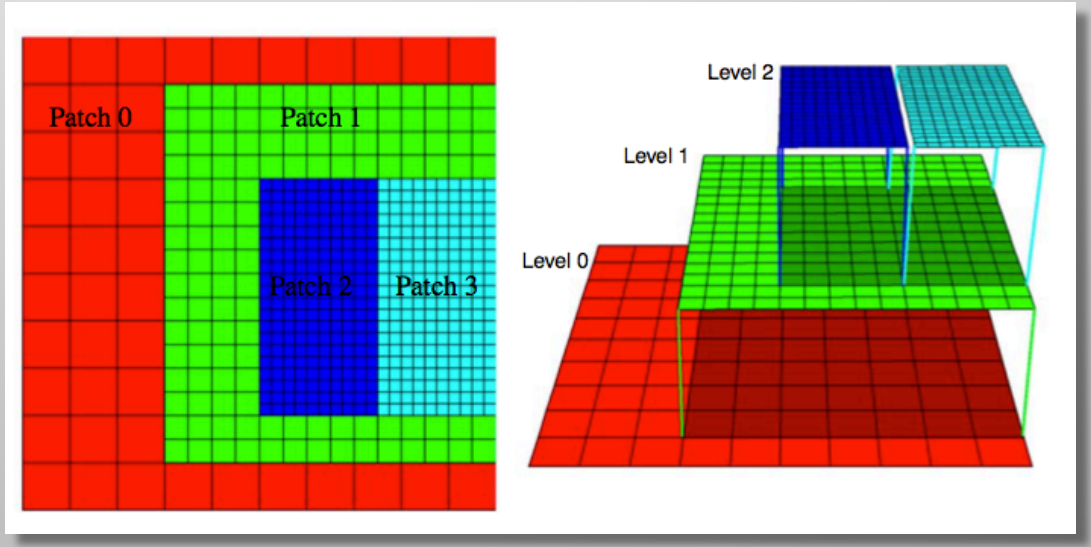
- Provides aggregation for meshes
- A mesh may be composed of large numbers of mesh “blocks”
- Allows data parallelism



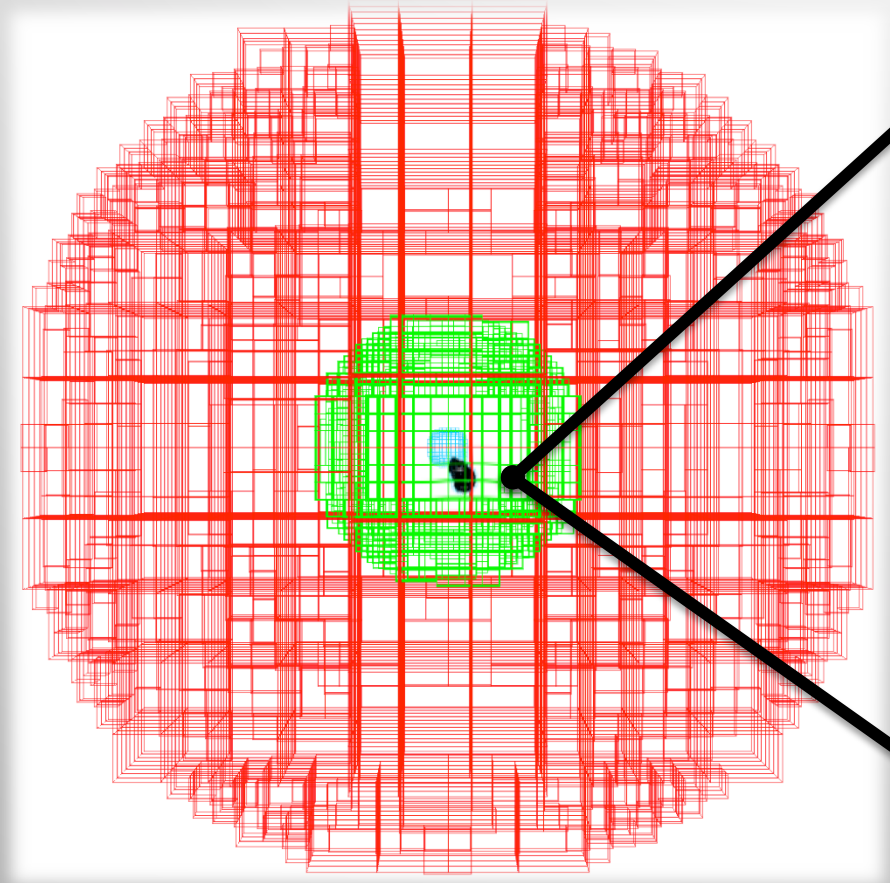
AMR meshes

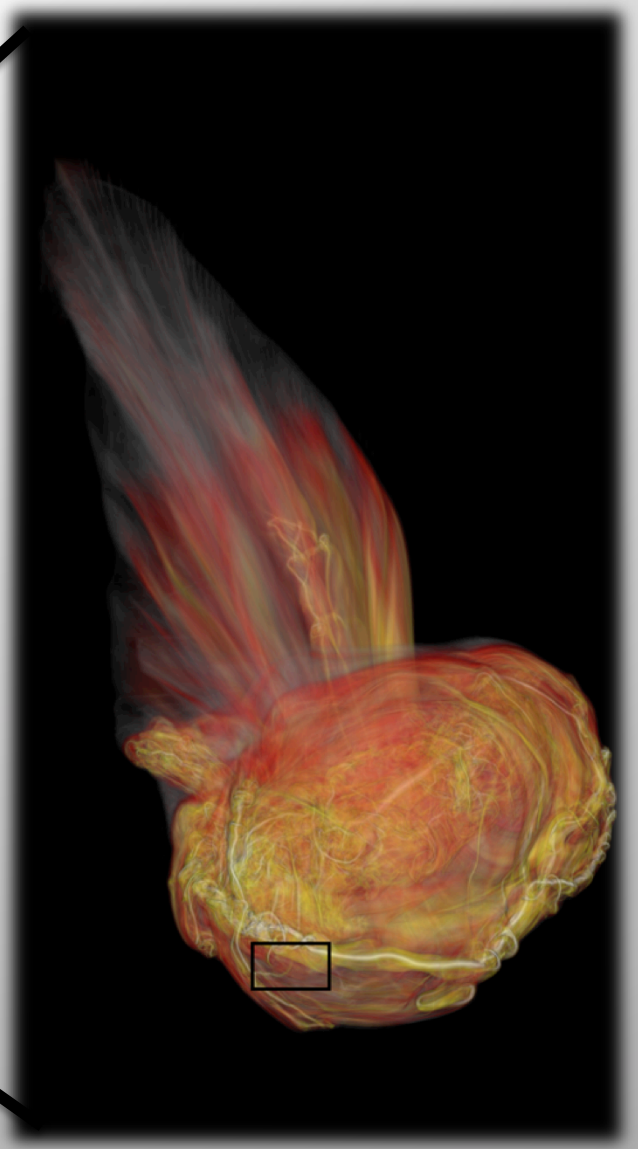
- Mesh blocks can be associated with patches and levels
- Allows for aggregation of meshes into AMR hierarchy levels

