ParaViewによる大規模可視化
Supercomputing 2008 チュートリアル

Kenneth Moreland[1]
Sandia National Laboratories

John Greenfield[2]
Sandia National Laboratories

W. Alan Scott[3]
Sandia National Laboratories

Utkarsh Ayachit[4]
Kitware Inc.

Berk Geveci[5]
Kitware Inc.

David DeMarle[6]
Kitware Inc.

Japanese translators

大嶋 拓也
Takuya OSHIMA
oshima@eng.niigata-u.ac.jp

荻野 佳
Kei OGINO
ogino@env.arch.t.u-tokyo.ac.jp

田中 秀和
Hidekazu TANAKA
tanaka-h@env.arch.t.u-tokyo.ac.jp

今野 雅
Masashi IMANO
imano@arch.t.u-tokyo.ac.jp

[1]kmorel@sandia.gov
[2]jagreen@sandia.gov
[3]wascott@sandia.gov
[4]utkarsh.ayachit@kitware.com
[5]berk.geveci@kitware.com
[6]dave.demarle@kitware.com

概要

ParaView は、大容量のデータセットを並列に分析および可視化することのできる、パワフル、オープンソース、かつダウンロード後即座に使用可能なアプリケーションです。ParaView は Red Storm や ASC Purple スーパーコンピュータによるシミュレーション結果の可視化のために、サンディア国立研究所のアナリストによって日常的に使用されており、さらにその他の、世界中の何千ものユーザによっても使用されています。柔軟な設定・拡張が可能であり、またスケーラブルな設計を目指し、ParaView は Visualization Toolkit (VTK) の上に構築され、それゆえ可視化コンポーネントの迅速な適用が可能です。このチュートリアルでは、ParaView の構造および並列可視化の基礎を学びます。実践的な演習によって、読者が科学データ可視化のための ParaView の使用法の基礎を学ぶことを目的としています。このチュートリアルは、現在のスーパーコンピュータで実行されるような大規模シミュレーションを可視化するための詳細な方法、および ParaView におけるスクリプティングおよび拡張法を特に取り上げます。

目 次

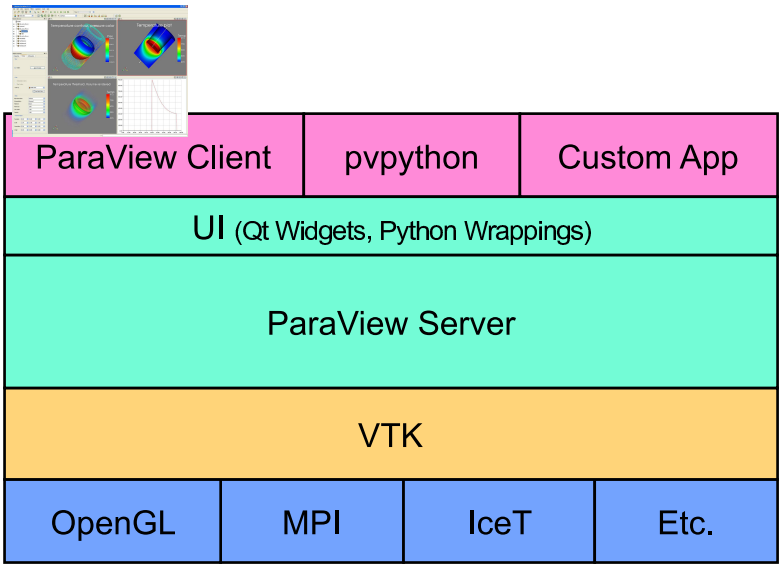
第1章　序

ParaView は、2次元および3次元データセットを可視化するためのオープンソースアプリケーションです。ParaView が扱えるデータセットの大きさは、ParaView が実行されるプラットフォームのアーキテクチャによって大きく変わります。ParaView によってサポートされるプラットフォームは、シングルプロセッサのワークステーションから、多数のプロセッサによって構成される分散メモリスーパーコンピュータやワークステーションクラスタまで、多岐に渡ります。並列コンピュータを利用することで、ParaView は巨大なデータセットを並列に処理し、その後で処理結果を集約することが出来ます。今までのところサンディア国立研究所では、最大で60億セルの構造化メッシュ、2億5千万セルの非構造メッシュ、そして数十億のセルからなる構造 AMR 格子の可視化に ParaView を使用しています。

ParaView の設計においては、以下のような多くの特徴により、他の科学データ可視化ソリューションとの差別化を図っています。

- オープンソース、スケーラブル、かつマルチプラットフォームな可視化アプリケーション。
- 大容量データセットを処理するための分散型計算手法のサポート。
- オープン、柔軟かつ直感的なユーザインターフェイス。
- オープンな規格に基づいた拡張性の高いモジュール化構造。
- 有償の保守およびサポート。

ParaView は世界中の多くの学術、公的、および商業機関によって利用されており、毎月およそ3千件ダウンロードされています。



ParaView としてユーザから見えるアプリケーションの実体は、ParaView 自体の機能を構成する何層ものライブラリ群の上に構築された、小さなクライアントに過ぎません。上の図に示すように、ParaView のほとんどの機能はライブラリとして実装されているため、ParaView の GUI をカスタムアプリケーションによって完全に置き換えることが可能です。さらに、ParaView に付属する **pvpython** アプリケーションによって、Python スクリプトを用いて可視化とポストプロセッシングを自動化することが可能です。

ParaView アプリケーションのそれぞれに対し、コードを最大限に共有化するためのユーザインターフェイス部品のライブラリが利用可能です。**ParaView サーバ・ライブラリ**によって、並列かつインタラクティブな可視化を実行するための抽象化レイヤが提供されます。ParaView サーバ・ライブラリによって、クライアントアプリケーションは ParaView が並列に実行されているか否か、またどのように並列実行されているか、といったことに関する問題の多くから解放されます。**Visualization Toolkit** (**VTK**) は、基本的な可視化およびレンダリング・アルゴリズムを提供します。VTK にはレンダリング、並列処理、ファイル入出力、並列レンダリングのような基本的な機能を提供するライブラリも含まれます。このチュートリアルでは、ParaView に付属するクライアントアプリケーションを使用して ParaView の使い方を説明しますが、ParaView 自体は、その高度に部品化された設計によって、大幅な柔軟性およびカスタマイズの可能性を有しています。

1.1 開発および資金提供

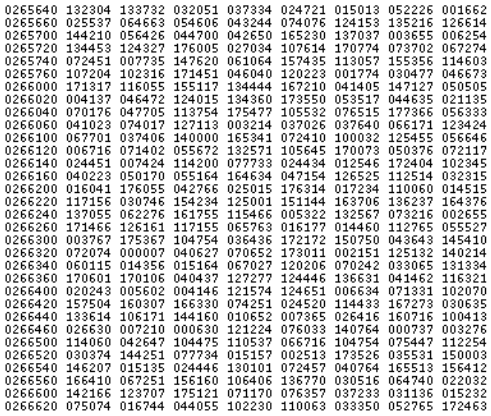
ParaView のプロジェクトは、Kitware Inc. とロスアラモス国立研究所の共同努力によって開始されました。初期の資金は、米エネルギー省 ASCI Views 計画との 3 年契約によって提供されました。最初の公開リリースである ParaView 0.6 は、2002

年 10 月にアナウンスされました。ParaView の開発は、Kitware Inc. とサンディア国立研究所、ロスアラモス国立研究所、陸軍研究所、およびその他多くの学術・政府機関との共同作業によって継続されました。

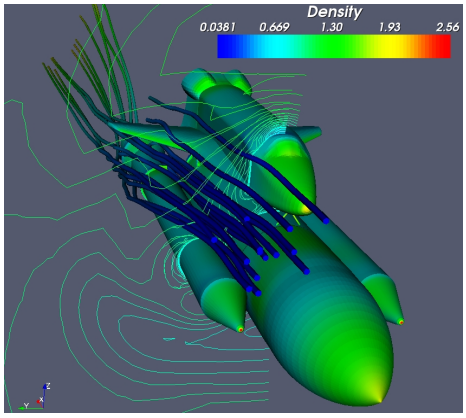
2005 年 9 月には、Kitware、サンディア国立研究所、および CSimSoft は ParaView 3.0 の開発を開始しました。この開発においては、ユーザインターフェイスをさらにユーザフレンドリに書き直し、定量解析のためのフレームワークを開発することに注力しました。ParaView3.0 は 2007 年 5 月にリリースされました。

ParaView の開発は、今日も続いています。サンディア国立研究所は、ASC 計画を通じ、ParaView の開発への資金提供を続けています。ParaView は、SciDAC Institute for Ultra-Scale Visualization (www.ultravis.org) の主要な開発プラットフォームとして統合されています。米エネルギー省は、ロスアラモス国立研究所、陸軍の中小企業技術革新制度、および ERDC との契約を通じて ParaView に資金提供しています。米国立科学財団もまた、中小企業技術革新制度によって ParaView に資金提供しています。フランス電力公社、Mirarco、石油関連企業など、その他の機関も ParaView への支援を行っています。さらに、ParaView はオープンソースプロジェクトであるため、スイス国立スーパーコンピューティングセンターのような機関も開発成果をコントリビュートしています。

1.2 可視化の基礎



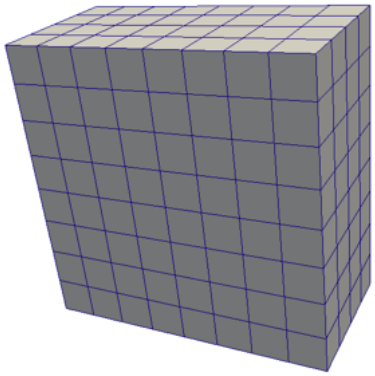
```
0265640 132304 133732 032051 037334 024721 015013 052226 001662
0265660 025537 064663 054606 043244 074076 124153 135216 126614
0265700 144210 056426 044700 042650 165230 137037 003655 006254
0265720 134453 124327 176005 027034 107614 170774 073702 067274
0265740 072451 007735 147620 061064 157435 113057 155356 114603
0265760 107204 102316 171451 046040 120223 001774 030477 046673
0266000 171317 116055 155117 134444 167210 041405 147127 050505
0266020 004137 046472 124015 134350 173550 053517 044635 021135
0266040 070176 047705 113754 175477 105532 076515 177366 056333
0266060 041023 074017 127113 003214 037026 037640 066171 123424
0266100 067701 037406 140000 165341 072410 100032 125455 056646
0266120 006716 071402 055672 132571 105645 170073 050376 072117
0266140 024451 007424 114200 077733 024434 012546 172404 102345
0266160 040223 050170 055164 164634 047154 126525 112514 032315
0266200 016041 176055 042766 025015 176314 017234 110060 014515
0266220 117156 030746 154234 125001 151144 163706 136237 164376
0266240 137055 062276 161755 115466 005322 132567 073216 002655
0266260 171466 126161 117155 065763 016177 014460 112765 055527
0266300 003767 175367 104754 036436 172172 150750 043643 145410
0266320 072074 000007 040627 070652 173011 002151 125132 140214
0266340 060115 014356 015164 067027 120206 070242 033065 131334
0266360 170601 170106 040437 127277 124446 136631 041462 116321
0266400 020243 005602 004146 121574 124651 006634 071331 102070
0266420 157504 160307 166330 074251 024520 114433 167273 030635
0266440 133614 106171 144160 010652 007365 026416 160716 100413
0266460 026630 007210 000630 121224 076033 140764 000737 003276
0266500 114060 042647 104475 110537 066716 104754 075447 112254
0266520 030374 144251 077734 015157 002513 173526 035531 150003
0266540 146207 015135 024446 130101 072457 040764 165513 156412
0266560 166410 067251 156160 106406 136770 030516 064740 022032
0266600 142166 123707 175121 071170 076357 037233 031136 015232
0266620 075074 016744 044055 102230 110063 033350 052765 172463
```



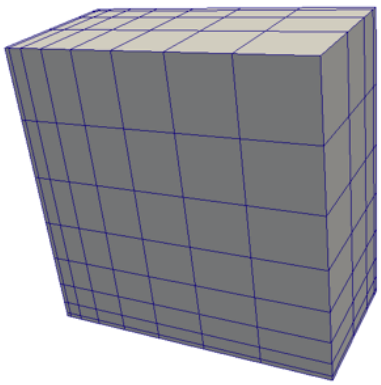
簡単に言うと、可視化の過程とは、生のデータを人間が見て理解することができる形式に変換することです。このことによって、データをより感覚的に理解することができるようになります。科学的な可視化では特に、2 次元あるいは 3 次元空間において明確な表現を有するデータを取扱います。シミュレーションのメッシュやスキャナのデータに由来するデータは、この種の分析に特に向いています。

データの可視化には、大別して 3 つの段階があります。すなわち、読込み、フィルタリング、レンダリングです。まず、データを ParaView に読み込みます。つぎに、データから特徴を生成、抽出、導出するためにデータを処理する**フィルタ**を、必要なだけ適用します。最後に、可視化された画像がデータからレンダリングされます。

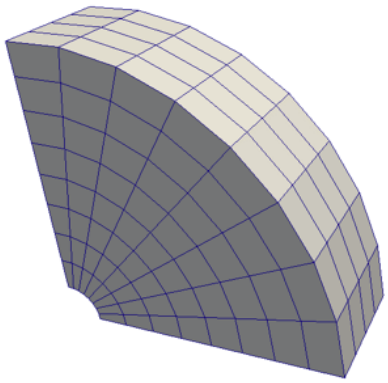
ParaView は空間的な表現としてデータを取扱うため、ParaView が使用する基本的な**データ型**はメッシュ (または格子) です。



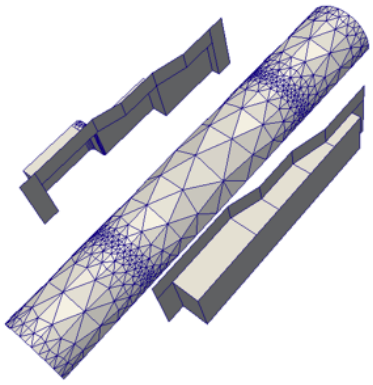
一様直線格子 (画像データ)
等間隔直線格子は 1、2、または 3 次元の配列データです。格子点は互いに直交し、各方向に等間隔に配置されます。



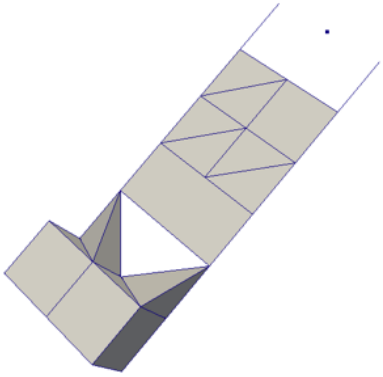
非一様直線格子 (直線格子)
等間隔直線格子と似ていますが、格子点間の距離を各軸方向に変化させることができます。



曲線格子 (構造格子)
曲線構造格子は、直線格子と同じトポロジを有します。ただし、曲線格子上の格子点は (セルどうしが重なったり、自分自身と交差しない限り) 任意の座標に置くことが出来ます。曲線格子は直線格子の比較的少ないメモリ使用量、暗黙のトポロジといった特徴を継承しつつ、メッシュの形状に大幅な自由度を与えることができます。



ポリゴン・データ (ポリ・データ)
ポリゴン・データセットは、点、直線、2 次元の任意多角形から構成されます。セル間の結合は任意であり、結合しないこともできます。
レンダリングにおける基本的なプリミティブは、ポリゴン・データによって表されます。全てのデータは、(ボリュームレンダリングを除いて) レンダリングの前に必ずポリゴン・データに変換されますが、この変換は ParaView によって自動的に行われます。



非構造格子
非構造格子データセットは、点、直線、2 次元の任意多角形、3 次元 4 面体、非線形セルから構成されます。ポリゴン・データと類似していますが、直接レンダリング出来ないような、3 次元の 4 面体や非線形セルも表現することができます。

これらの基本的なデータ型に加えて、ParaView は**マルチブロック**データもサポー

トしています。データセットがグループ化されたときや複数のブロックから成るファイルが読み込まれた時には、必ずマルチブロックのデータセットが生成されます。ParaView はさらに、**階層化適応メッシュ分割**(**AMR**)、**階層化一様 AMR**、**八分木**データセットを表現するためのデータ型を有します。

1.3 さらなる情報

様々なところから、ParaView に関するさらなる情報を入手することができます。初心者には、ParaView アプリケーションの  ボタンをクリックすることでアクセス可能な、オンライン・ヘルプがあります。このオンライン・ヘルプに加え、Amy Henderson Squillacote によって書かれ、Kitware から入手可能な *The ParaView Guide* は、ParaView の全てに関するガイドとして役立ちます。