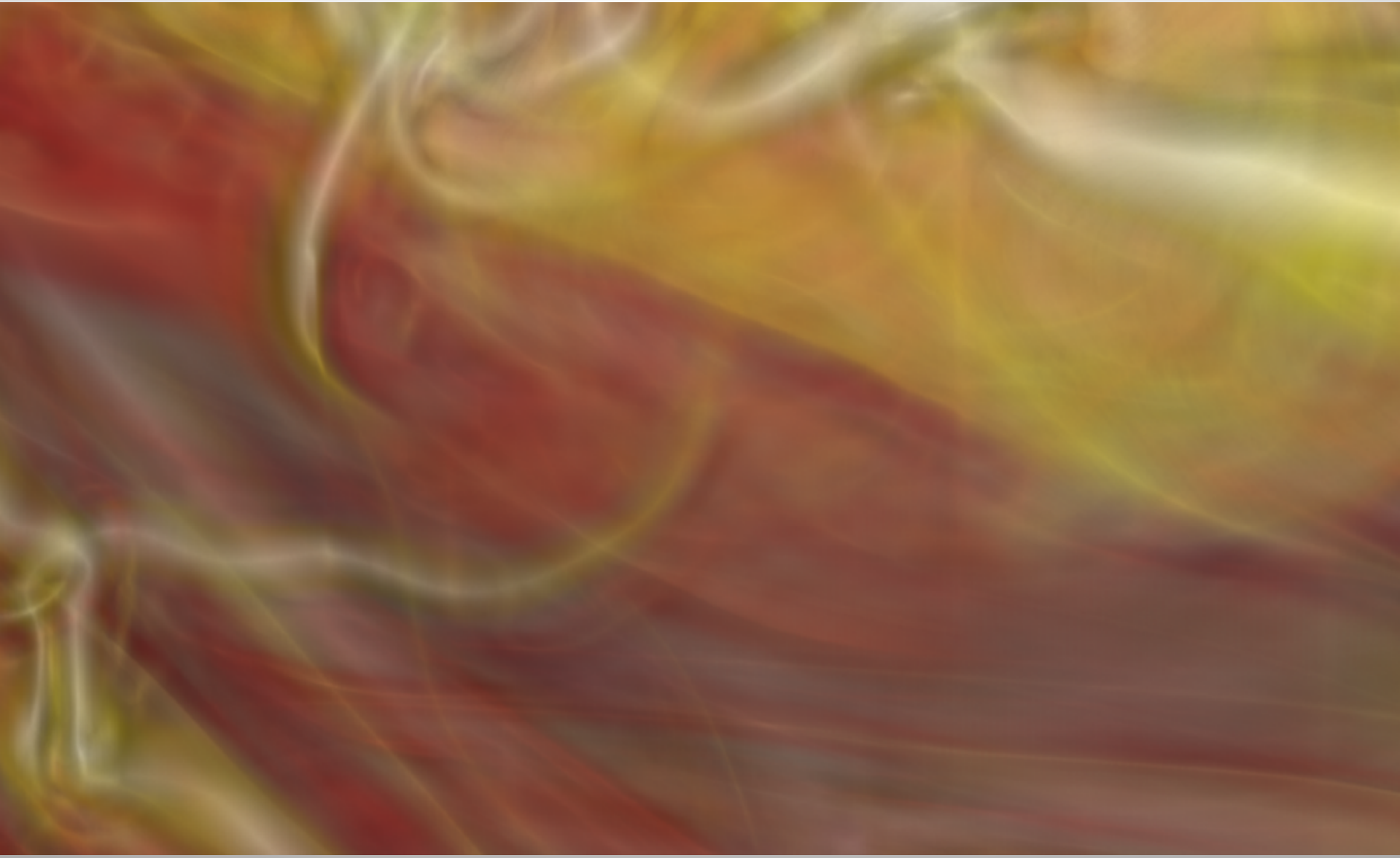


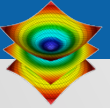


AMR Example: Image vs. Data Resolution

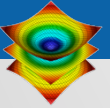








VisIt's Core Abstractions

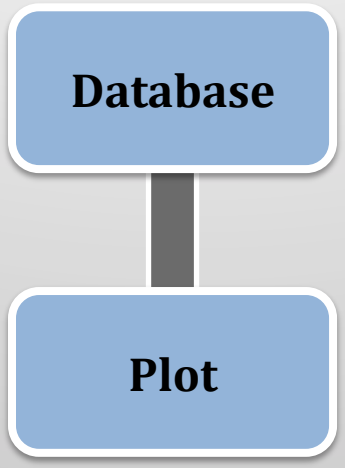


VisIt's core abstractions

- **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



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

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



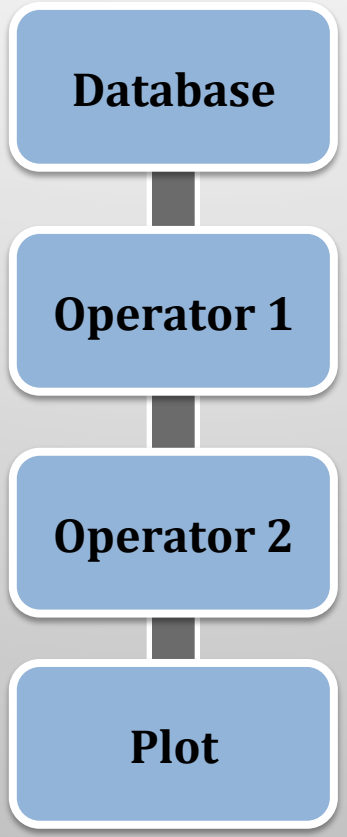
Open a database, which reads from a file (example: open file1.hdf5)

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

Plot a variable in the database (example: Pseudocolor 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