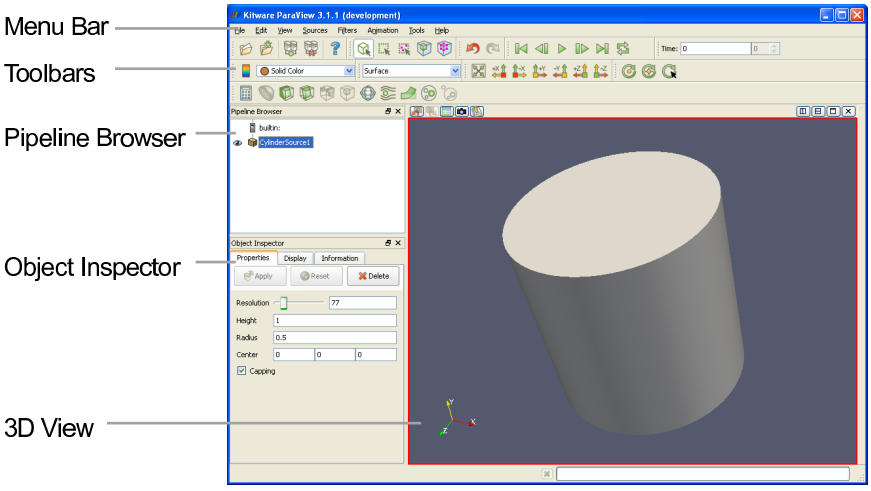
ParaView のウエブページwww.paraview.orgもまた、ParaView に関するさらなる情報を入手する絶好のサイトです。このサイトから、メーリングリスト、Wiki、よくある質問集、さらには有償サポートサービスへのリンクを辿ることができます。

第2章　基本的な使用法

それでは、ParaView を使ってみましょう。ここから先の内容を進めるには、ParaView をインストールして頂く必要があります。既に ParaView をお持ちでなければ、www.paraview.orgからダウンロードすることができます (ダウンロードへのリンクをクリックして下さい)。ParaView は、大抵のアプリケーションと同じようにして起動します。Windows では、スタート・メニューから起動します。Macintosh では、インストールしたアプリケーションバンドルを開いて下さい。Linux では、コマンドプロンプトから `paraview` を実行して下さい (パスが設定されている必要があります)。

このチュートリアルの例では、http://www.paraview.org/Wiki/SC08_ParaView_Tutorial から入手可能なデータを使用します。このデータを、簡単にアクセス出来る適当なディレクトリにインストールして下さい。このチュートリアルによってファイルを読み込むよう指示された時は、このデータをインストールしたディレクトリから読み込んで下さい。

2.1　ユーザインターフェイス



ParaView の GUI は、実行されているプラットフォームにおける形式 (ルック・アンド・フィール) に従いますが、基本的には全てのプラットフォームで同じように操作します。ここに示したレイアウトは、ParaView が最初に起動された時のデフォルトのレイアウトです。GUI は以下の要素から構成されます。

メニューバー 他のプログラムと同様に、メニューバーから大部分の機能を利用することが可能です。

ツールバー ツールバーによって、ParaView の中でも最も一般的に使用される機能を素早く利用することができます。

Pipeline Browser (パイプライン・ブラウザ) ParaView は、パイプラインによってデータの読み込みとフィルタリングを管理します。パイプライン・ブラウザによって、パイプラインの構造を表示し、パイプライン・オブジェクトを選択することができます。ParaView 3 のための再設計によって、パイプライン・ブラウザは、パイプライン構造をインデントして表現する便利なリストとなりました。

Object Inspector (オブジェクト・インスペクタ) オブジェクト・インスペクタでは、現在選択されているパイプライン・オブジェクトの設定を表示および変更することができます。オブジェクト・インスペクタには、3 つのタブがあります。**Properties** (プロパティ) タブには、オブジェクトの状態に関して設定可能なオプションが提示されます。**Display** (ディスプレイ) タブには、ビュー画面でのオブジェクトの表現方法に関するオプションが提示されます。**Information** (情報) タブには、パイプライン・オブジェクトによって生成されたデータに関する基本的統計情報が表示されます。

3D View (3D ビュー) GUI の残りの部分はデータの提示に使用され、ここでデータを表示・操作・分析することができます。この部分は、データの幾何的表現を提示する 3D View で初期化されます。

ここで留意すべきは、GUI のレイアウトは大幅に設定変更が可能であり、そのためウインドウの見た目の変更が容易であることです。ツールバーは各所に移動し、あるいは隠すこともできます。ツールバーの表示を変更するには、View → Toolbars メニューを使用して下さい。パイプライン・ブラウザとオブジェクト・インスペクタは、いずれも**ドック可能な**ウインドウです。すなわち、これらの要素は GUI の中を各所に移動したり、フローティング・ウインドウとして切離したり、完全に隠すことができます。これらの 2 つのウインドウは ParaView の操作に重要なので、これらを隠した後に再度必要となった場合は、View メニューから表示することができます。

2.2 ソース

ParaView にデータを入力するには、2 とおりの方法があります。すなわち、データをファイルから読み込むか、または**ソース**オブジェクトによって生成する方法です。全てのソースは Source メニューにあります。ソースによって例えば、ビューに注記を

追加することもできますし、ParaView の機能を (実際に動作させながら) 調べたい時にも非常に便利です。簡単な例から始めましょう。Source メニューから、Cylinder (シリンダ) を選択して下さい。そうすると、パイプライン・ブラウザに CylinderSource1 という項目が追加され、選択された状態となります。さらに、オブジェクト・インスペクタには、シリンダ・ソースのプロパティが表示されています。ここではデフォルトの設定で良いので、Apply ボタン 

をクリックして下さい。

Apply をクリックすると、シリンダ・オブジェクトが 3D ビューウインドウの右側に表示されます。この 3D ビューは 3D ビュー上でマウスドラッグすることで、操作することができます。回転、パン、ズーム等の操作を行うには、異なるマウスのボタン (左、中、右) を押しながらのドラッグを試してみて下さい。また、シフト、Ctrl、Alt キーなどのモディファイアキーを押しながらマウスボタンを押してみて下さい。

シリンダ・オブジェクトは真の円筒ではなく、多角形**小面**による円筒の近似であることにお気づきと思います。シリンダ・ソースのデフォルト設定では、わずか 6 小面からなる非常に粗い近似が生成されます。(実際のところ、このオブジェクトは円筒というよりプリズムに近く見えます。) より円筒らしい表現が必要であれば、Resolution のパラメータ値を増やすことで作成することができます。

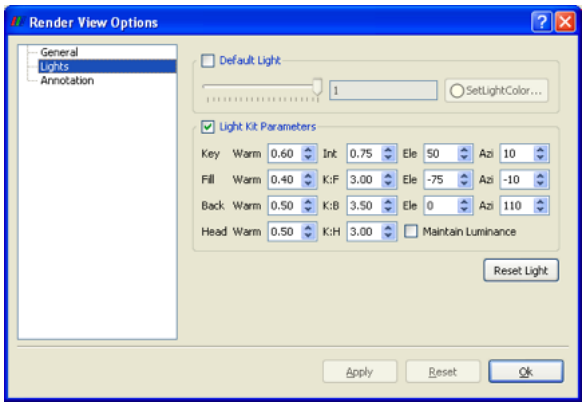
スライダもしくはテキスト入力フィールドを使用して、解像度を 50、またはそれ以上にして下さい。ここで、Apply ボタン 

が再び緑色 (または、Mac では青色) になったことに留意して下さい。これは、オブジェクト・インスペクタに対して行った変更がすぐには適用されないためです。ハイライト状態のボタンは、パイプライン・オブジェクトのいずれかの設定と、表示されているデータが整合していないことを示しています。Apply ボタンをクリックするとこれらの変更が適用され、Reset ボタン 

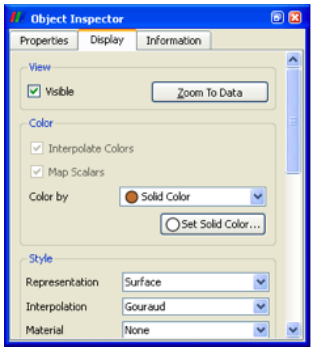
は、設定を最後に適用された状態に戻します。ここでは、Apply ボタンをクリックして下さい。真の円筒とほとんど見分けがつかない程、解像度が変わる筈です。

ここで ParaView 3 の新機能である、ツールバーの取消し  および再実行  ボタンを紹介します。データの可視化は多くの場合試行錯誤の過程であり、一つ前の状態に戻れることが有用な状況があります。実際、データを作成する前の時点にまで取消しで戻って、再実行を行うことができます。この操作を、ここで試してみて下さい。さらに、カメラ操作の取消し  、カメラ操作の再実行  という特別なボタンもあります。これらによって、今までに試行したカメラのアングルを行ったり来たりすることができ、マウス操作の誤りによって完璧なカメラアングルが崩れることを心配する必要がなくなります。

また、オブジェクトがどのようにレンダリングされるかを選択するための設定も、多数あります。3D ビューの上には  ボタンがあり、レンダリングの設定を変更することができます。これをクリックするとダイアログボックスが開き、背景色、ライティング、注釈などを変更することができます。



オブジェクト・インスペクタの Display タブにも注目して下さい。このタブには、選択したオブジェクトのレンダリング設定があります。例えば表示・非表示の切替え、表示色、データセットの表現方法などを設定できます。利便性のため、これらのビュー設定の幾つかとオブジェクトの表示設定は、ParaView の GUI 上の他の場所 (ツールバー) にも存在します。



シリンダ・ソースについては以上です。オブジェクト・インスペクタの Properties タブを選択し、削除  をクリックすることで、パイプライン・オブジェクトを削除することができます。

2.3 データの読込み

ParaView の GUI はいくらか練習しましたので、実際のデータを読込んでみましょう。ご想像のとおり、File メニューの最初の項目に Open コマンドがあり、さらにツールバーにもファイルを開くためのボタンがあります。ParaView は多くのファイル形式をサポートし、ファイル形式のリストは新たな形式が追加される度に長くなっています。以下は現在のところ利用可能な読込みモジュールです。

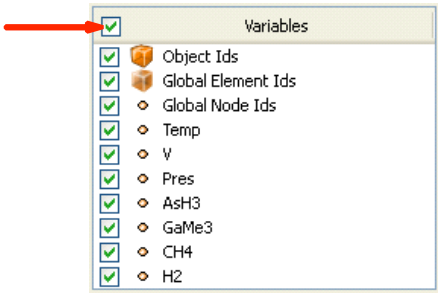
- ParaView Data (.pvd)
- VTK (.vtp, .vtu, .vti, .vts, .vtr)
- VTK Multi Block (.vtm, .vtmb, .vtmg, .vthd, .vthb)

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

モジュール化されたParaViewの設計によって、新たなVTKの読込みモジュールのParaViewへの統合が容易になっています。従って、新たなファイルフォーマットがサポートされていないか頻繁にチェックされることをお勧めします。もしお探しのファイル読込みモジュールがParaViewに含まれていないようであれば、ParaViewメーリングリスト (paraview@paraview.org) で調べてみて下さい。ParaViewからは見えないものの、VTKには含まれており、容易に追加出来るファイル読込みモジュールも多数あります。さらにはVTKフレームワークへの組み込みが可能でありながら、VTKにはまだ組み込まれていない読み込みモジュールも多数あります。誰かがあなたの必要な読込みモジュールをそのような形で持っており、快く提供してくれるかもしれません[1] 。

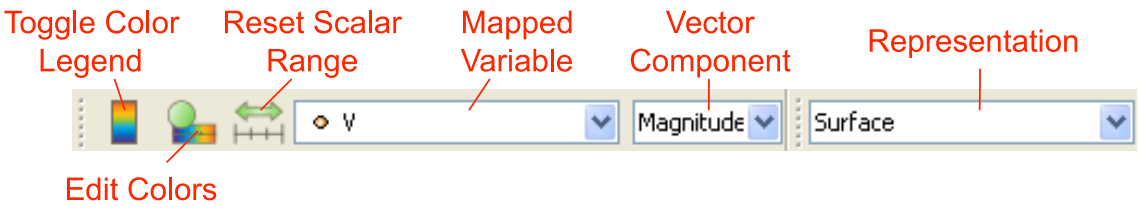
それでは、初めてのファイルを開いてみましょう。Open ツールバー (もしくはメニュー項目) をクリックし、disk_out_ref.ex2 を開いて下さい。ファイルを開く操作は2段階の操作であり、この段階では、まだ読込んだデータは見られないことに注意して下さい。その代わり、オブジェクト・インスペクタに、どのようにデータを読込むかを指定するための設定が提示されています。

[1]訳注: 例えば、訳者の一人の大嶋は、以下でOpenFOAM-1.5以降に対応したOpenFOAMフォーマットの並列読込みモジュールを公開しています (2009年8月現在)。
http://openfoamwiki.net/index.php/Contrib_Parallelized_Native_OpenFOAM_Reader_for_ParaView

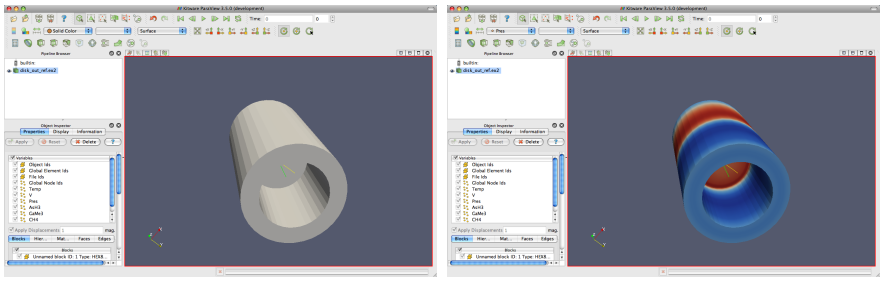


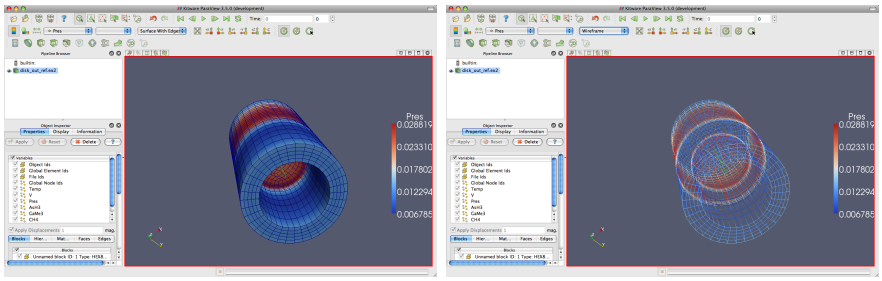
Variables (変数) リストのヘッダ (最上部) にあるチェックボックスをクリックし、全ての変数が読込まれるようにして下さい。これは小さいデータセットですので、メモリに読み込むデータ量が過大になることを心配する必要はありません。全ての変数を選択したら、 Apply をクリックして全てのデータを読込んで下さい。データが読込まれたら、一方がくり抜かれた円筒のような形状が表示されます。このデータは、加熱された回転ディスク周囲の気流シミュレーションの結果です。表示されているメッシュはディスク周りの空気で、シミュレーションの領域境界が円筒形になっています。真ん中のくり抜かれた領域は、もしこのシミュレーションのためにこの部分もメッシングされていれば、加熱されたディスクが存在するはずの場所です。

データのフィルタリングに進む前に、データの表現方法の幾つかを簡単に見ておきましょう。データの表現に関する最も一般的な設定は、2 つのツールバーに配置されています。



データの表現方法を幾つか試してみて下さい。変数選択 (上図の Mapped Variable) を使用して、表面を Pres 変数で色付けしてみて下さい。次に色の凡例表示 (上図の Toggle Color Legend) をオンにして、実際の圧力の値を見てみて下さい。メッシュの構造を見るには、表現方法 (上図の Representation) を Surface With Edges にします。Wireframe 表現にすると、セル構造とメッシュの内部を見ることができます。





2.4 フィルタ

これまでのところ、とりあえずはデータの読込みに成功し、そのデータの情報を幾らか見ることができました。つまり、メッシュの基本的な構造を表示し、メッシュの表面に何らかのデータをマッピングできるようになりました。しかしながら、これから見るように、データの表面をただ眺めるだけでは判らない興味深い特徴が、このデータには多数あります。(スカラーやベクトルのような) 異なった型の多くの変数が、このメッシュには関連づけられています。さらに、メッシュはソリッド (中実) なモデルであることに注意して下さい。興味深い情報の多くは、内部に存在しています。

フィルタを適用することで、データに関してさらに多くのことを発見できます。フィルタとは、データから特徴を生成、抽出、または導出するためにデータを処理する機能ユニットです。フィルタは読込みモジュール、ソース、あるいは他のフィルタに接続され、それらのデータに対し何らかの形で変更を加えます。これらの互いに接続されたフィルタによって、**可視化パイプライン**が形成されます。ParaView では、非常に多数のフィルタが利用可能です。以下は対応するフィルタ・ツールバー上のアイコンをクリックすることで利用出来る、最も一般的なフィルタです。

Calculator (電卓) 格子点またはセル毎に、ユーザによって定義された数式を評価します。

Contour (コンター) スカラー場がユーザによって指定された値に等しくなるような点、曲線、面を抽出します。この面は**等値面**とも呼ばれます。

Clip (クリップ) 形状を半空間で切断します。その結果、ユーザによって定義された面のいずれか一方の側の形状が削除されます。

Slice (スライス) 形状を面で切断します。その結果はクリップと同様ですが、面上の形状だけが残されます。

Threshold (しきい値) スカラー場の指定された範囲に存在するセルを抽出します。

Extract Subset (サブセットの抽出) 抽出すべきボリュームあるいは間引き率を指定し、格子のサブセットを抽出します。

Glyph (グリフ) グリフ、すなわち単純な形状をメッシュの各格子点に配置します。グリフはベクトルによって方向を指定し、ベクトルまたはスカラーによってスケーリングすることができます。