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

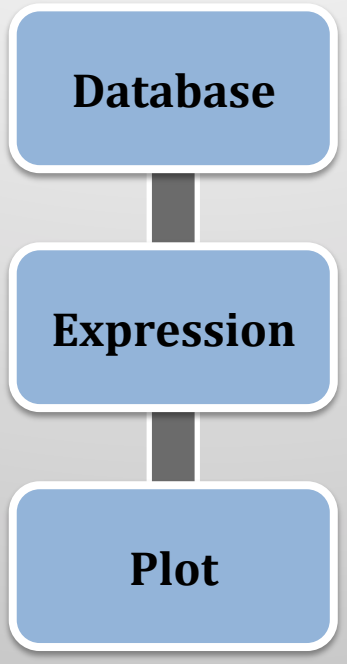
Apply an operator to transform the data (example: Slice operator)

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

Plot a variable in the database (example: Pseudocolor 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



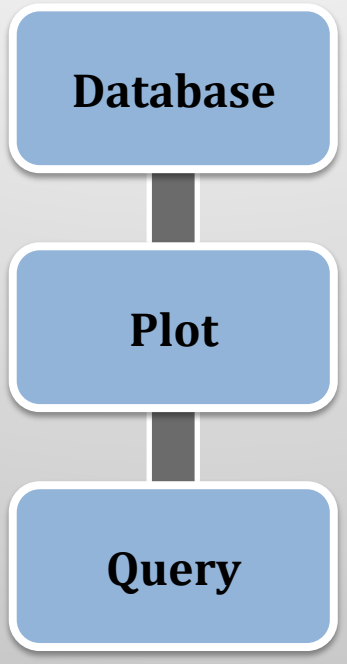
Open a database, which reads from a file (example: open file1.hdf5)

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

Plot the expression variable (example: Pseudocolor 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



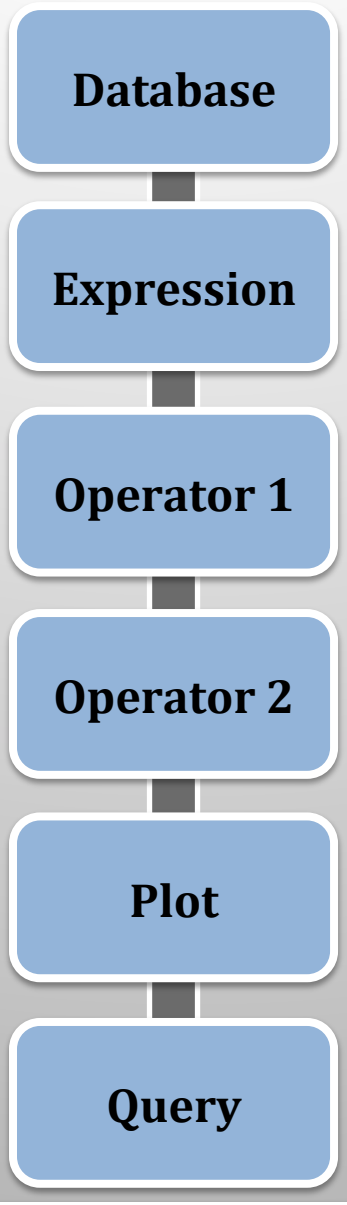
Open a database, which reads from a file (example: open file1.hdf5)

Plot a field from the file (example: density + Pseudocolor plot)

Extract quantitative information (example: integrate density to find mass)

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



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

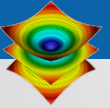
Create derived quantities from fields in the file (example: magnitude(velocity))

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

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

Plot a field (example: speed + Pseudocolor plot)

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



Practical Tips for Using VisIt

Practical Tips for Using VisIt

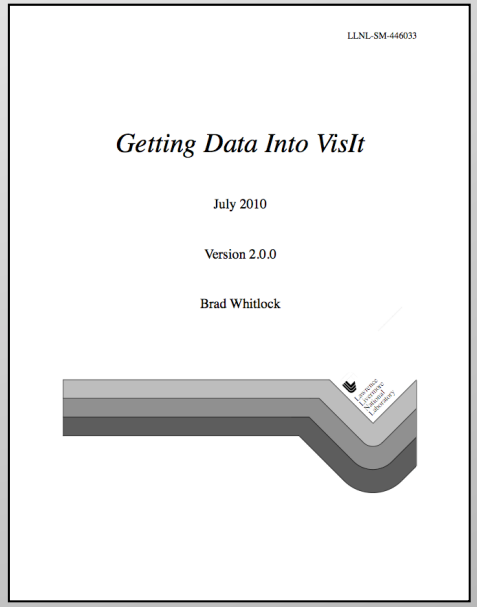
- How to get VisIt to read your data
- How to get help when you run into trouble

How to get VisIt to read your data.