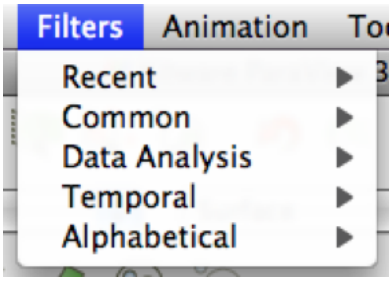
Stream Tracer (流線追跡) ベクトル場にシード点を配置し、(定常の) ベクトル場をそれらのシード点から追跡します。

Warp (vector) (ワープ (ベクトル)) メッシュの各格子点を、与えられたベクトル場で変形させます。

Group Datasets (データセットのグループ化) 複数のパイプライン・オブジェクトの出力を、一つのマルチブロック・データセットに統合します。

Extract Group (グループの抽出) マルチブロック・データセットを構成する要素 (ブロック) を抽出します。

これらの 11 種のフィルタは、ParaView で利用可能なもののごく一部の例です。Filters メニューには、データ処理のためのフィルタがさらに多数存在します。ParaView からは現在のところ 80 以上のフィルタが利用可能であるため、簡単にフィルタを見つけられるよう、Filters メニューは以下のようにサブメニューに整理されています[2]。



Recent (最近使ったフィルタ) 最も最近使われたものが一番上に来るようソートされた、最も最近使われたフィルタのリストです。

Common (一般的) 最も一般的なフィルタです。これはフィルタ・ツールバーにリストされているフィルタと同じで、前述のとおりです。

Data Analysis (データ分析) 定量的な値をデータから取り出すためのフィルタです。これらのフィルタはメッシュ上でデータを計算したり、メッシュから要素を抽出したり、データをプロットするのに使用します。

Temporal (時間依存型) 時間によって変化するデータを分析、あるいは加工するフィルタです。全てのフィルタは各時刻のスナップショットに対して実行されるため、時間によって変化するデータに対して使用可能です。しかしながらこの分類のフィルタは特に、利用可能な時間範囲を走査し、データが時間とともにどのように変化しているかを調べることができます。

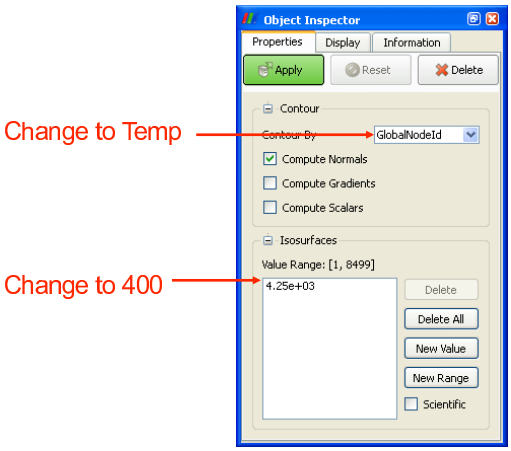
[2]訳注: フィルタが多すぎて探しにくい場合は、Ctrl + Space (Mac では Option + Space) のキー操作によって、フィルタ名で検索することもできます。

Alphabetical (アルファベット順) 全ての利用可能なフィルタのアルファベット順のリストです。あるフィルタがどこにあるか判らなければ、このリストには確実に存在します。また、このリストにしか存在しないフィルタも多数存在します。

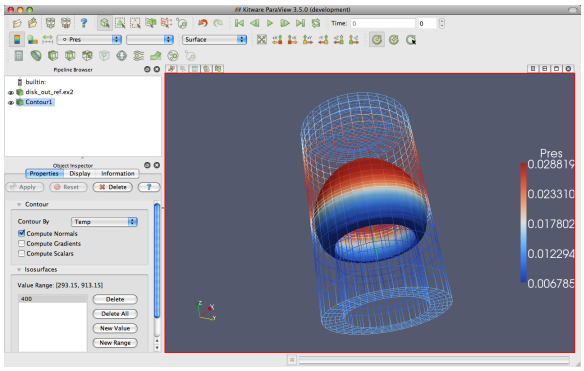
フィルタの一部はメニュー上でグレーになっていて、選択できないことにお気づきかと思います。多くのフィルタは特定の型のデータにしか適用できないため、常に使用出来る訳ではありません。ParaView はそのようなフィルタをメニューおよびツールバーから選択不可にし、それらのフィルタが使用不可であることを示し (また、それを強制し) ます。

このチュートリアルでは多くのフィルタを使用しますが、それでも全てを試せる訳ではありません。それぞれのフィルタのさらなる情報については、*The ParaView Guide* をご覧下さい。

それでは最初のフィルタを適用しましょう。パイプライン・ブラウザ上で disk_out_ref.ex2 が選択されていることを確認し、フィルタ・ツールバーまたは Filters メニューから、Contour  フィルタを選択して下さい。パイプライン・ブラウザの読込みモジュールの下に新たな項目が追加され、オブジェクト・インスペクタはフィルタの設定画面にアップデートされます。ファイルの読込みと同様に、フィルタの適用も 2 段階の操作です。フィルタの作成後、フィルタの適用前に設定を変更することができます (そしてそれはほぼ大抵の場合、必要な操作です)。



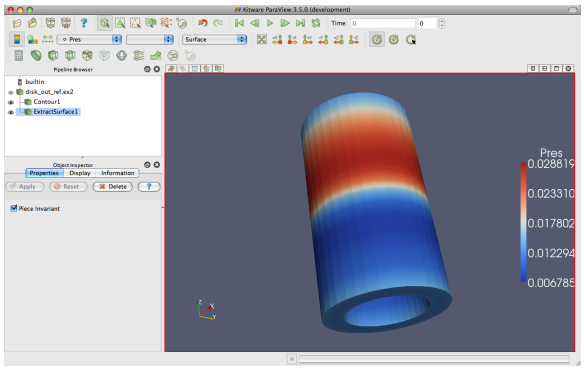
コンター・フィルタを使って、温度が 400 K の等値面を作成することにしましょう。まず、Contour By の設定を Temp 変数に変更します。つぎに、等値面の値を 400 に変更します。最後に、  をクリックします。等値面がボリュームの内部に現れる筈です。等値面はフィルタへの入力となった disk_out_ref.ex2 と同様に、圧力で色付けされています。



もう少し、フィルタを色々試してみましょう。内部にあるものの表示をしばしば妨げるメッシュ表面のワイヤフレーム表示の代わりに、表面の一部を切り取った状態にしましょう。この操作には 2 つのフィルタが必要です。すなわち、1 つ目のフィルタで表面を抽出し、2 つ目で表面の一部を切り取ります。

それでは表面を抽出するフィルタを追加しましょう。それは以下の手順で行います。

1. パイプライン・ブラウザで disk_out_ref.ex2 を選択します。
2. メニューバーから Filters → Alphabetical → Extract Surface を選択します。
3.  ボタンをクリックします。

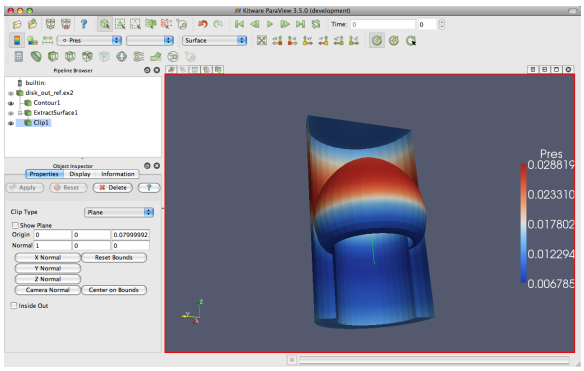


ワイヤフレームはソリッドな表面で置き換えられました。コンターが見えなくなりましたが、心配する必要はありません。表面に隠された状態で、そのまま存在しています。もしフィルタの適用後の表示に何も変化が無ければ、1 番目の disk_out_ref.ex2 を選択する操作を忘れて、違うオブジェクトにフィルタを適用したのかもしれません。もし ExtractSurface1 オブジェクトが disk_out_ref.ex2 に直接接続されていなければ、それが間違えた操作です。もしそのとおりであれば、フィルタを削除して再び操作を行うことができます。

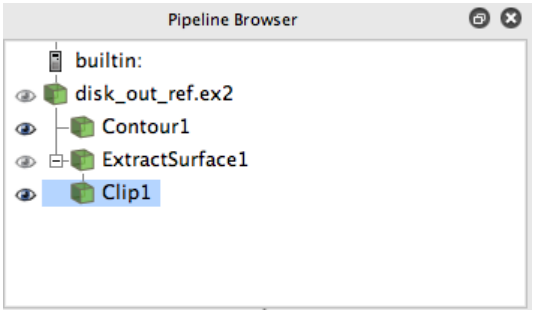
つぎに内部の等値面が見えるよう、表面を切り取りましょう。

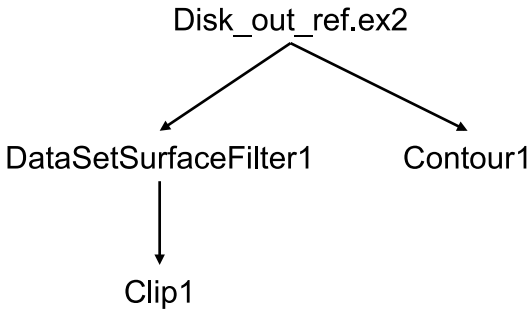
1. パイプライン・ブラウザ上で、ExtractSurface1 が選択されていることを確認します。

2. Clip  フィルタを、ツールバーまたは Filters メニューから作成します。
3. オブジェクト・インスペクタの Show Plane チェックボックス  のチェックを外します。
4.  ボタンをクリックします。



これでメッシュ表面を切り取った内側から、等値面コンターが見える筈です。コンターがはっきり見えるためには、メッシュを回転させる必要があるかもしれません。





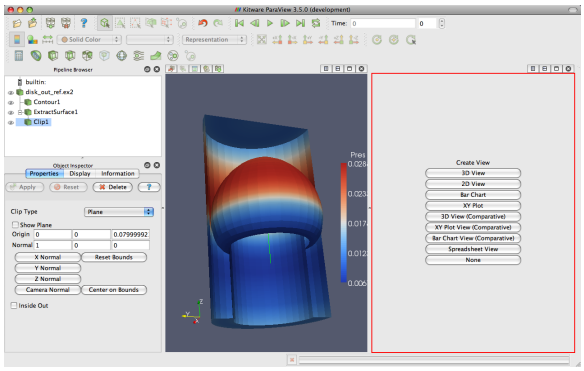
パイプラインにフィルタが幾つか追加されましたので、パイプライン・ブラウザでのこれらのフィルタの配置を見てみましょう。パイプライン・ブラウザによって、今までに作成されたパイプライン・オブジェクトは判りやすいリストとなっており、また、パイプライン・オブジェクトを選択し、それらのオブジェクトの隣にある目のアイコン  をクリックすることで**表示・非表示**を簡単に変更出来るようになっています。しかしさらに、このリスト中の項目のインデントと、インデントを辿って右側へ折れ曲がった線にも注目して下さい。これらの機能はパイプラインの**連結状態**を示しています。これは右の図の従来型のグラフと同じ情報を、ずっとコンパクトなスペースに表しています。パイプライン・オブジェクトの従来型配置の問題点は、多くのスペースが必要であり、さほど大規模でないパイプラインでさえ完全に見渡すには GUI の大部分が必要なことでした。しかし、このパイプライン・ブラウザは、完全でありながらコンパクトです。

2.5 マルチビュー

科学上の目的で時折、1 つの変数に絞って検討することがあります。しかし、多くの重要な物理現象は、互いに何らかの影響を与え合う複数の変数で記述されます。それゆえ、1 つの視点で多くの変数を表示するのは大変難しいといえます。ParaView はデータを多様な視点で表示し、それらを相互に関連付ける機能を持っており、そのような機能は複雑な可視化データを考察する上で有用です。

今までのところ、可視化では 2 つの変数に注目しています。すなわち、圧力を色のグラデーションで表し、温度による等値面を抽出して表示しています。一見、2 つの変数をきれいに配置できているようにも見えますが、それらを分かりやすく関連付けることは出来ていません。複数の視点を使うことで、両者の関係を分かりやすくすることが出来ます。各視点がデータの個別な特徴を表示し、さらにそれらを組み合わせることで、より理解しやすい可視化を提示することができます。

各ビューの上部には小さなツールバーがあり、このツールバーの右側にビューを作成したり、削除するためのボタンがあります。全部で 4 つのボタンがあります。 もしくは ボタンを押すことで、既に有るビューをそれぞれ水平もしくは垂直に分割して新たなビューを作成することが出来ます。 ボタンはビューを削除し、そのビューによって占められていたスペースは隣接するビューで占められます。 ボタンによって、選択したビューを一時的にビュー画面いっぱいに拡大して表示することができ、 で元の状態に戻ります。ここで、 ボタンを押してください。



既存のビューが 2 つに分割され、右側に空白のビューが作成されました。ここにデータを新しく可視化して表示させることが出来ます。右側のビューは、赤い線で囲われていることに注意して下さい。これは、このビューが**アクティブ・ビュー**であることを表しています。パイプライン・ブラウザやオブジェクト・インスペクタなどの、一つのビューに関する情報を表示したり操作を行うためのウイジェット (画面要素) は、このアクティブ・ビューに対して働きます。この新しいビューに、メッシュの温度分布を可視化していきます。

1. (右側の) 新しく生成した空白のビューが、赤い線で囲われていることを確認してください。どのビューでも、ビューをクリックすることでアクティブにすることが出来ます。

2. パイプライン・ブラウザ上の disk_out_ref.ex2 の隣にある眼のマーク  をクリックして、元のデータを表示してください。

3. パイプライン・ブラウザ上で disk_out_ref.ex2 を選択し、ツールバーで表示する変数を Solid Color から Temp に変え、メッシュ表面を温度分布で色付けしてください (この時点で色の凡例も表示した方が良いでしょう)。

メッシュの外側に色が表示されましたが、このままでは境界面しか見ることが出来ず、あまり興味深いものではありません。内部の温度分布を見るためにはメッシュを切り取る (クリップする) 必要があります。

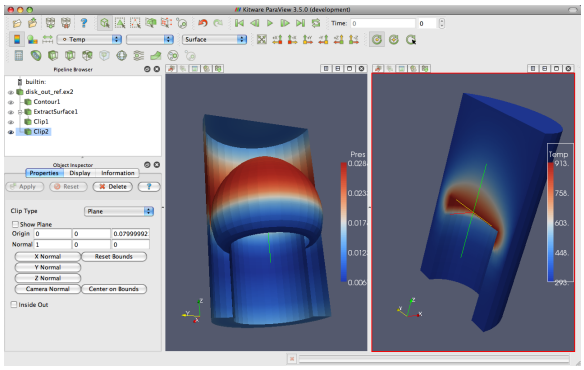
4. disk_out_ref.ex2 に Clip フィルタ  を追加します。

5. オブジェクト・インスペクタの Show Plane チェックボックス 

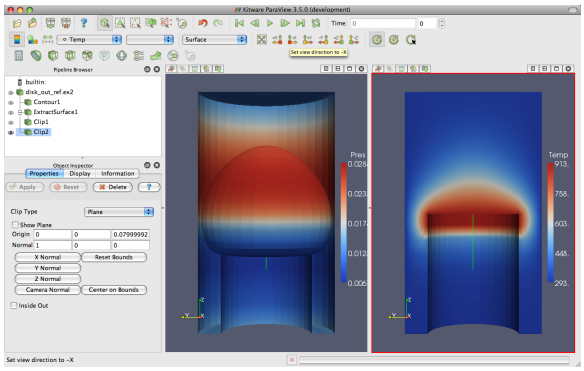
のチェックを、はずしてください。

6. 