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

https://wci.llnl.gov/simulation/computer-codes/visit/manuals

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


LLNL-SM-446033

Getting Data Into VisIt

July 2010

Version 2.0.0

Brad Whitlock

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

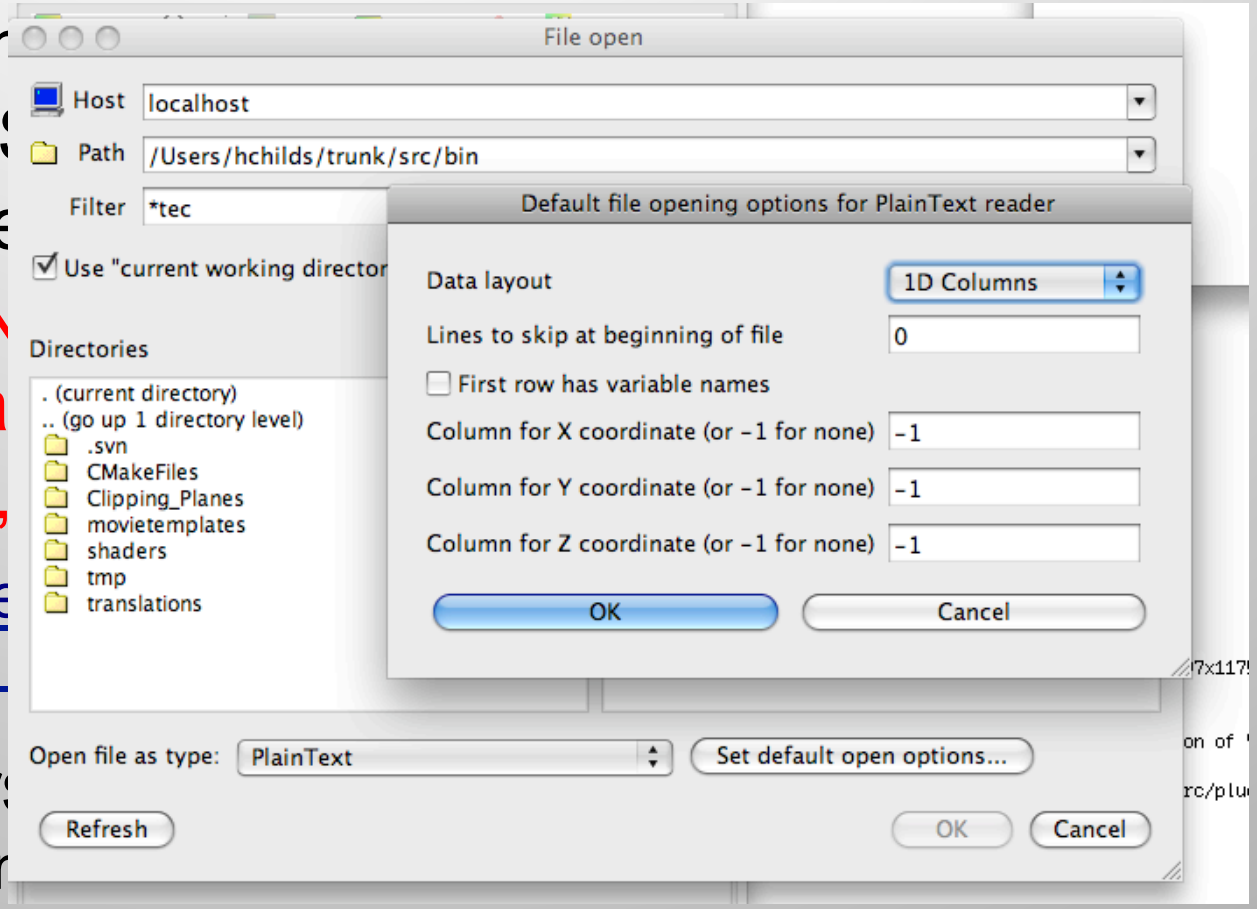
http://www.visitusers.org/index.php?title=Detailed_list_of_file_formats_VisIt_supports

- 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, Chom
  Exodus, FLA
  (TIFF, PNG, e
  NASTRAN, N
  PLOT3D, Pla
  Tecplot, VTK,

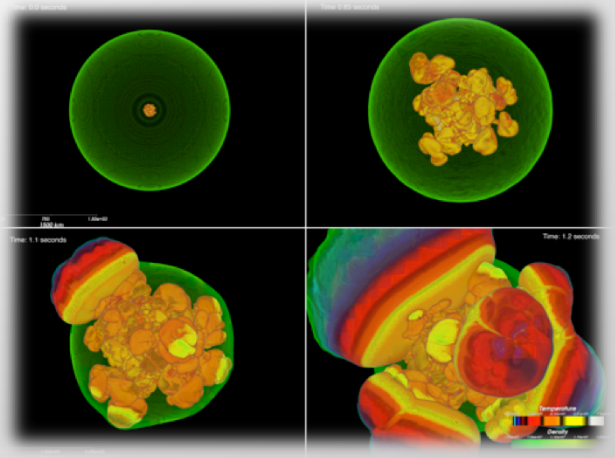
http://www.visituse
title=Detailed_list



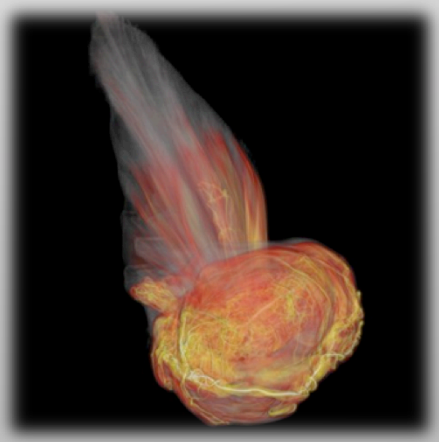
- Some reader
  - For some forr
    a VisIt developer has encountered previously (e.g. Tecplot).

Application Code Formats

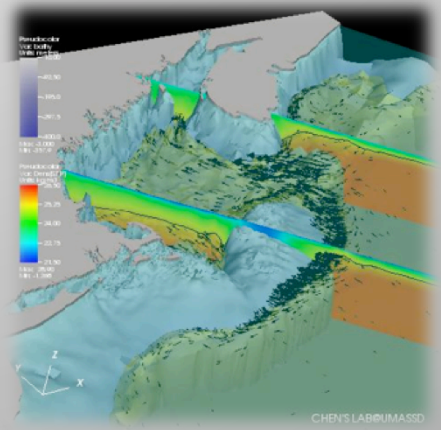
- ANSYS
- Cale
- CASTRO
- CCM
- DDCMD
- Dyna3D
- Enzo
- FLASH
- FVCOM
- Gadget
- LAMMPS
- NASTRAN
- Nek5000
- OVERFLOW
- PATRAN
- Pixie
- S3D
- ZeusMP