ボタンをクリックします。

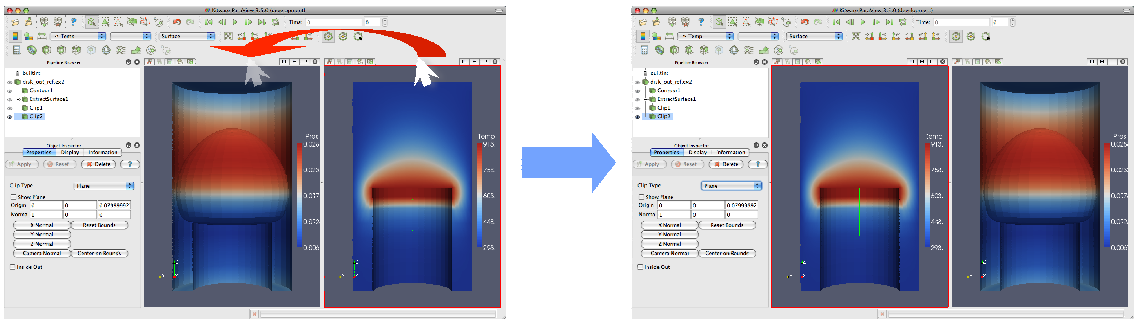


画面に 2 つのビューが出来ました。1 つは圧力の情報を、もう 1 つは温度の情報を表示しています。これらを比較したいところですが、向きが異なるため比較は困難です。片方の分布ともう片方の分布の関係を見るにはどうしたらいいでしょうか。**カメラのリンク**を追加し、2 つのビューが常に同じ視点から描画されるようにすることで、この問題は解決されます。カメラをリンクさせることは、とても簡単に出来ます。まず 1 つのビュー上で右クリックし、ポップアップメニューから Link Camera... を選択します。(もし使用している OS が Mac でマウスに右クリックがない場合、同様の操作をメニューオプションの Tools → Add Camera Link... で行うことができます。) 次に別のビューをクリックします。これで、2 つのカメラがリンクされました。片方を動かすと、もう一方も同じように動きます。

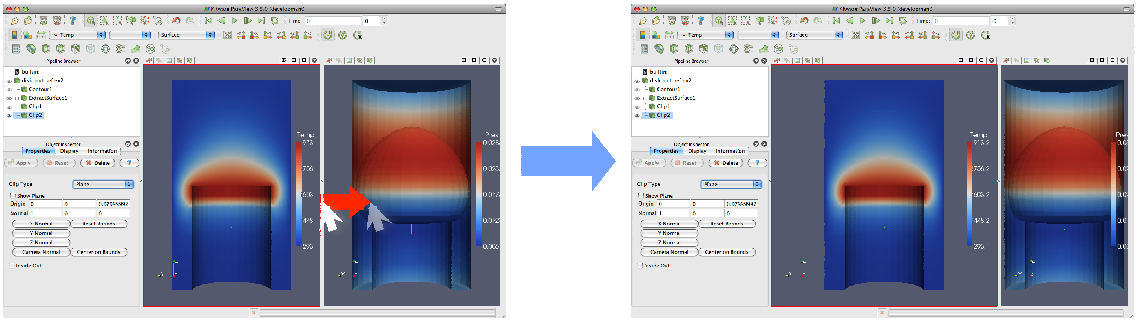


カメラがリンクされたことで、2つのビューを比較することが出来ます。 ボタンをクリックして、断面を真上から見てください。加熱されたディスクとの境界面で、温度が最も高くなっていることが分かります。それだけでは、特筆するようなことではありません。空気温度は熱源近くで最大となり、離れるに従って小さくなっていくことは予想できます。しかし圧力は、同じ地点で最大とならないことが分かります。空気の圧力は、ディスクの上の境界面からやや離れた点で最大となっています。この結果を基に、今回の物理現象について興味深い仮説を立てることが出来ます。空気の圧力には2つの力が作用していることが考えられます。1つ目の力は、上方の空気が下方の空気を押さえつける働きをする、重力による作用です。2つ目の力は、熱せられた空気の密度が小さくなり、上方に昇ろうとする浮力による作用です。空気の圧力が最大値を取る位置から、これらの2つの力が釣り合う点が判ります。このように温度と圧力を同時に観察しなければ、この結論は得られません。

もちろん ParaView の複数視点の機能は、同時に2つ以上の視点も表示できます。各々のビューには1セットずつ、複数視点の為の分割ボタンがあります。ビュー分割ボタン を使うことでさらにビューを作成し、ワークスペースを分割することが出来ます。そして分割したビューは ボタンで、いつでも削除することが出来ます。



各ビューの表示位置も固定ではありません。ビューのツールバー上 (ボタンでない部分) をクリックし、マウスのボタンを押したまま別のビューのツールバーの上にドラッグすることで、2つのビューの表示位置を入れ替えることが出来ます。この操作によって、2つのビューは即座に入れ替わります。



2つのビューの間をクリックし、マウスのボタンを押したままどちらか一方のビューの方向にドラッグすることで、ビューのサイズを変更することも出来ます。ビューの境界はマウスの動きに従って動き、ビューの大きさもそれに合わせて調整されます。

2.6 更なる検討

このシミュレーション結果から、他にどのような結論を導けるでしょうか。シミュレーションは、熱せられた回転ディスク上の空気の流れを表す速度場も算出しています。ParaView を使って、空気の流れを確認します。

既に作成した2つのビューを残しておくために、新しいビューを作成します。 でどちらかのビューを縦に分割し、新しいビューに集中するため、新しいビューを で最大化しておきます。パイプライン・ブラウザの disk_out_ref.ex2 の隣にある眼のマーク をクリックして、元のデータセットを表示します。以下の手順で空気の流れを可視化します。

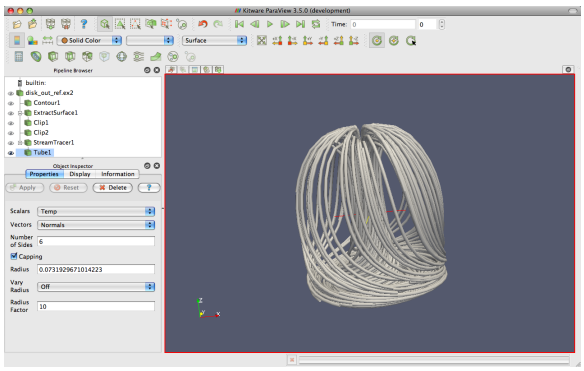
1. パイプライン・ブラウザ上で disk_out_ref.ex2 を選択します。
2. disk_out_ref.ex2 に流線追跡フィルタ を追加します。
3. デフォルト設定のまま、 Apply ボタンをクリックします。

メッシュの表面に代わって、渦巻いた線が表示されます。新しい形状は、その前の形状と中心位置が異なります。**カメラのリセット** コマンドを使うことで、即座に新しい形状の中心位置に視点を合わせることが出来ます。このコマンドは、表示されている形状を現在のビューに収め、中心を移動させると共に、回転の中心を表示されている形状の中心に一致させます。

多くの線が互いに接するように表示され、また陰影もないため、見分け難くなっています。線は1次元の構造ですが、陰影を表現するには2次元の面が必要です。チューブ・フィルタを使用することで、流線の周りに2次元の面を生成することができます。

4. メニューバーから、Filters → Alphabetical → Tube を選択します。

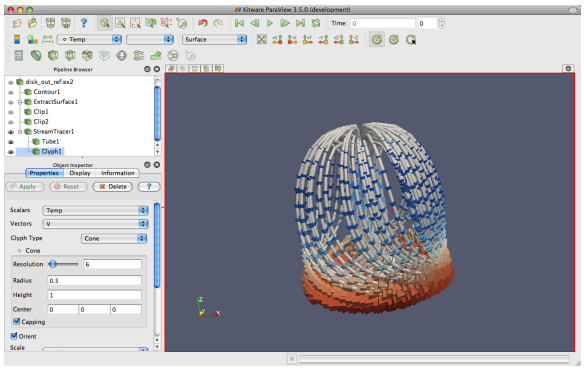
5. Apply ボタンを押します。



流線が鮮明に表示されるようになりました。流線を側面から見てみると、空気が熱せられると上昇し、冷えると下降することによる円形の対流を見て取れます。流線を回転させて、Z 軸を見下ろすようにして熱せられた円盤がある筈の辺りを見ると、回転するディスクとの摩擦によって空気が円形に流動しているのが分かります。

より高度な表示をしてみましょう。グリフを流線に追加することで、流れの方向と大きさを表示することができます。

1. パイプライン上で StreamTracer1 を選択します。
2. グリフ・フィルタ  を StreamTracer1 に追加します。
3. オブジェクト・インスペクタの Vectors の設定 (上から 2 番目の設定) を V に変更します。
4. オブジェクト・インスペクタの Glyph Type の設定 (上から 3 番目の設定) を Cone に変更します。
5. Apply ボタンを押します。
6. Temp 変数によって、グリフを色付けします。



このようにして、流線がたくさんの矢印で表示されるようになります。矢印の向きは速度の方向を示し、大きさは速度の大きさに比例しています。この新しい表示方法を用いて、次の問題に答えてみましょう。

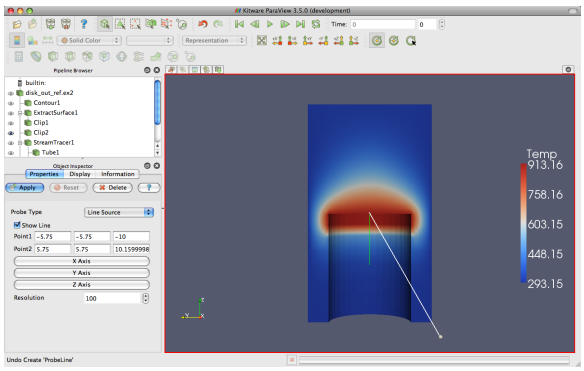
- 空気の速度が最大となるのはどの地点ですか？ディスクの近く、もしくは遠くですか？ディスクの中央付近、それとも周縁付近ですか？
- 円盤はどちらの向きに回転していますか？
- ディスクの表面上では、空気はディスクの中心に向かって流れていますか？それとも周縁部分に向かって流れていますか？

以上を完了したら、ビューのツールバーにある を押すことで、全てのビューを元に戻すことが出来ます。新しいビュー上で右クリックし、Link Camera... を選択し、他の 2 つのビューのどちらかとカメラをリンクさせます。これで、3 つのビューすべてのカメラをリンクさせることが出来ました。

2.7 プロット

ParaView のプロット機能はデータを掘り下げ、定量的な解析を可能にします。プロットは基本的にフィルタによって作成されます。プロットを作成するためのフィルタは、全て Filters メニューの下の Data Analysis サブメニューにあります。空間内の線分上におけるメッシュの場の値をプロットするフィルタを作成してみましょう。

1. パイプライン・ブラウザ上の disk_out_ref.ex2 をクリックし、アクティブな状態にします。
2. メニューバーから Filters → Data Analysis → Plot Over Line を選択します。



アクティブなビューに、両端に球の有る直線が、表示されているデータを貫く形で描画されます。これらの球のどちらかをマウスでドラッグすることで、3D ビュー内を移動させることが出来ます。画面内の球を動かすたびに、オブジェクト・イン

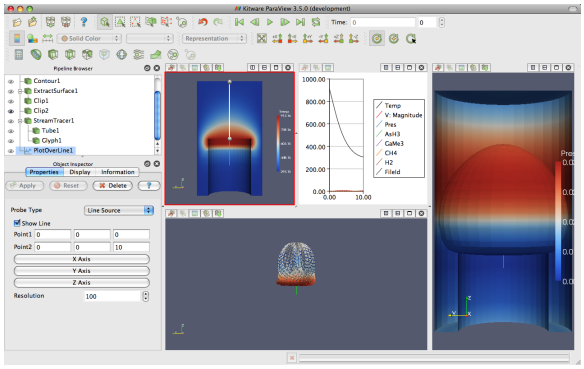
スペクタの項目の幾つかも変化することがわかります。目的の場所にマウスポインタを移動させ、p キーを押すことでも球を移動させることが出来ます。この場合、マウスポインタの位置に存在する面に、球を順次移動させることになります。この方法で球を移動させることが出来るのは、ソリッドレンダリングされた面に対してのみであることに気をつけてください。ボリュームレンダリングに対しては、この方法は使えません。

この表現は **3D ウィジェット**と呼ばれています。なぜなら、これは 3 次元空間内で操作される GUI コンポーネントだからです。ParaView には、多くの 3D ウィジェットの例があります。特にこのライン・ウイジェットは、空間内で線分を指定するのに使用されます。他にも、点や面を指定するためのウィジェットがあります。

3. 線分がディスクの基部からメッシュ全体の頂部へ垂直に結ばれるように、マウスを使うか、p キーを押すか、もしくはオブジェクト・インスペクタで数値指定をして調整してください。

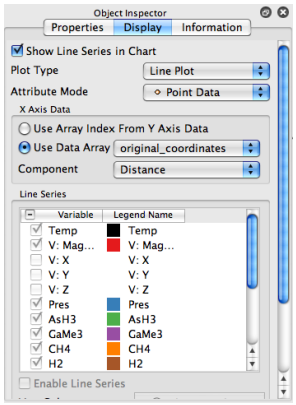
4. 線分を目的の位置に配置できたら、 

ボタンをクリックします。

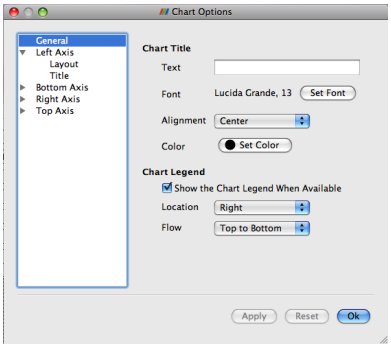


プロットの際に行える操作がいくつかあります。マウスの中央ボタンをクリックした状態で上下にドラッグすると、拡大・縮小を行う事ができます。右ボタンでドラッグすると、ドラッグ範囲へのズームができます。左ボタンでドラッグすると、プロット画面をスクロールすることができます。  コマンドを使うことで、プロットの全体像が画面に収まるようにすることも出来ます。

3D レンダリングと同様に、プロットはビューであると考えられます。どちらも別々の方法によってではありますが、データの表現方法を提供しています。そのため、3D ビューと同様の操作をプロットに対して行うことが出来ます。オブジェクト・インスペクタの Display タブをみると、色、線のスタイル、ベクトル成分、凡例など、プロットの線それぞれの表現方法に関する様々な設定があることがわかります。



プロット・ビュー上端の ボタンによって、ラベル、凡例、軸の数値範囲など、プロット全体の設定を変更するためのダイアログを呼び出すことも出来ます。



他のビューと同じように、File → Save Screenshot を選ぶことで、プロットの画像を保存することが出来ます。さらに、プロットをレポートやその他の文書で使いやすいように、PDF フォーマットのベクトルグラフィックスで保存することも出来ます。他のビューと同様に、プロットを移動させることも出来ます。

こういった特徴を利用することで、プロットからより多くの情報を引き出すことが出来ます。例えば、プロットを使って圧力と温度の値を比較することが出来ます。

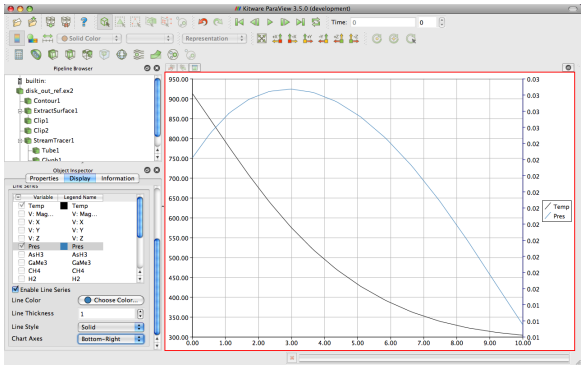
1. GUI 上でプロットを移動させたい場所を選択し、分割、削除、サイズ変更、位置の変更を使って、プロットをその場所に移動させます。

2. プロットをアクティブな状態にし、Display タブで Temp と Pres 以外の変数を全てオフにします。

Temp と Pres の変数は、異なる単位を持ちます。この 2 つを同じスケールで比べるのは、有用ではありません。したがって、同一のプロット上で各変数を各々のスケールで表示し、比較することも出来ます。ParaView の直線プロットでは、左軸と右軸それぞれに異なったスケールを設定することができ、各変数を別々の軸にしたがってスケールさせることができます。

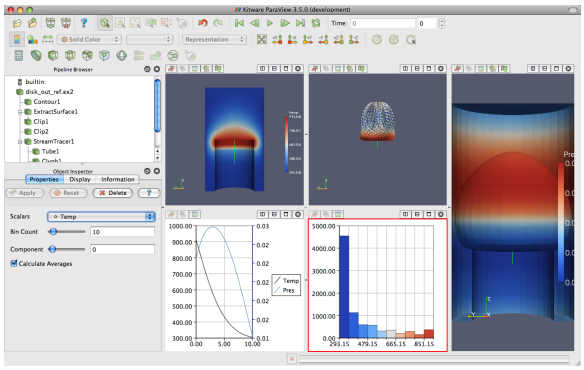
3. Display タブ上で Pres を選択します。

4. Chart Axis を Bottom - Right に変更します。



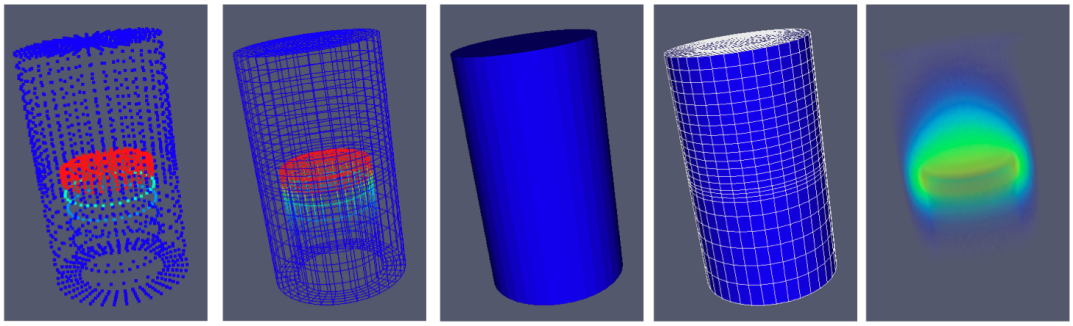
このプロットから、2.5 節で行った考察が正しいことが確認できます。温度がプレートの表面で最大値を取り、そこから離れるに従って下がるのに対し、圧力は少し上昇してから下降し始めます。さらに、圧力が最大値をとるのは (そして、空気中で力が釣り合うのは) 3 単位長さだけ離れた位置であることもわかります。

ParaView は、任意の数の異なった種類のビューに対応しています。これによって研究者や開発者は、新しい視点でデータを見ることが出来ます。さらにもう一つのビューの例を示すために、パイプライン・ブラウザの disk_out_ref.ex2 を選択し、さらに Filters → Data Analysis → Histogram  を選択してみましょう。温度のヒストグラムを作成し、  ボタンを押します。



2.8 ボリューム・レンダリング

ParaView には、データの表現方法がいくつかあります。サーフェス、ワイヤフレーム、およびそれらの組合せといった、いくつかの例を既に見てきました。ParaView はさらに、データセット表面上の点、もしくは単純にバウンディング・ボックスを表示することも出来ます。



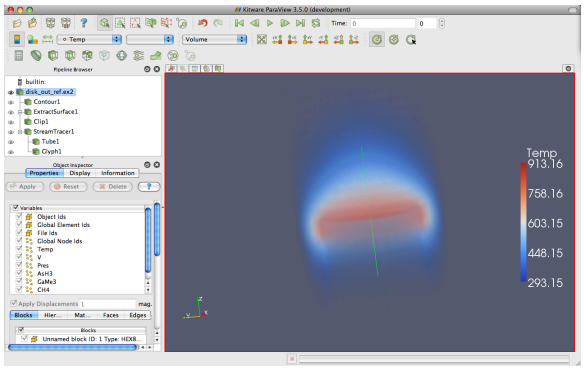
点　ワイヤフレーム　サーフェス　サーフェスとエッジ　ボリューム

ボリューム・レンダリングは、ParaView 上でデータを表現する上で強力なテクニックです。ボリューム・レンダリングでは、中実なメッシュが、メッシュ内の各点における色と濃度を表すスカラー場によって、半透明の塊としてレンダリングされます。サーフェス・レンダリングとは違い、ボリューム・レンダリングでは、ボリュームを通してデータセット内部の特徴に至るまで把握することが出来ます。

ボリューム・レンダリングは、オブジェクトの表現方法を変更することで簡単に実現できます。温度の表示をボリューム・レンダリング表示に変えてみましょう。

1. クリップされたメッシュの表面に温度が表示されているビューを選択します。
2. 適用されているクリップ・フィルタを削除します。まず、パイプライン上でクリップ・フィルタを選択し、 

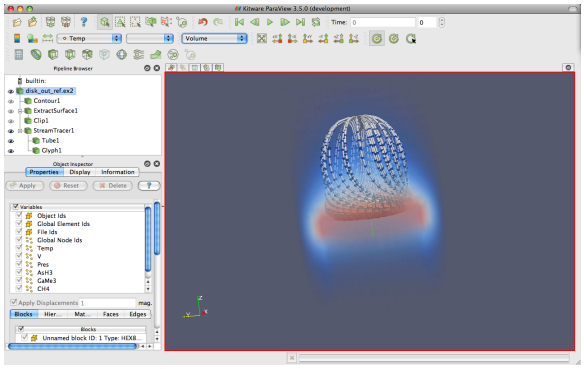
ボタンを押します。クリップされていたメッシュの代わりに、中実なメッシュ (disk_out_ref.ex2) 全体が表示されます。
3. パイプライン・ブラウザ上で、disk_out_ref.ex2 が選択されていることを確認してください。表示される変数を Temp に、表現方法を Volume にそれぞれ変更します。



不透明なメッシュの代わりに、半透明なボリュームが表示されました。画像を回転させる際に、パフォーマンスの関係でその画像が一時的に簡単な画像に置き換えられますが、これについての詳細は後ほど記します。ParaView のボリューム・レンダリングの優れた点は、他のオブジェクトのサーフェス・レンダリングと同時に使

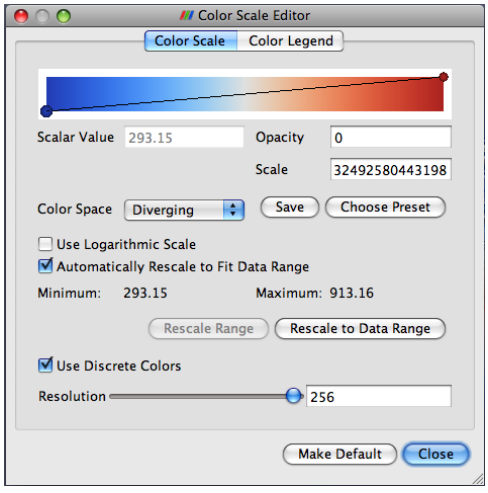
用できることです。これによって対象のボリューム・レンダリングの周囲の状況を再現したり、また他の表示と組み合わせることで、より多くの情報を伝えることが出来ます。例えば、温度のボリューム・レンダリングを流線を表示したビューと組み合わせることが出来ます。

1. 流線を表示しているビューを選択します。
2. パイプライン・ブラウザ上の disk_out_ref.ex2 の隣にある  をクリックし、元のデータを表示します。また disk_out_ref.ex2 のラベルをクリックし、そのオブジェクトが選択されている状態にします。
3. 表示される変数を Temp にし、表現方法を Volume に変更します。



ボリュームを通して、温度の表示とともに流線が表示されます。

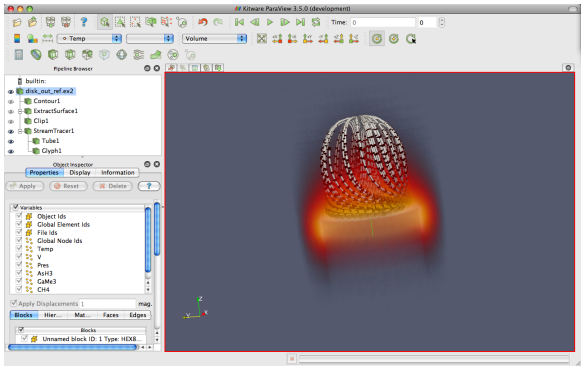
デフォルトでは、ParaView はサーフェス・レンダリングで使われているのと同じ色を、値の範囲の最低値を不透明度 0、最高値を不透明度 1 に設定して描画します。ParaView では**伝達関数** (スカラー変数を表示する際の色や不透明度に関する設定) の変更も簡単に行えます。パイプライン・ブラウザ上でボリューム・レンダリングされたオブジェクトを選択した状態で、カラーマッピングの編集  ボタンをクリックしてください。



それによって現れるダイアログは、伝達関数を編集する設定を提供します。ダイアログ上部のグラデーションになったボックスは、色で伝達関数を、黒色の線プロットで不透明度を表しています。伝達関数上の丸い点は、**制御点**を表しています。制御点とは特定のスカラー値に設定された色と不透明度のことであり、制御点間の色と不透明度は補間値が与えられます。バー上の空白部分をクリックすると、新しい制御点が作成されます。既に有る制御点上でクリックすると、その点を選択します。選択された制御点は、ボックスの中をドラッグしてスカラー値と透明度を変更でき、選択された制御点上でもう 1 度クリックすることで新しくダイアログボックスが表示されます。選択された制御点は、backspace キーもしくは delete キーを押すと削除されます。それではここで、制御点の作成と変更を行ってみてください。

カラーバー直下には、テキスト入力ウィジェットがあり、選択された制御点の Scalar Value (スカラー値) と Opacity (不透明度) の数値を指定できます。Scale の設定は、不透明度の計算における単位長を調節します。数字が大きいほど、ボリュームが透明になります。Color Space の設定では、色の補間方法を変更できます。このパラメータは制御点の色には影響しませんが、制御点間の色には大きく影響します。Use Logarithmic Scale チェックボックスによって、色のスケールを対数尺度に変換することも出来ます。

伝達関数の設定は面倒になりがちですので、Save ボタンをクリックすることで設定を保存することが出来ます。Choose Preset ボタンによって現れるダイアログでは、作成したカラー・マップや、ParaView に最初から用意されているカラー・マップを、管理および適用することができます。ここで Choose Preset ボタンを押し、ダイアログボックス中の Black-Body Radiation を選択して、OK をクリックしてください。これでボリューム・レンダリングは、より熱を表現するのに相応しいようになりました。

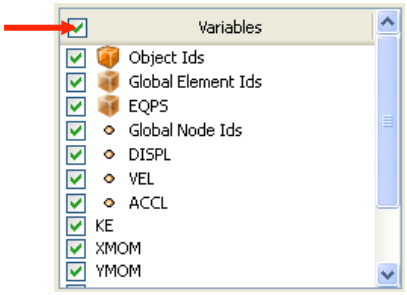


ボリューム・レンダリングのカラー・マッピングのみならず、バー・チャート・ビューによるヒストグラムを含む、全てのビューでの Temp のカラー・マッピングが変更されていることに注意してください。これによってビュー同士の一貫性を保証し、同じ変数を違う色や違う値の範囲でマッピングしてしまうことによる混乱を避けることが出来ます。

2.9 時間

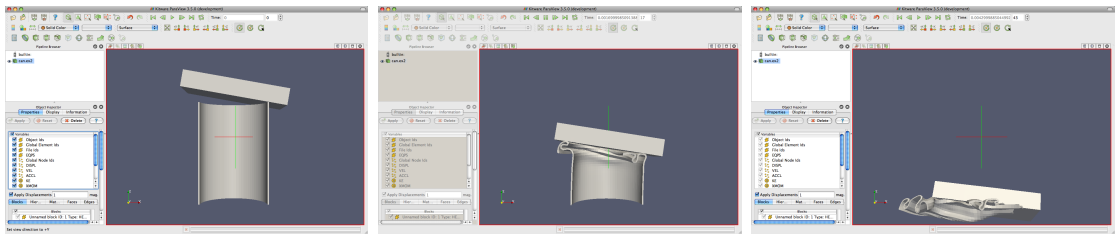
ここまで disk_out_ref のシミュレーションを余すところなく分析して来ましたので、次は新しいシミュレーションに進んで、ParaView がどのように時間を扱っているのかを見てみましょう。まず全てのデータを消去して、初期状態に戻りましょう。 ボタンを使用するのが一番簡単な方法です。このボタンについては後ほど詳しく書きますが、今のところは ParaView を再起動するのと概ね同等である、と簡単に理解していてください。

Open から can.ex2 ファイルを選択します。これは時間経過と共に変化するデータを持つ、簡単なシミュレーションです。

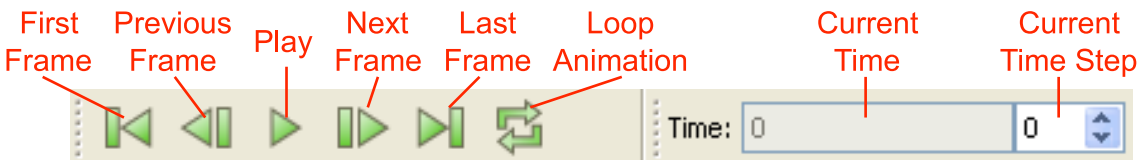


先程と同じように、変数リストの最上部のチェックボックスをクリックして、全ての変数が読み込まれるようにし、 Apply ボタンを押します。

ボタンを押してカメラをメッシュの方に向けます。ツールバー上の再生ボタン を押して、落下するレンガが缶と衝突するメッシュを ParaView がアニメーションする様子を見てみましょう。



時間変化するデータを扱う際にすることは、これだけです。ParaView には時間の概念が組み込まれていて、データ内の時間と自動的にリンクするようになっています。時間を扱うためのツールバーを理解してください。



アニメーションの保存も同様に簡単です。メニューから File → Save Animation を選択します。ParaView では、ダイアログでどのようにアニメーションを保存するかの設定ができ、そして自動的に時間を反復してアニメーションを保存します。

ユーザーが陥る最大の落とし穴は、値の範囲が時間変化する場合の色のマッピングです。例を見るために、まず時間を最初の時刻ステップに設定  し、EQPS 変数をオンにし、色の凡例  をオンにします。ここで、アニメーションを最後まで再生  して下さい (もしくは最後の段階までスキップ  します)。色づけはあまり効果を発揮していません。ただし、データ範囲にスケールを調整する  ボタンを押すことで、簡単に問題は解決できます。

これは一見、欠陥のように見えますが、そうではありません。これは 2 つの避けがたい要因のためです。1 つ目は、スカラー場の可視化を行うと、スケール範囲はその時刻ステップでのデータの値の範囲に合わせて設定されます。理想を言えば、スケールの範囲は、データの全ての時間を通しての最大値と最小値に設定されるべきです。しかしながら、それを実現しようとすると ParaView は最初の読み込みで全てのデータをチェックすることになり、大規模データを処理する際にひどく動作が遅くなってしまいます。2 つ目は、時間を追ってアニメーションさせる際には、例えデータの値の範囲が変わってもスケールの範囲は固定である必要があるからです。アニメーションの再生に合わせて表示するスケールの範囲を変更することは、データを誤って解釈する原因になります。スケールの範囲が暗黙に変動するよりは、当初設定したとおりのカラー・マッピングの範囲から逸脱する方が断然良いでしょう。とはいえ、代表的な時刻ステップで  を押すか、カラー・スケールのダイアログを開き  、データの範囲を指定することで、このような問題を簡単に解消することができます。

2.10 選択機能

ParavView 3 のリリースにおいて大いに進歩し、なお進化し続けている特徴は選択機能です。選択はいつでも行うことができ、また ParaView は現在選択されている部分を、全てのビュー間で同期された形で保持しています。つまり、あるビューで何か選択すると、その選択範囲は同じオブジェクトを表示している全てのビューで表示されます。

ParaView では、一つのデータセットにおける点、セル、ブロックを選択することができます。また、多種の ID リストアップ機能、空間中の位置指定、スカラー値の指定などの様々な方法によって、選択範囲に含まれる要素を指定することができます。これから、それらの組み合わせのいくつかを見てみましょう。

選択部分を生成する最も簡単な方法は、要素をまさに 3D ビューの中から選び出すことです。3D ビューにおける全ての選択操作は、**ラバー・バンド**という選択機能によって行います。つまり、3D ビュー内でマウスをクリックしたりドラッグすることで、箱状の範囲内の要素を選択することができます。ラバー・バンドによる選択方法はいくつかの種類があり、選択設定のツールバーにあるアイコンのいずれかを選ぶか、ショートカット・キーのいずれかによってそのうちの一つを選ぶことができます。次の 3D 選択方法が可能です。

Select Cells On (Surface) (表面上のセルを選択する) ビュー内で見えるセルを選択します。(ショートカットは s)

Selects Points On (Surface) (表面上の点を選択する) 視点内で見える点を選択します。

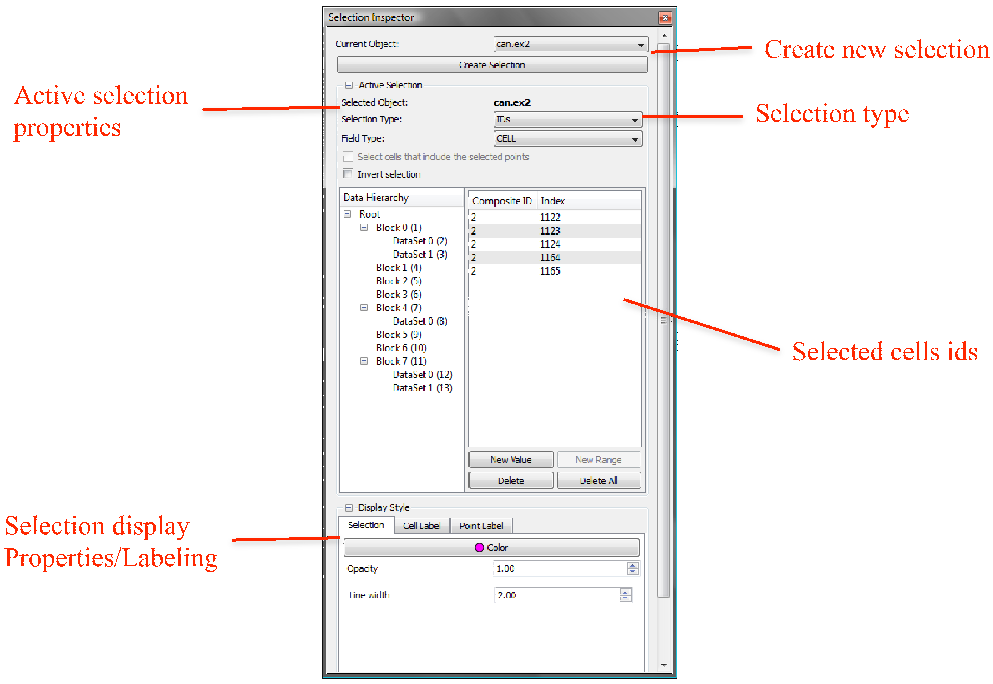
Select Cells Through (Frustum) (表面および内部のセルを選択する) ラバー・バンド内に存在する全てのセルを、対象オブジェクト内部のセルも含めて選択します。

Select Points Through (Frustum) (表面および内部の点を選択する) ラバー・バンド内に存在する全ての点を、対象オブジェクト内部の点も含めて選択します。

Select Blocks (ブロックを選択する) マルチブロック・データセットから、ブロックを選択します。(ショートカットは b)

s と b のショートカットによって、それぞれセルやブロックを手早く選択できます。マウスカーソルを選択中の 3D ビュー上のどこかに置いてから、それらのキーを入力しましょう。そして、選択したいセルやブロックをクリックしましょう (あるいは複数の要素上にラバー・バンドをドラッグしましょう)。

ここで選択機能を試してみて下さい。



選択要素は**セレクション・インスペクタ**で管理できます。セレクション・インスペクタは View → Selection Inspector で見ることができます。セレクション・インスペクタでは、選択されている全ての点やセルを見るほかにも、選択を修正すること