FLASH



CASTRO

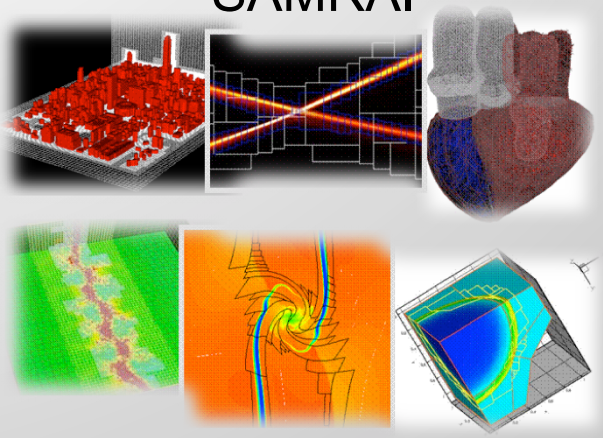


FVCOM

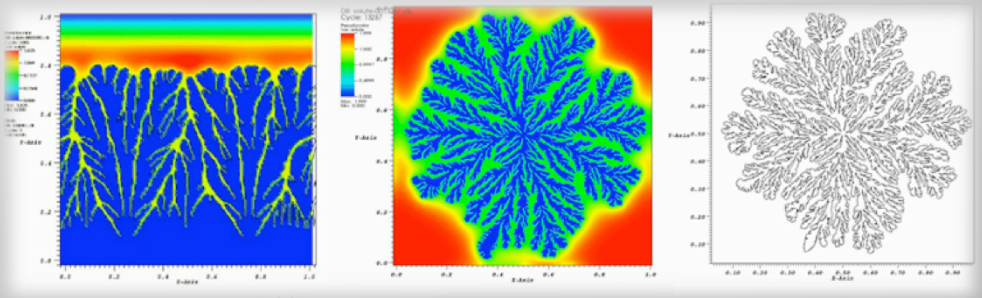
Application Toolkit Formats

- Adventure I/O
- BoxLib
- Chombo
- ITAPS
- OpenFOAM
- SAMRAI
- Spheral

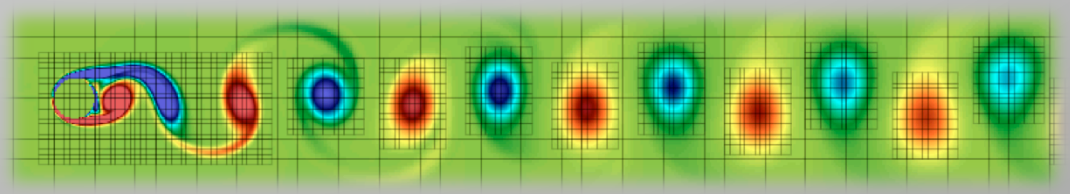
SAMRAI



ITAPS



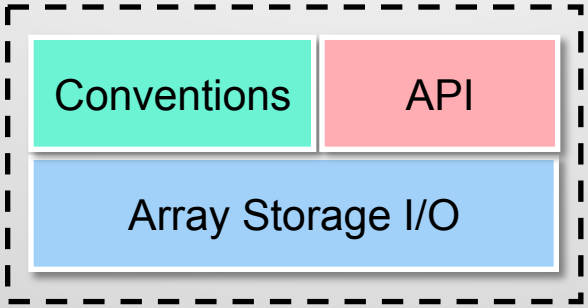
Chombo



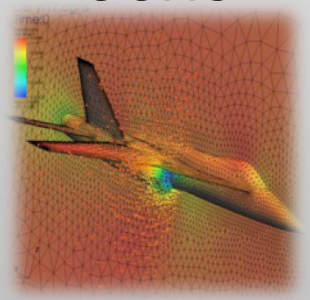
General Scientific Data Formats

- ADIOS
- CGNS
- Exodus
- HDF5
- H5Part
- NETCDF
- PDB
- Silo
- XDMF

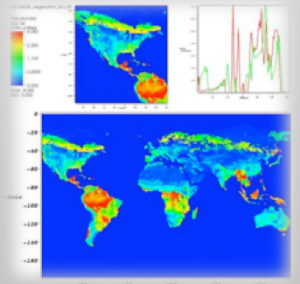
Common Structure



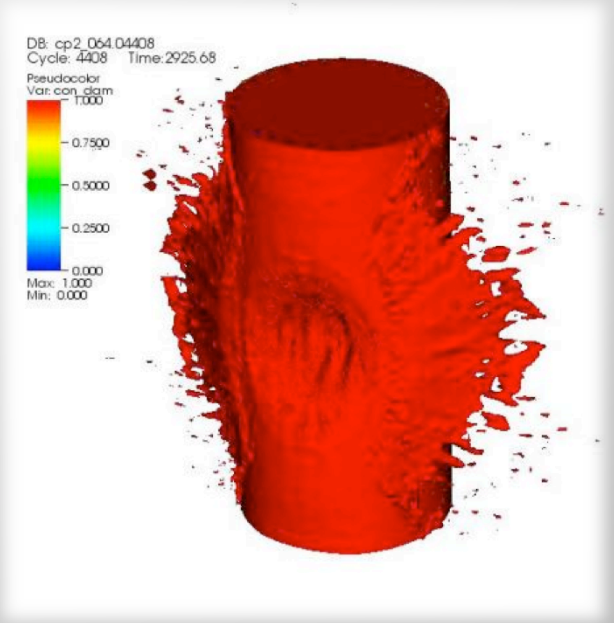
CGNS



NETCDF



Silo / Ale3d



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.

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/simulation/computer-codes/silo

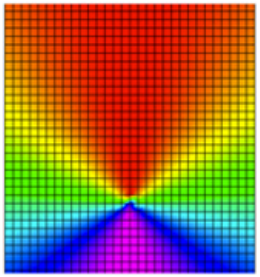
Silo features

WCI | B Codes – SILO

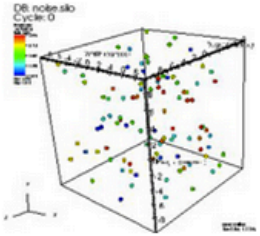
https://wci.llnl.gov/codes/silo/

Welcome to Silo

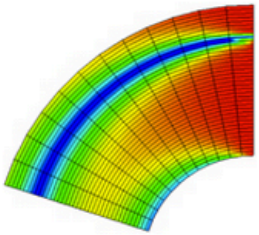
A mesh and field I/O library and scientific database