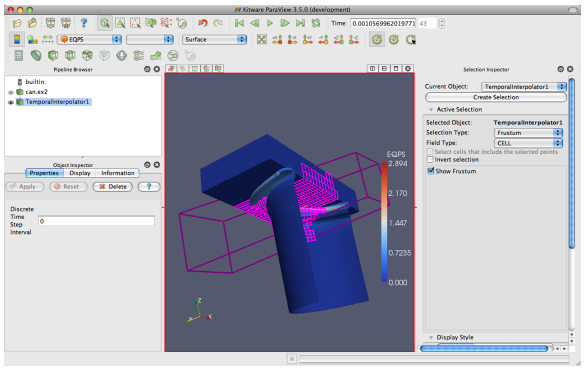
もできます。またセレクション・インスペクタを使って選択要素にラベルを付加し、要素の特定をしやすくすることができます。

セレクション・インスペクタを少し試してみましょう。Selection Inspector を開いてください。そしてラバー・バンドを使って選択範囲を作成し、Selection Inspector で結果を見てみてください。また ID を変更したり、Invert Selection のチェックボックスによって選択範囲を反転して、選択範囲を変更してみてください。

すると / の表面選択ツールは点/セルのリストを表示し、 のブロック選択ツールはブロックのリストを表示しますが、 / の内部選択ツールはどちらも表示していないことに気づくでしょう。これは、これらのツールが空間内の領域を選択しているからです。選択領域を見たい場合は、Show Frustum をクリックし、3D ビューを回転させてください。

ここで、一連の点やセルを特定する選択と、空間内の領域を特定する選択には、根本的な違いがある事を記しておくべきでしょう。この違いを理解するには、以下を試してください。

1. Select Cells Through ツールを使って、選択領域を生成してください。
2. Selection Inspector 内の Show Frustum のチェックボックスをクリックして、3D ビューを回転してください。

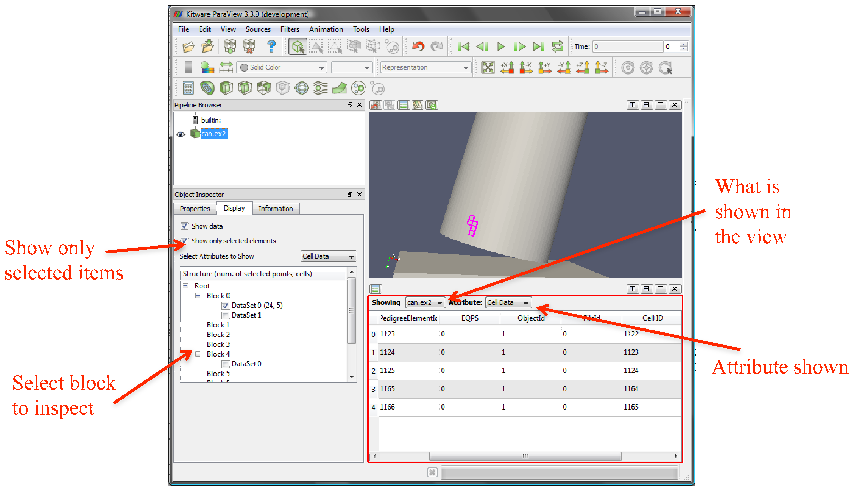


3. アニメーションを少しだけ再生 しましょう。このとき、選択領域は固定されたまま、どのセルが領域内あるいは領域外へ移動するかによって、選択要素が変化していることに注意して下さい。
4. Selection Type を IDs に変更してください。
5. もう一度再生 してください。今度は、選択されたセルが位置にかかわらず変化しないことに注意して下さい。

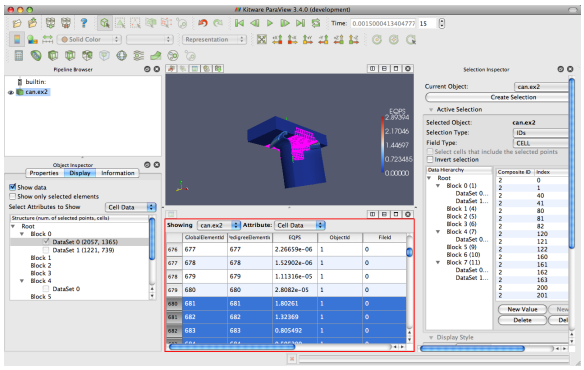
スプレッドシート・ビューは選択と組み合わせて使い、分析を掘り下げるために重要なツールです。スプレッドシート・ビューでスカラー場の実際の数値を読むことができ、また選択の仕組みも活用することで目的の数値を特定しやすくなります。

1. ビューを分割してください (あるいは)。
2. 新しいビューにて、Spreadsheet View ボタンをクリックしてください。
3. もし can.ex2 が見えていなかったら、(対応する をクリックして) 見えるようにしてください。

お分かりのとおり、スプレッドシート・ビューはかなり簡潔です。スプレッドシート・ビューの上部にある 2 つのコンボ・ボックスについて記します。1 つ目の Showing で、表示するデータセットを (パイプライン・ブラウザを使わずに) 手早く選ぶことができます。2 つ目の Attribute で、異なるタイプのフィールド・データから 1 つを選び出すことができます。一度に 1 種類のデータ (たとえばポイント・データまたはセル・データ) しか表示できません。なぜなら、それぞれの種類における行の数や列の種類は異なるからです。スプレッドシート・ビューはまた、それ自身の Display パネルを持っています。



4. Attribute のコンボ・ボックスから Cell Data を選びます。
5. スプレッドシート・ビューをスクロールして、ハイライトされた行を探します。(Display パネルにて、異なるブロックを選択しなければならないかもしれません。)

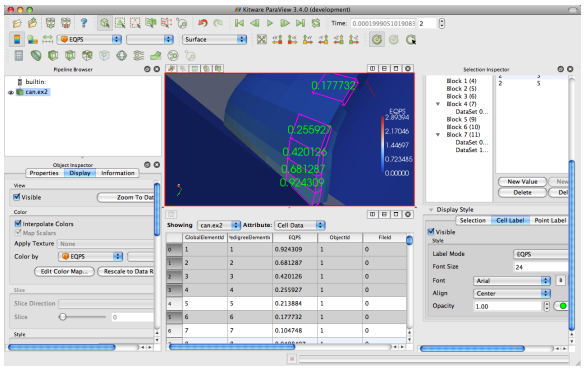


それらのハイライトされた行は、現在選択中の要素の一部です。このようなビュー間の選択の連携は、複数のビューを関連付ける重要な仕組みです。とはいえ、この例では、スプレッドシート・ビューにおいて選択された要素を識別するのは難しいかもしれません。時には、選択されたデータだけを見たいこともあります。

6. Display パネルにて、Show only selected elements をオンにしてください。

3D ビューにおいて作成された選択に含まれる要素のみが、スプレッドシート・ビューに現れるのが見て取れました。この関連付けは、逆向きにも働きます。すなわち、スプレッドシート・ビューで選択を作成すると、それらは 3D ビューにも表示されます。加えて、Selection Inspector を使って 3D ビュー内の選択要素にラベルを付加することもできます。

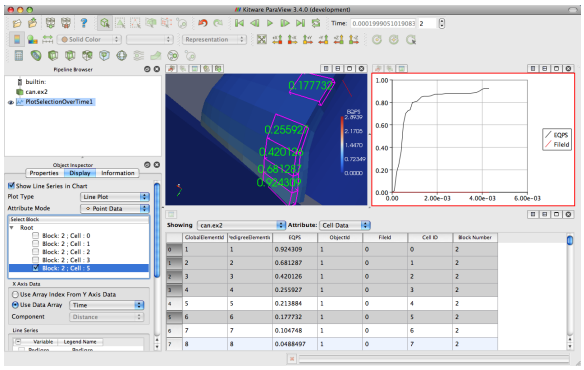
1. Show only selected elements のチェックを外します。
2. スプレッドシート・ビューで２〜３行を選択します。
3. 3D ビューで選択された結果を探します。
4. Selection Inspector 内の (下部にある) Cell Label タブをクリックします。
5. Visible のチェックを入れます。
6. Label Mode を EQPS に変更します。



ParaView では、フィールド・データを時間に沿ってプロットできます。全部のデータを全ての時刻ステップでプロットしたいことは滅多にないでしょうから、これらのプロットは選択部分に対して行われるようになっています。

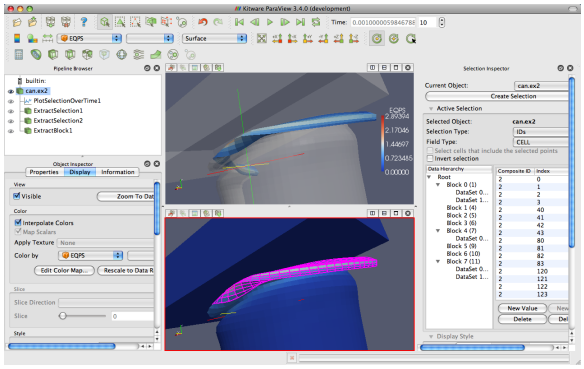
1. 選択が有効な状態で、Plot Selection Over Time フィルタを追加してください (Filters → Data Analysis → Plot Selection Over Time )。
2. Object Inspector 内の Copy Active Selection をクリックします。
3.  をクリックします。

4. Display パネルで、プロットする別のブロック (選択した要素のそれぞれと一致します) を選択してください。



また、選択された点やセルを個別に見たり、それらについてのみ何らかの処理を行うために、選択部分を抽出することもできます。これは Extract Selection フィルタによって行われます。このフィルタを実行してみてください。

1. セルのラベル表示をオフにします。
2. 内部も含めたセルの選択 で、多めのセルを選択します。
3. Extract Selection フィルタを適用します (Filters → Data Analysis → Extract Selection)。
4. オブジェクト・インスペクタ内の Copy Active Selection ボタンをクリックします。
5. Apply ボタンを押します。

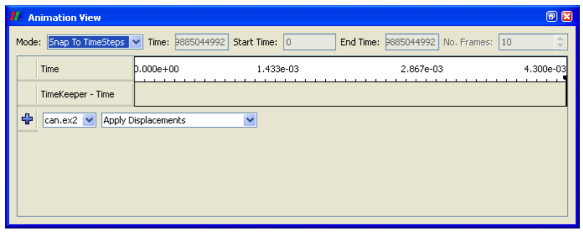


ビュー内のオブジェクトは、先ほど選択したセルに置き換えられます。(この画像では、抽出したセルを全体のデータと関連させて見せるために、半透明な表面と、もとの選択部分を示すもう一つのビューを加えています。) フィルタを選択部分を抽出したパイプライン・オブジェクトに単に付加するだけで、抽出されたセルに対しての演算を行うことができます。

抽出されたセルに対する操作は一通り終わったので、これらを消してしまいましょう (つまり、ExtractSelection1 フィルタを Delete します)。Selection Inspector を閉じて、プロットとスプレッドシート・ビューも削除して構いません。これでおしまいです。また PlotSelectionOverTime1 フィルタも Delete して構いません。

2.11 時間の制御

ParaView には、時間とアニメーションを制御する強力な設定が多数あります。これらの大多数は、**アニメーション・ビュー**を通じて利用します。メニューから、View → Animation View をクリックします。



まず、アニメーション・ビューの上部にあるウイジェットを試してみましょう。アニメーションの**モード**設定は、ParaView がどのように再生中に時間を進めるかを定義します。3 つのモードを利用できます。

Sequence (シーケンス) 開始、終了時刻を与えれば、アニメーションを決められたフレーム数に等間隔に分割します。

Real Time (リアル・タイム) ParaView は決められた秒数の再生時間となるよう、アニメーションを再生します。実際のフレーム数は、フレーム間の更新時間に依存します。

Snap To TimeSteps (時刻ステップにスナップする) ParaView はデータで定義された時刻ステップどおりに再生します。

時間を含んだファイルを読込むときは常に、ParaView は自動的にアニメーション・モードを Snap To TimeSteps に変更します。そのためデフォルトでは、データを読込んで、再生 ▶ を押せば、データで定義されたとおりの各時刻ステップを見ることができます。これが最も一般的な使い方です。

それ以外の使い方は、シミュレーションの書出したデータの時刻ステップ間隔が、可変であるときに発生します。このような場合にはおそらく、時間のインデックスよりも、シミュレーション上の時間に従ってアニメーションを再生したいでしょう。問題ありません。残り 2 つのアニメーション・モードのうちの 1 つに切り替えられます。別の使い方は、再生速度を変えたい場合です。アニメーションの速度を速く

したり、遅くしたいことがあるでしょう。残り 2 つのアニメーション・モードでは、それができます。

やってみましょう。アニメーション・モードを、Real Time に変更しましょう。初期状態では、アニメーションはデータで指定されたとおりの時間の範囲となり、再生時間は 10 秒間になっています。アニメーションを再生  しましょう。結果は似ていますが、アニメーションは、シミュレーション上の時間に比例して進行し、10 秒間で完了します。

さて、Duration を 60 秒に変更してみましょう。アニメーションは明らかに、ゆっくりと再生されます。あなたのコンピュータでの画面更新が遅くなければ、アニメーションが前に比べて断続的になっていることにも気づくでしょう。これはデータセットの時間解像度を上回ったからです。これは多くの場合、望ましい挙動と言えます。データの中でまさにどれが表示されているかが分かるからです。しかし、プレゼンテーション用にアニメーションを作りたいのなら、より滑らかなアニメーションが求められるでしょう。

ParaView には、これを可能にする特別なフィルタが存在します。それは**テンポラル・インターポレータ**と呼ばれています。このフィルタは、メッシュ座標等の位置的データおよびフィールド・データに対して、元のデータセットで定義された時刻ステップの間を補間します。この機能は、最近の ParaView と VTK パイプライン構造の進歩によって可能になりました。このフィルタを試してみましょう。パイプライン・ブラウザの can.ex2 を選択してハイライト表示し、Filters → Temporal → Temporal Interpolator を選択しましょう。 

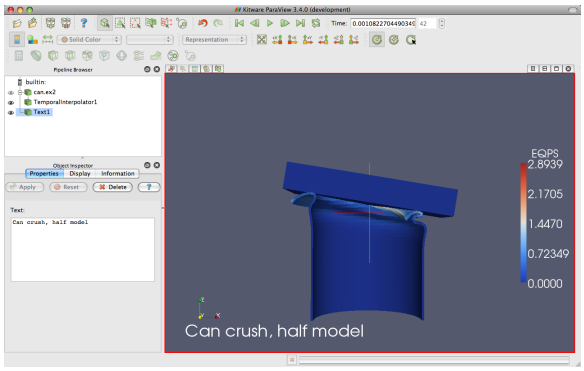
を押して、必要ならばアニメーション・ビューをリアル・タイム・モードに戻しましょう。さらにビューを分割  し、一方に TemporalInterpolator1 を、他方に can.ex2 を表示し、カメラをリンクしましょう。アニメーションを再生  し、効果を見ましょう。

2.12 テキストによる注釈

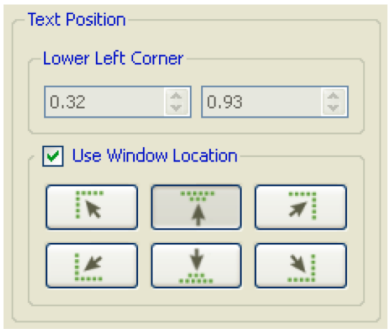
ParaView をコミュニケーションのための道具として用いるとき、作成した画像にテキストの注釈をつけることは時に役に立ちます。ParaView 3 では、3D ビューにいつでも好きな時に注釈をつけることが非常に簡単になりました。ビュー内に単にテキストを配置する特別な**テキスト・ソース**があります。やってみましょう。

1. メニューバーから、Sources → Text を選択します。
2. オブジェクト・インスペクタのテキスト入力欄に、適当な文を入力します。
3. 