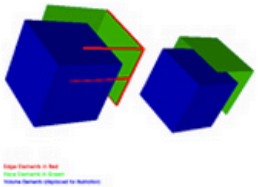
Structured Rectilinear Mesh



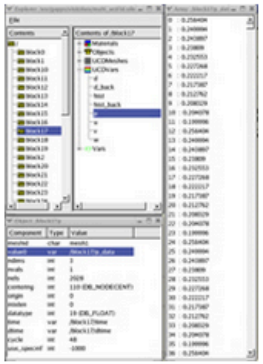
Gridless Point Mesh



Structured (Curvilinear) Mesh



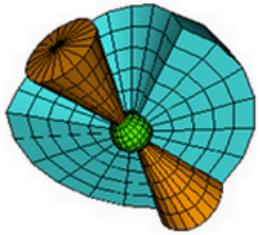
Arbitrary Subsets



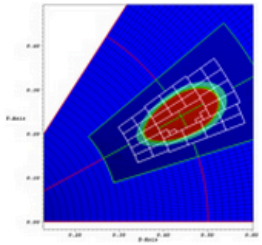
Silex browser for Silo files



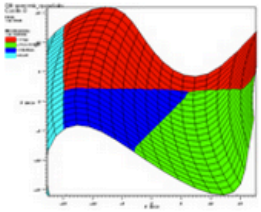
Constructive Solid Geometry (CSG) Mesh



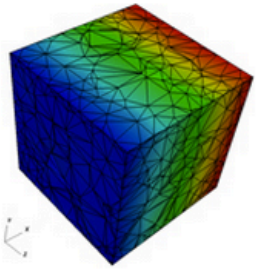
Unstructured Zoo (UCD) Mesh



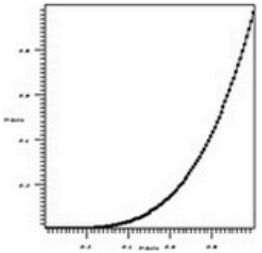
Adaptive Mesh Refinement (AMR) Mesh



Mixing Materials



Arbitrary Polyhedral Mesh

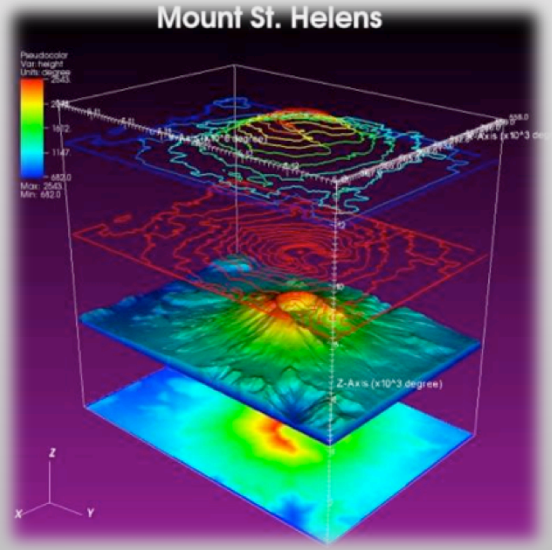


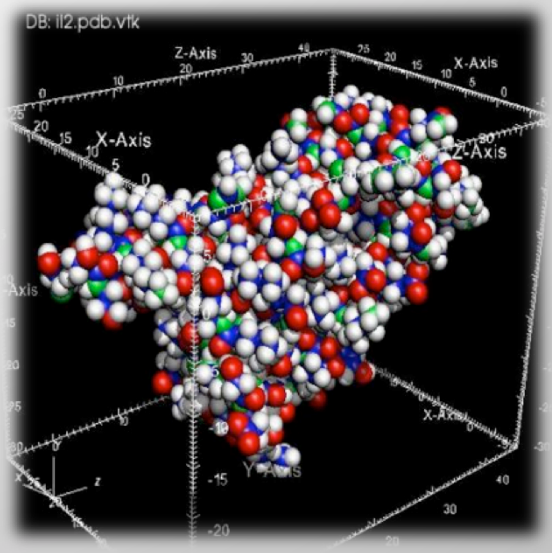
XY Curve

Specialized Scientific Data Formats

- BOW
- FITS
- GDAL
- MatrixMarket
- ProteinDataBank
- ESRI Shapefile
- XYZ

DEM from GDAL

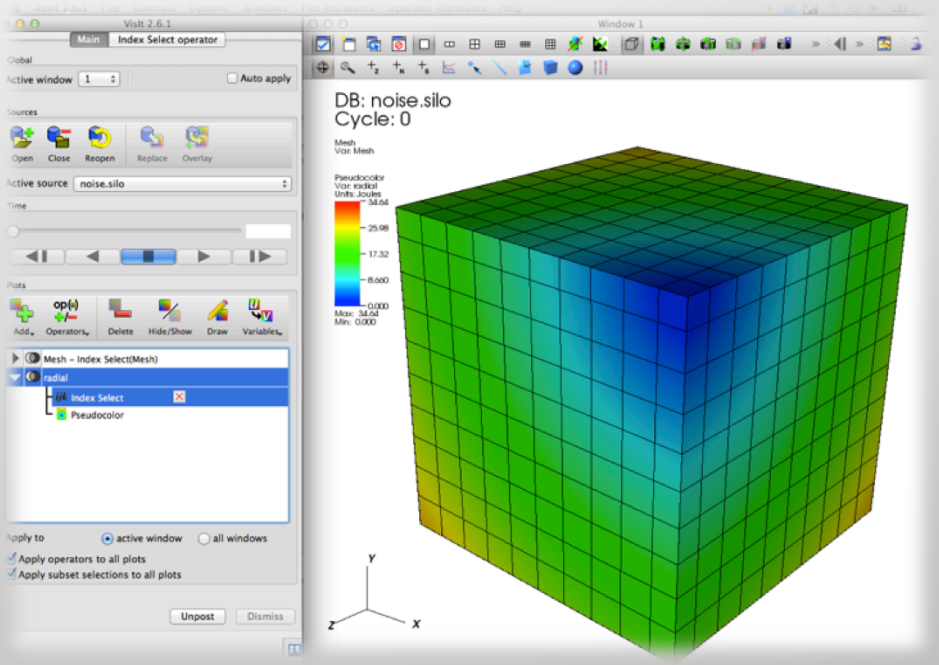




Protein Data Bank

Visualization Formats

- VTK
- EnSight
- GMV
- Plot3D
- Tecplot
- Vis5D
- Xmdv

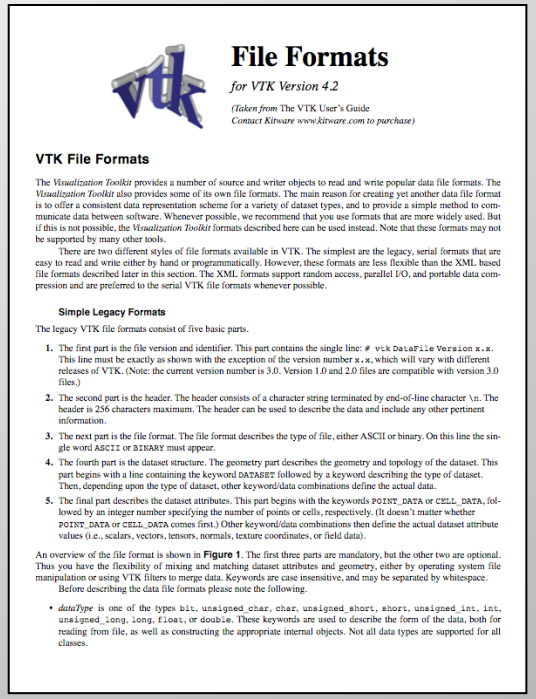


VTK File Format

- 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: `# vtk DataFile 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 operating 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 a 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()
```

Graphics Formats

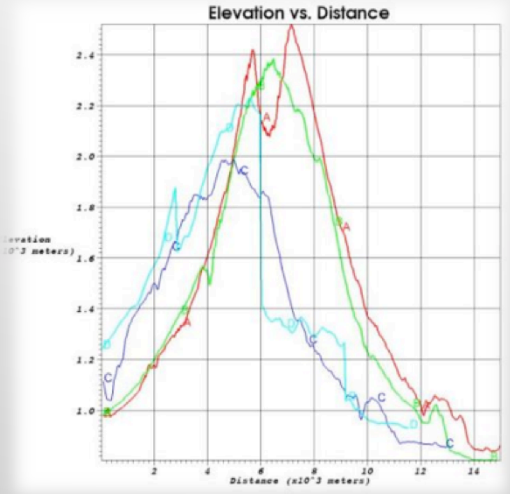
- Image
  - (PNG, JPEG, TIFF, BMP, etc.)
- RAW
- STL
- Wavefront OBJ

Carina Nebula



General ASCII Data Formats

- Curve2D
- Lines
- PlainText
- Point3D



	i=0	i=1	i=2	i=3	i=4	i=5
j=7	2.517243	2.550414	2.581495	2.609803	2.634335	2.653569
j=6	2.472034	2.503052	2.531701	2.557125	2.578064	2.592629
j=5	2.427398	2.456259	2.482482	2.505081	2.522616	2.532976
j=4	2.383583	2.410415	2.434426	2.454567	2.469347	2.476640
j=3	2.340819	2.365857	2.388012	2.406262	2.419193	2.424868
j=2	2.299279	2.322814	2.343538	2.360524	2.372542	2.377986
j=1	2.259063	2.281395	2.301101	2.317398	2.329294	2.335572
j=0	2.220195	2.241595	2.260633	2.276686	2.289012	2.296766