ボタンを押します。

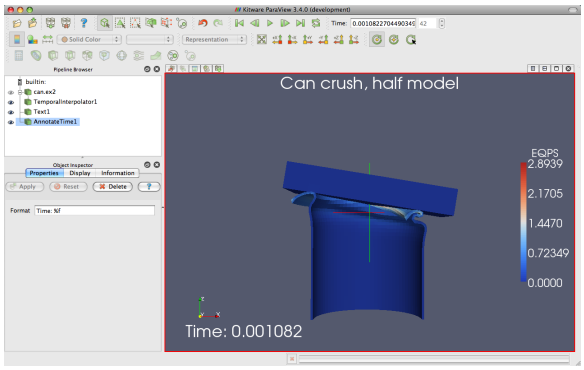


入力したテキストは、3D ビュー内に現れます。マウスでドラッグするだけで、好きなところにこのテキストを配置できます。オブジェクト・インスペクタの Display タブでは、テキストのサイズ、フォント、色の追加設定ができます。また、テキストを最もよく使う位置に配置するためのボタンもあります。



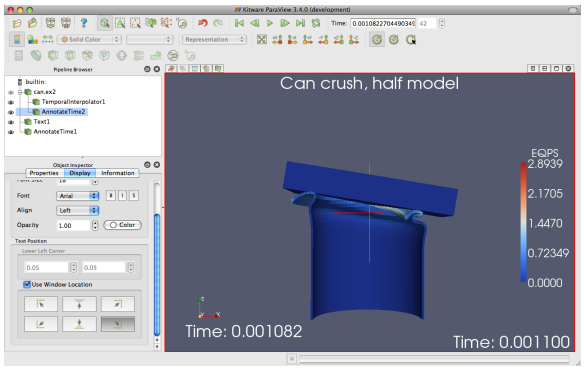
しばしば現在の時刻の値を、テキスト注釈に入れる必要があるでしょう。標準のテキスト・ソースで現在の時間の値を入力するのは退屈で、間違いを起こしやすく、さらにアニメーションを作る時は、これが不可能です。そこで、現在のアニメーションの時刻を文字列に挿入する特別な**アノーテート・タイム**・ソースがあります。

1. Animate Time ソース (Sources → Annotate Time) を追加します。
2. 必要ならば注釈の位置を変更します。
3. 再生  し、時刻の注釈がどう変化するか観察します。



現在のアニメーションの時刻が、データ・ファイルから読み込まれた時刻ステップと同じでない場合があります。時には、データ・ファイルに保存されている時刻がいつであるかを知ることが重要なことがありますが、ここにフィルタとして機能するアノーテート・タイムの特別版があります[3]。

1. can.ex2 を選びます。
2. Filters → Alphabetical → Annotate Time を選択します。
3. Apply をクリックします。
4. 必要ならば注釈の位置を変更します。
5. 再生 し、時刻の注釈がどう変化するか観察します。



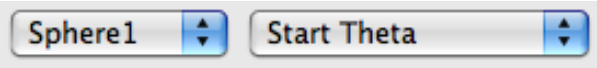
2.13 アニメーション

私たちは既に、時間を含むデータ・セットをアニメーションさせる方法を見てきました (を押します)。しかし、ParaView のアニメーション能力はそれ以上のものです。ParaView では、全てのパイプライン・オブジェクトのほぼ全てのプロパティをアニメーションさせることができます。それを今から実行しますが、まずは現在の ParaView の状態を消去するために ボタンを押してください。これで単純なアニメーションを作る準備ができました。

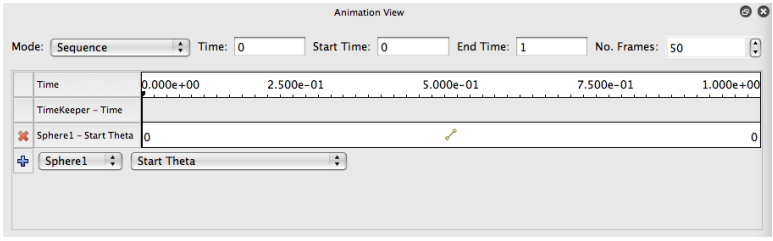
1. スフィア・ソース (Sources → Sphere) を生成し、Apply をクリックします。
2. そして、アニメーション・ビュー・パネルが可視状態である (もし可視状態でなければ、View → Animation View) ことを確認します。
3. No. Frames の設定を 50 に変更します (10 では速すぎるでしょう)。

[3]訳注: 以下の再生時には、リアル・タイム・モードとなっていることを確認して下さい。

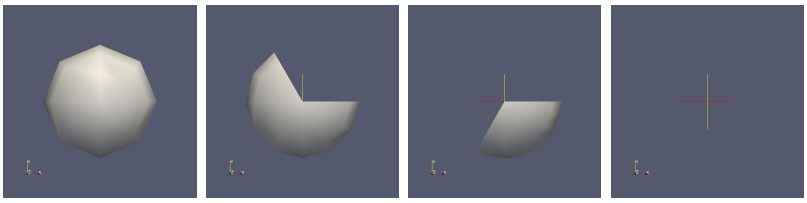
4. アニメーション・ビュー下部のプロパティ選択ウィジェットで、1つ目のボックスでSphere1を、2つ目のボックスでStart Thetaを選択します。



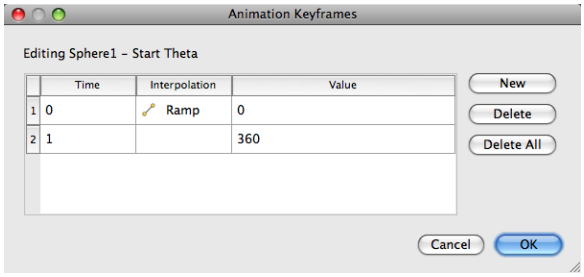
ボタンを押します。



ここで行ったことは、Sphere1オブジェクトのStart Thetaプロパティのための**トラック**を作成する操作です。トラックは、アニメーション・ビューでは水平なバーで表されます。トラックには、ある特定の瞬間におけるプロパティの値を指定する**キー・フレーム**があります。キー・フレーム間では、プロパティの値は補間されます。トラックを生成すると、2つのキー・フレームが自動的に生成されます。1つは開始時刻の最小値をもつキー・フレームで、もう1つは終了時刻の最大値をもつキー・フレームです。ここで設定したStart Thetaプロパティは、球形状の経度方向の開始位置を定義します。このアニメーションを再生 させると、球が展開して、ついには開ききって消滅するのが見られるでしょう。

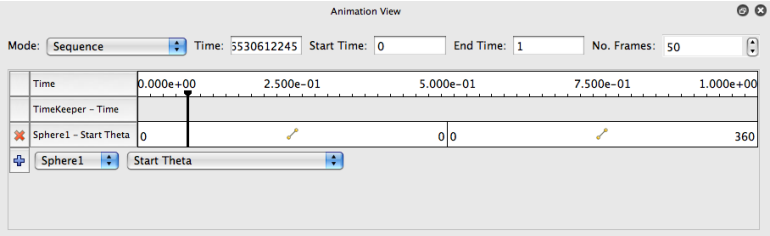


トラックはその上でダブルクリックすることで、修正できます。ダブルクリックすると、キー・フレームを追加、削除および修正するのに使えるダイアログ・ボックスが表示されます。



この機能を、アニメーションに新たなキー・フレームを追加するのに使います。

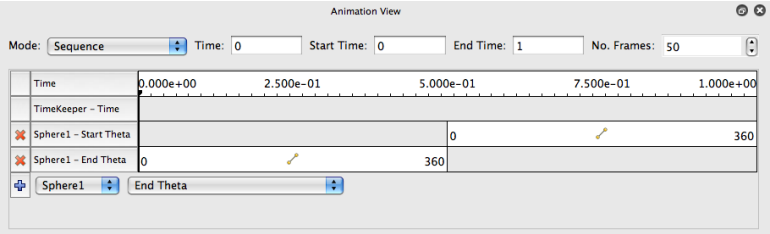
5. Sphere1 – Start Theta トラックをダブルクリックします。
6. Animation Keyframes ダイアログ内の New ボタンをクリックします。これで時刻 0.5 のところに新たなキー・フレームが作成されます。
7. 1 番目のキー・フレームの値を 360 に、2 番目のキー・フレームの値を 0 に修正します。
8. OK をクリックします。



アニメーションを再生すると、球ははじめ大きくなり、その後また小さくなります。

アニメーションさせられるのは 1 つのプロパティに限定されません。いくつのプロパティでも、必要なだけアニメーションできます。ここでは、2 つのプロパティを変更するアニメーションを作ってみましょう。

1. Sphere1 – Start Theta トラックをダブルクリックします。
2. Animation Keyframes ダイアログ内で、(時刻ステップ 0 の) 最初のトラックを Delete します。
3. アニメーション・ビューで、Sphere1 オブジェクトの End Theta プロパティのトラックを作成します。
4. OK をクリックします。
5. Sphere1 – End Theta のトラックをダブルクリックします。
6. 2 番目のキー・フレームの時刻を 0.5 に変更します。

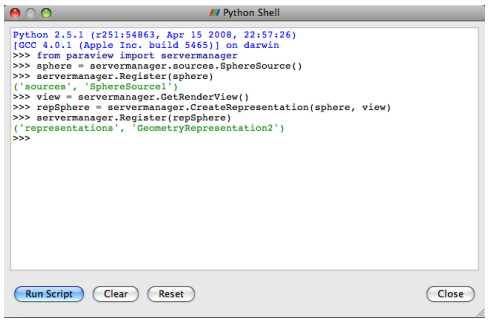


このアニメーションでは、球が生成・消滅するのが見られますが、今回は球形状の描画範囲の変化部分が同じ方向に回転しています。それによって、このアニメーションをループ  させたときに、とても満足のいくアニメーションとなっています。

2.14 スクリプティング

ParaView の設定を変更し、自動化する方法は多数あります。そのための最も便利な方法の一つは、ParaView に内蔵された Python スクリプティングの機能を利用することです。Python バインディングはこのチュートリアルの範囲を超えますが、それを利用するための方法を述べておきます。Python バインディングに関するさらなる情報は、ParaView Wiki (http://www.paraview.org/Wiki/images/2/26/Servermanager.pdf) から得ることができます。

ParaView から Python を使用する最も簡単な方法は、内蔵の Python シェルを起動する方法です。メニューから、Tools → Python Shell を選択して下さい。これによって Python シェルのダイアログ・ボックスが現れ、前もって用意したスクリプトを実行したり、保存されたステートを読込んだり、パイプライン・オブジェクトを操作したり、プラグインを読込んだり、といった様々なコマンドを実行することができます。

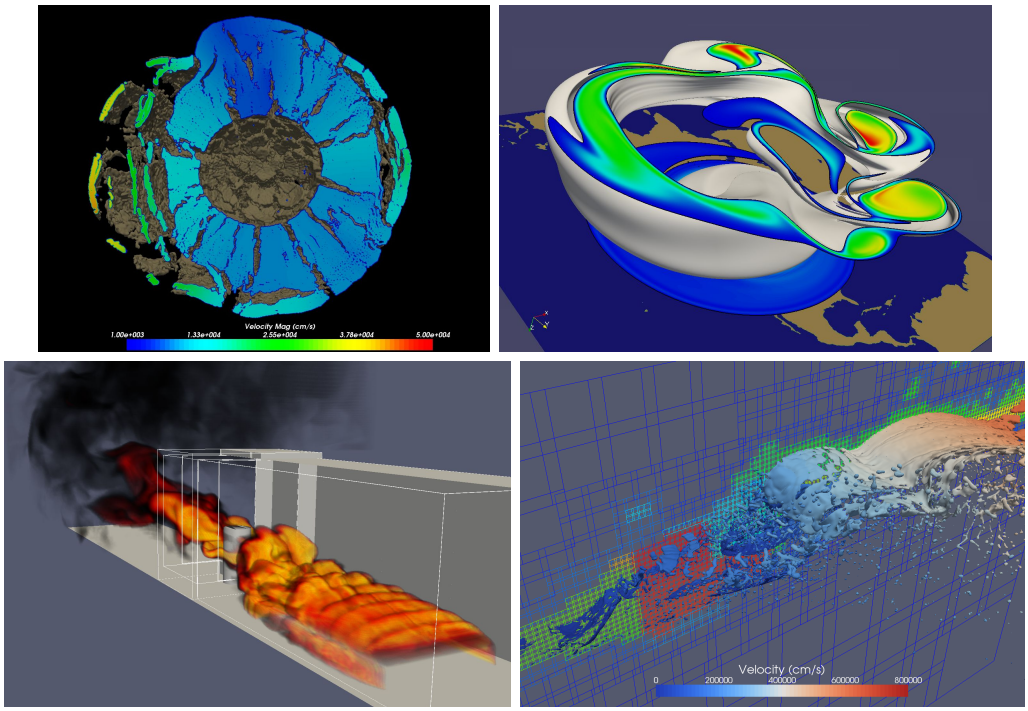


パイプライン内からデータを操作するために、Python を使用する方法もあります。**プログラマブル・フィルタ** (Filters → Data Analysis → Programmable Filter によって選択) と呼ばれる、特別なフィルタがあります。このフィルタによって、オブジェクト・インスペクタから Python スクリプトを入力することができます。このスクリプトは、パイプラインがアップデートされる度に実行されます。スクリプトからはデータに直接アクセスすることが可能で、そのデータをどのようにでも操作することができます。プログラマブル・フィルタの真に素晴らしい点は、並列モードでも動作することです。データが分散並列マシン上に存在する場合には、Python スクリプトもまたそれらのマシン上に分散され、シリアルに実行されているのと同じようにそれらのデータに対して実行されます。したがって、ユーザの側で特段の配慮をすることなく、それらのデータに対する並列スクリプティングを行うことができます。

ときには、ParaView の GUI を全く使用せずに (したがって、ユーザが介入することなく) Python スクリプトによってポストプロセッシングおよび可視化を自動化することが便利なことがあります。これは、ParaView に付属する `pvpython` アプリケーションによって可能です。`pvpython` アプリケーションは単に、ParaView バイ

ンディングを既に読込んだ状態の Python インタプリタです。このプログラムをスクリプトと共に実行して、ParaView を完全に自動化することができます。ParaView には、pvbatch と呼ばれる、似たようなプログラムも付属します。pvpython と異なり、pvbatch はクライアント／サーバ接続を確立すること無く並列に実行することが可能ですが、GUI ライブラリの幾つかは利用できません。

第3章　大規模モデルの可視化



サンディア国立研究所では、ここに示した例のような Red Storm スーパーコンピュータで実行された大規模シミュレーションのデータを可視化するために、ParaView を頻繁に使用しています。左上の画像は、ゴレブカ小惑星の中心で起爆した 10 メガトン爆発の、10 億セル以上の規模での CTH による衝撃波シミュレーションです。右上の画像は、高緯度で極大気を捕捉する極周囲のジェット気流である極渦の崩壊の、10 億セル以上の規模での SEAM 気候モデルシミュレーションです。左下は、横風火災における物体の、1000 万の非構造 6 面体格子による SIERRA/Fuego/Syrinx/Calore を用いた弱連成シミュレーションです。右下は、AMR データを生成する CTH によるシミュレーションです。ParaView は、数十億セル、数十万ブロック、そして 11 レベルの階層構造からなる CTH のシミュレーションによる AMR データの可視化に使用されてきました (ここでは示しませんが)。

本節では、ParaView の並列可視化機能を用いて、このような大規模メッシュを可視化する方法を解説します。本節は前節ほど実践的ではありません。その代わり、大規模な並列可視化を行うための概念的な知識を学んで頂くことになります。ここでは基本的な ParaView の構造および並列アルゴリズムを示し、これらの知識をどのように利用するかを実演します。

3.1 ParaView の構造

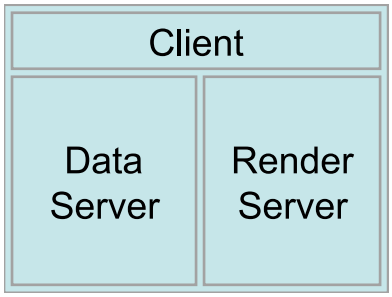
ParaView は 3 層のクライアント・サーバ構造になっています。ParaView におけるそれら 3 つの論理的なユニットは、以下のとおりです。

データ・サーバ データの読込み、フィルタリング、書出しを担当するユニットです。パイプライン・ブラウザに表示される全てのパイプライン・オブジェクトは、データ・サーバが保持しています。データ・サーバは並列実行が可能です。

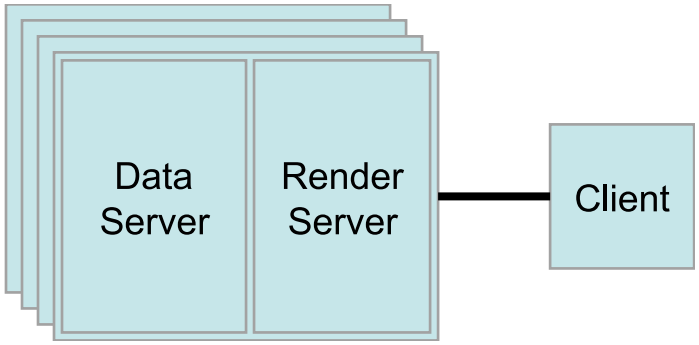
レンダー・サーバ レンダリングを担当するユニットです。レンダー・サーバも並列実行が可能で、その場合は内蔵の並列レンダリング機能が有効化されます。

クライアント 可視化の作成を担当するユニットです。クライアントはサーバにおけるオブジェクトの作成、実行、削除を制御しますが、実際のデータは全く保持しません (そのため、クライアントがボトルネックとなることが無く、サーバが大規模データへとスケーリングすることができます)。GUI もまた、クライアントに含まれます。クライアントは常にシリアルに実行される、非並列化アプリケーションです。

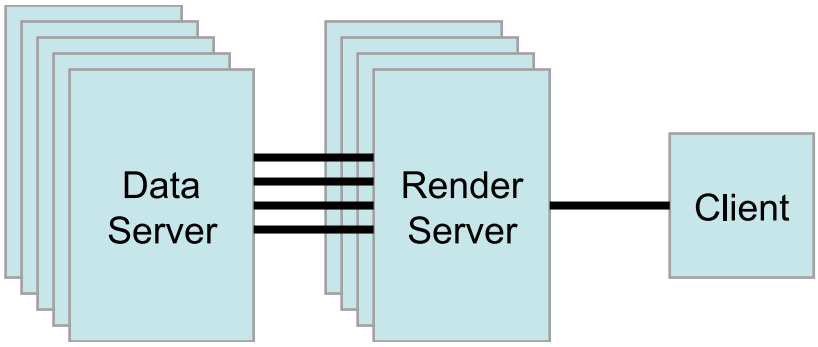
これらの論理ユニットは、物理的に分割されている必要はありません。論理ユニットは多くの場合、同一のアプリケーションに含まれ、それらのユニット間での通信を不要としています。ParaView は、以下の 3 つのモードで実行されます。



最初のモードは**スタンドアローン**・モードで、既に使用してきたとおりのモードです。スタンドアローン・モードでは、クライアント、データ・サーバ、レンダー・サーバは全て一つのシリアル実行アプリケーションに統合されています。`paraview` アプリケーションを実行すると、ParaView の全ての機能をすぐに使えるよう、**ビルトイン**・サーバに接続されています。