```
2.72727272727 7.57851239669
2.77777777778 7.78703703704
2.82828282828 7.99938781757
2.87878787879 8.21556473829
2.92929292929 8.4355677992
2.9797979798 8.65939700031
3.0303030303 8.8870523416
3.08080808081 9.11853382308
3.13131313131 9.35384144475
3.18181818182 9.59297520661
3.23232323232 9.83593510866
3.28282828283 10.0827211509
3.33333333333 10.3333333333
3.38383838384 10.587771656
3.43434343434 10.8460361188
3.48484848485 11.1081267218
3.53535353535 11.374043465
3.58585858586 11.6437863483
3.63636363636 11.9173553719
3.68686868687 12.1947505357
3.73737373737 12.4759718396
```

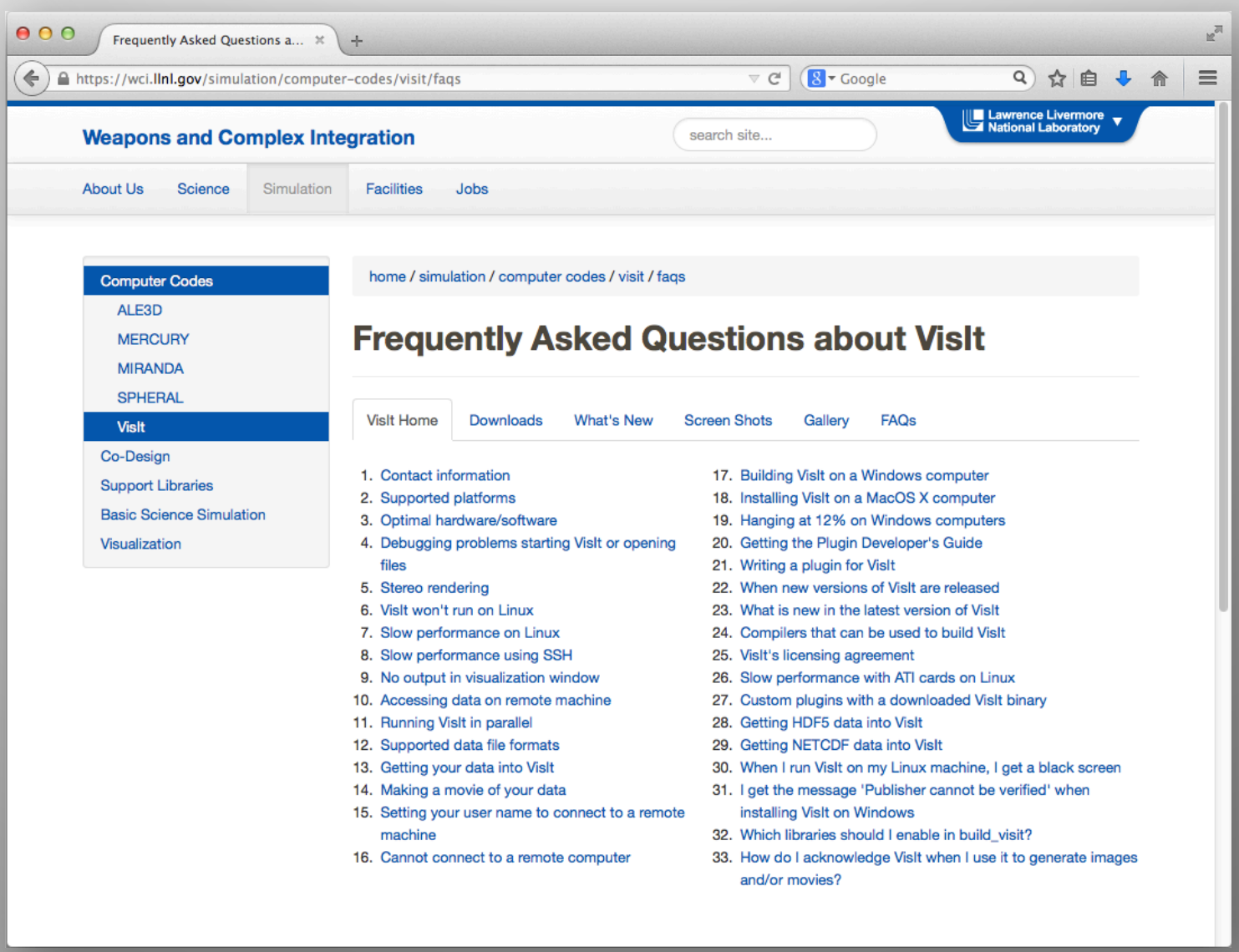
Practical Tips for Using VisIt

- 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
  - https://wci.llnl.gov/simulation/computer-codes/visit/faq
- 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/simulation/computer-codes/visit/manuals

FAQ: https://wci.llnl.gov/simulation/computer-codes/visit/faqs

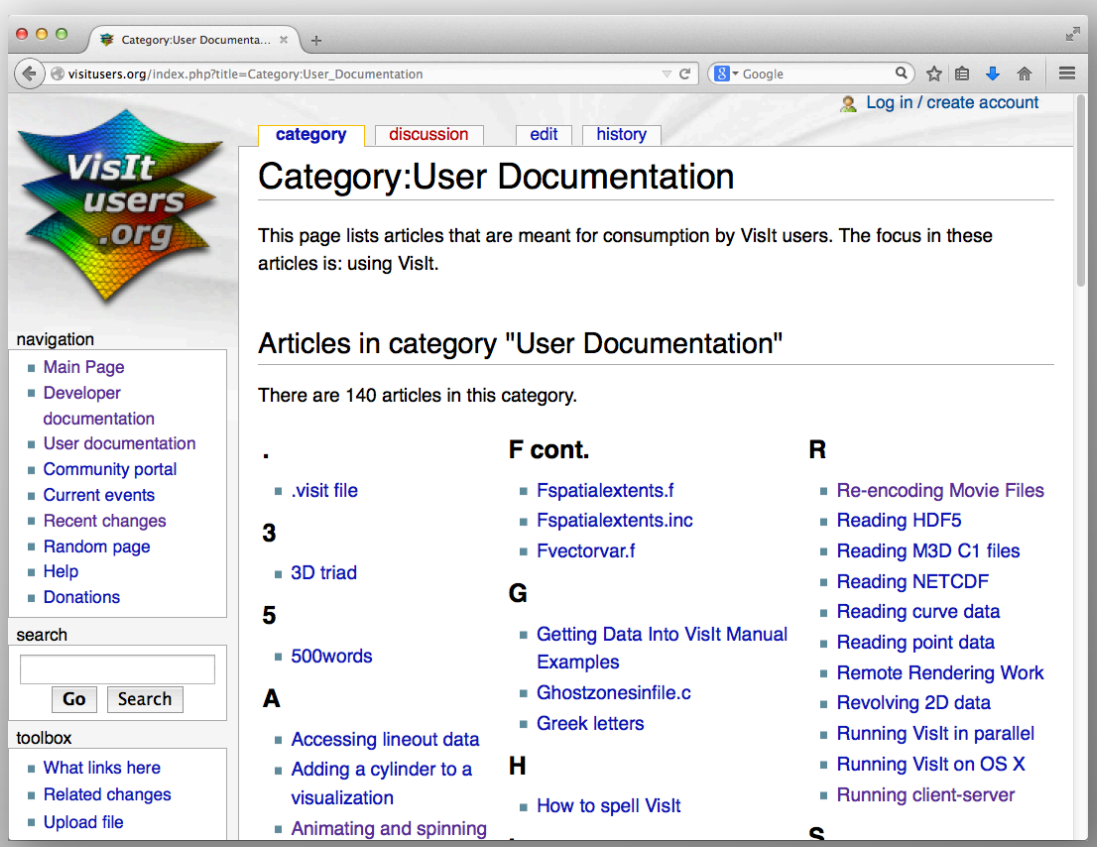


VisIt-users Mailing List

- 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"



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 pseudoco
location of the legend (turn off Let VisIt manage
number of Tic Marks, and the label appearance (
set these properties in a python script? If so, hov

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

Thanks,
Jennifer

Back to top

PM

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

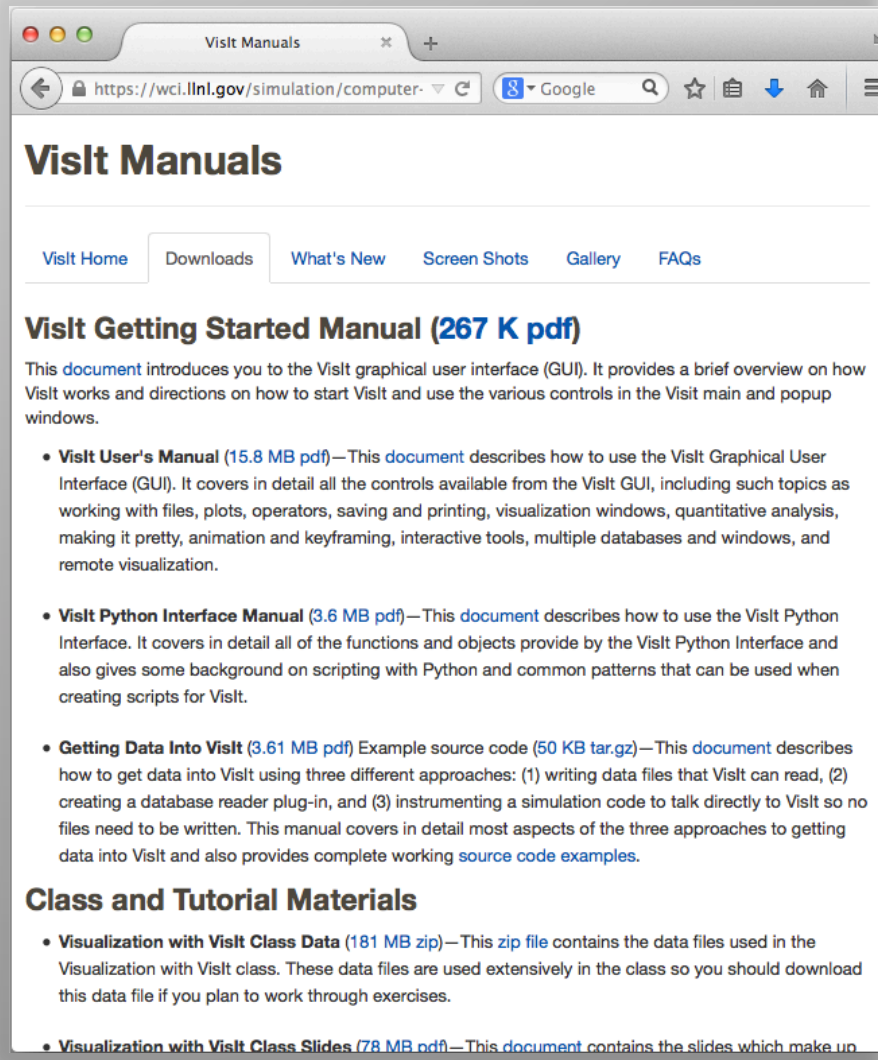
```
>>> GetAnnotationObjectNames()
('Plot0003',)
>>> a = GetAnnotationObject("Plot0003")
>>> a
active = 1
managePosition = 1
position = (0.05, 0.9)
xScale = 1
yScale = 1
```

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.
- Example Lists:
  - visit-help-asc, visit-help-scidac

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