2 つ目のモードは**クライアント－サーバ・**モードです。クライアント－サーバ・モードでは、pvserver プログラムを並列マシン上で実行し、それに paraview クライアント・アプリケーションを接続します。pvserver プログラムはデータ・サーバとレンダー・サーバの両方を含んでおり、したがってデータ処理とレンダリングの両方が pvserver で実行されます。クライアントとサーバはソケットを通じて接続します。ソケット通信は比較的低速な通信手法であるため、このソケットを通じたデータ転送量は最小に抑えられています。



3 つ目のモードは、**クライアント－レンダー・サーバ－データ・サーバ・**モードです。このモードでは、全ての 3 つの論理ユニットが個別のプログラムとして実行されます。クライアント－サーバ・モードと同様、クライアントはレンダー・サーバに 1 つのソケットによって接続されます。レンダー・サーバとデータ・サーバはレンダー・サーバのプロセスそれぞれにつき 1 つづつ、多数のソケットによって接続されます。ソケットを通じたデータ転送量は最小に抑えられています。

クライアント－レンダー・サーバ－データ・サーバ・モードはサポートされてはいますが、このモードはまず推奨されません。このモードの元々の意図は、大規模でパワフルな計算プラットフォームと、小規模なグラフィックス・ハードウエアが内蔵された並列マシンからなる異種混在環境を活用することでした。しかしながら実際には、データ・サーバからレンダー・サーバへの転送所要時間によって、あらゆる利点が相殺されてしまうことが判りました。もし計算プラットフォームのほうがグラフィックス・クラスタより非常に大きい場合は、ソフトウエア・レンダリングを使用して下さい。もし両プラットフォームが概ね同じ規模である場合は、グラフィックス・クラスタで全ての処理を実行して下さい。

3.2 ParaView サーバのセットアップ

スタンドアローンの ParaView をセットアップするのは通常、難しくありません。コンパイル済みのバイナリをダウンロードし、コンピュータにインストールすれば、すぐに実行可能です。しかしながら、ParaView サーバのセットアップはどうしても、それよりは難しくなります。まず、サーバを自前でコンパイルしなければなりません。並列プログラミングのためのライブラリである MPI には様々な種類のものがあ

り、さらに個々の MPI は並列コンピュータの通信ハードウエアに合わせて変更が可能であるため、全ての考えうる組合せについて信頼できるバイナリファイルを提供するのは不可能です。

ParaView を並列マシンでコンパイルするには、以下が必要です。

- CMake クロスプラットフォーム・ビルディング・ツール (www.cmake.org)
- MPI
- OpenGL (または、他に利用可能なものが無ければ Mesa 3D www.mesa3d.org)
- Qt 4.3 (オプション)
- Python (オプション)

オプションのライブラリ無しでコンパイルした場合は、それに対応する機能を利用できなくなります。Qt 無しでコンパイルした場合は、GUI アプリケーションを利用できません。Python 無しでコンパイルした場合は、スクリプト機能を利用できません。

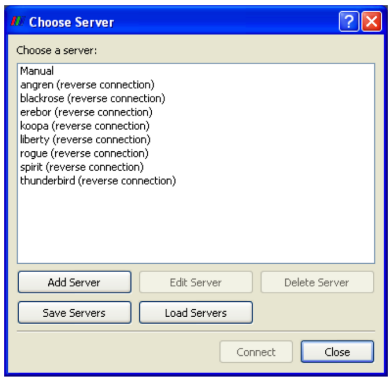
ParaView をコンパイルするには、まず CMake を実行し、コンパイルのための設定とシステム上のライブラリの指定を行います。これによって Makefile が生成され、その Makefile を使用して ParaView をビルドします。ParaView サーバのビルドに関するさらなる詳細については、ParaView Wiki をご覧下さい。

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

ParaView を並列に実行するのもまた、必然的にスタンドアローンのクライアントを実行するより難しくなります。リモート・コンピュータにログインし、並列ノードを確保し、並列プログラムを起動し、接続を確立し、ファイアウオールのトンネリングを行うといった、実行環境に依存する幾つもの段階を経る必要があります。

クライアントーサーバ間接続は、`paraview` クライアント・アプリケーションによって確立されます。サーバに接続し、またサーバへの接続を解除するには、  および  ボタンを使用します。ParaView の起動時には、ビルトイン・サーバという特別なサーバに自動的に接続されます。また、サーバへの接続が解除された時にも、常にビルトイン・サーバに接続されます。それら両者の例は、既に見たとおりです。

 ボタンをクリックすると、接続可能な既知のサーバのリストを含んだダイアログが現れます。このサーバのリストはサイト、またはユーザごとに設定できます。

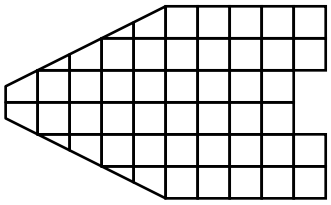


どのようにしてサーバに接続するかは、Add Server ボタンをクリックして GUI で設定するか、または XML の設定ファイルによって設定することができます。サーバ接続を指定するための方法は幾つかありますが、最終的にはサーバを起動するためのコマンドと、サーバの起動後に接続するためのホスト名を設定することになります。サーバ接続の確立に関するさらなる詳細については、ParaView Wiki をご覧下さい。

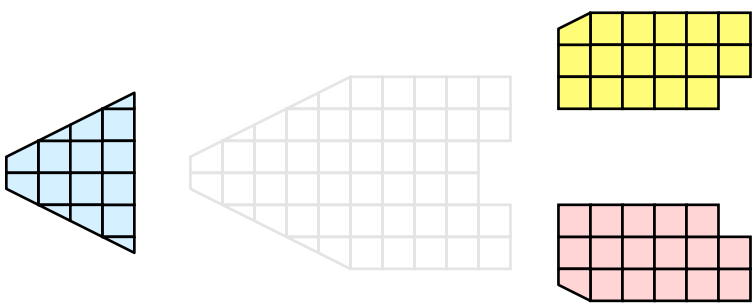
http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server#Running_the_Server

3.3 並列可視化アルゴリズム

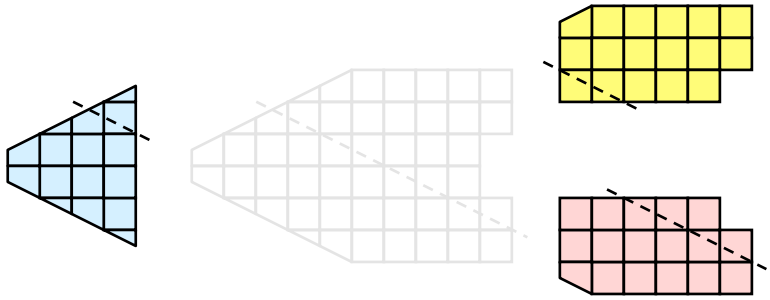
ひとたび並列化のフレームワークさえ作成すれば、並列可視化を行うのは単純である点で、私たちは幸運といえます。データはメッシュに含まれていますから、それはすなわち、既にデータがセルによって細かな断片に細分化されていることになります。それらのセルをプロセスごとに分割することで、分散並列マシン上で可視化を行うことができます。具体的に示すため、以下の非常に簡略化されたメッシュを考えます。



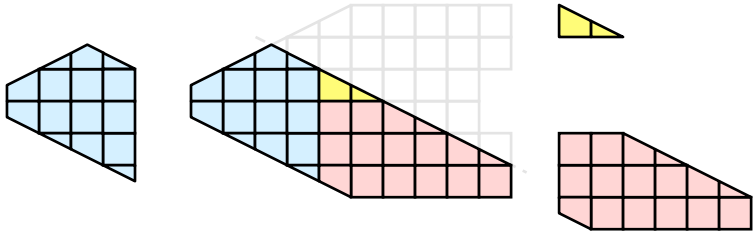
このメッシュについて、3 つのプロセスを使用して可視化を行いたいとします。以下に青、黄、ピンク色の領域で示すように、このメッシュのセルを分割します。



ひとたび分割されれば、幾つかの可視化アルゴリズムは、それぞれのプロセスにおいてローカルに保持されるセルに対してアルゴリズムを独立に実行することで、処理することができます。クリッピングを例にしましょう。クリッピングを行う切断面を定義し、この切断面をそれぞれのプロセスに与えます。

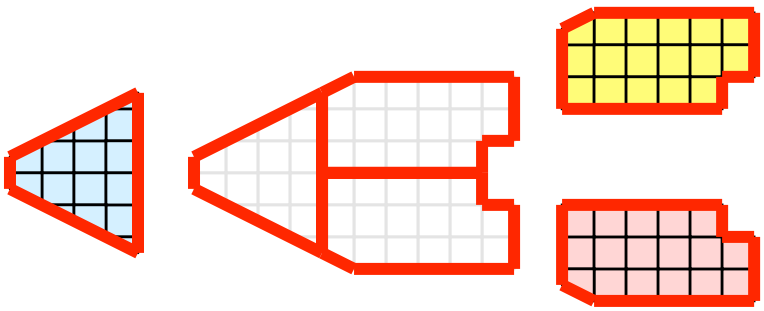


それぞれのプロセスは、この切断面によって独立にクリッピングを行うことができます。最終的に得られる結果は、このクリッピングをシリアルに実行した場合と同じです。(大規模データでは明らかに、絶対に行うべきではありませんが) これらのセルをもし、再び結合したとすれば、クリッピング演算が正しく行われたことがわかります。



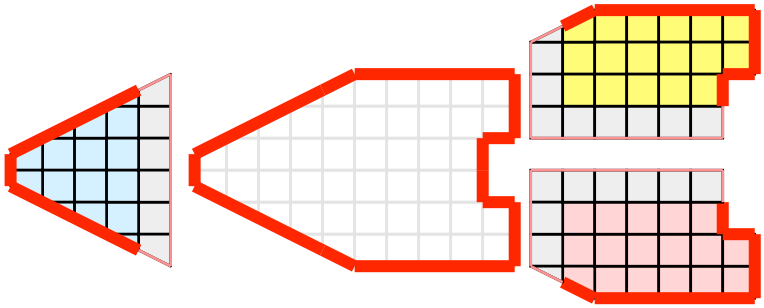
3.4 ゴースト・レベル

しかしながら残念なことに、可視化アルゴリズムを分割されたセルに対して闇雲に実行しても、必ず答えが正しいとはとは限りません。簡単な例として、**外部界面**アルゴリズムを考えて下さい。外部界面アルゴリズムとは、1 つのセルにのみ属するセル界面のすべてを探索する、すなわちメッシュの境界面を特定するアルゴリズムです。



おっと。全てのプロセスが外部界面アルゴリズムを個別に実行すると、多くの内部界面が外部界面として誤って特定されるのが判ります。これは、あるパーティションのセルが別のパーティションのセルに隣接するところで起こります。プロセスが別のパーティションのセルにアクセスする方法がありませんので、これらの隣接セルが存在することを知る方法はありません。

ParaView や、その他の並列可視化システムで採用されている解決法は、**ゴースト・セル**の利用です。ゴースト・セルとは、あるプロセスによって保持されていますが、実際には別のプロセスに属するセルのことです。ゴースト・セルを使用するには、それぞれのパーティションにおける全ての隣接セルを特定しなければなりません。そして、それらの隣接セルをそのパーティションにコピーし、コピーされたセルをゴースト・セルとして、以下の例では灰色のセルで示されるようにマークします。



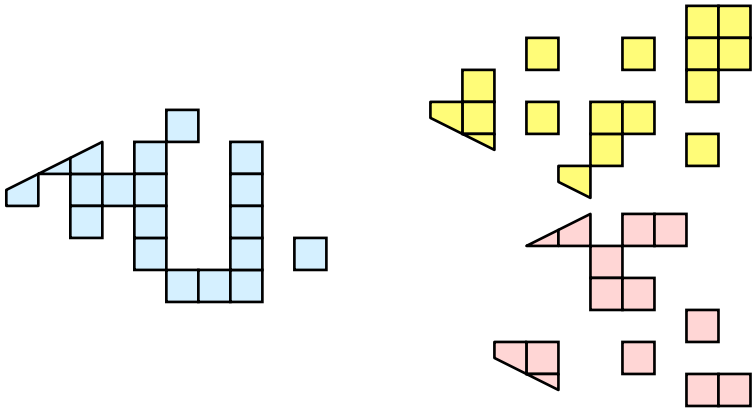
外部界面アルゴリズムをゴースト・セルに対して実行すると、それでもなお幾つかの内部界面を外部として誤って特定しているのが判ります。しかしながら、これらの全ての誤って特定された界面はゴースト・セルに属しており、したがってそれらの界面は属するセルの「ゴーストである」との状態を継承しています。それらの「ゴーストな」界面は ParaView によって取り除かれ、正しい解に辿り着きます。

この例では 1 層のゴースト・セル、すなわちパーティション内のセルに直接隣接するセルのみを示しました。ParaView では、複数の層からなるゴースト・セルを取出すことが可能です。すなわち、それぞれの層が、下のゴースト層あるいは元のデータそのものに含まれていない一つ下の層の隣接セルを含んでいます。これは、それぞれがゴースト・セルの層を必要とするようなフィルタを直列に使用する際に便利です。それらのフィルタは、それぞれ上流側に対して追加のゴースト・セルの層を要求し、下流側へデータを送る際に 1 層だけ取り除きます。

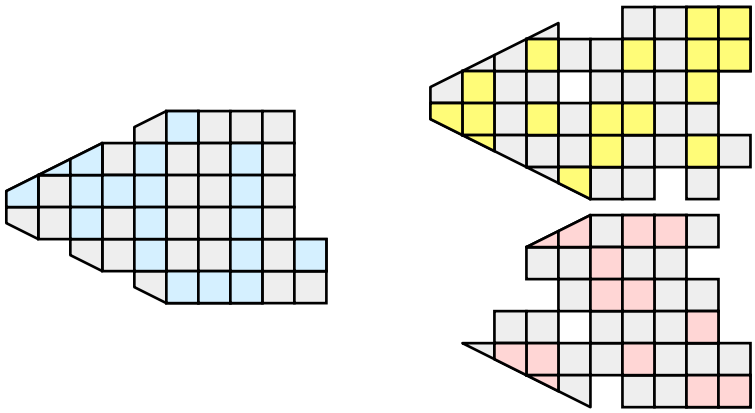
3.5 データの分割

ここではデータを分割して分散させるため、データの分割法の影響に関して議論しておくこととします。前述の例で示されたデータは**空間的にコヒーレントな**分割法と言えます。すなわち、各パーティションのセルは全て空間中のコンパクトな領

域に存在しています。その他にもデータを分割する方法は存在します。例えば、ランダムに分割する方法も有り得ます。



ランダム分割には、良い面があります。というのも、パーティションの作成および負荷分散が容易であるからです。しかしながら、ゴースト・セルに関しては、深刻な問題が存在します。



この例では、1 層のゴースト・セルによって、全プロセスにおいてデータセットのほぼ全体が複製されてしまっていることが判ります。したがって、並列処理の利点がほぼ失われてしまっています。ParaView においてゴースト・セルは頻繁に使用されるため、ParaView ではランダム分割は使用されていません。

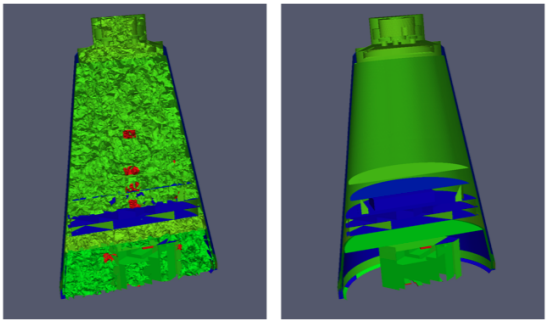
3.6 D3 フィルタ

前節では、並列可視化のための負荷分散およびゴースト・セルの重要性を述べました。本節では、負荷分散のための方法を述べます。

負荷分散およびゴースト・セルは、構造データ (画像データ、直線格子、および構造格子) を読み込んだ際には、ParaView によって自動的に取扱われます。構造データが暗黙に有するトポロジのために、データを空間的にコヒーレントなパーティションに分割し、隣接セルの位置を特定するのが容易であるためです。

ところが、非構造格子 (ポリ・データおよび非構造格子) を読み込む際には、全く異なった状況になります。非構造格子には、暗黙のトポロジや隣接セルの情報がありません。このときの ParaView の動作は、どのようにデータがディスクに書かれたかに依存します。したがって、非構造格子が読み込まれる際には、そのデータがどの程度良好に負荷分散されるかについては保証されません。さらに、そのようなデータがゴースト・セルの情報を含んでいる可能性は低いため、幾つかのフィルタの出力は正しくない可能性があります。

幸い ParaView には、非構造格子の負荷分散を行って、ゴースト・セルを作成するためのフィルタがあります。このフィルタは、distributed data decomposition (分散データ分割) の頭文字をとって D3 と呼ばれます。D3 フィルタの使用法は簡単で、(Filters → Alphabetical → D3 に存在する) このフィルタを、分割し直したいデータに適用するだけです。



D3 の最も一般的な使い方は、非構造格子データ読み込みモジュールに対して直接適用する方法です。読み込まれたデータが如何にうまく負荷分散されていても、以降のフィルタが正しいデータを生成するよう、ゴースト・セルを抽出することは重要です。上記の例は、ある非構造格子へ表面抽出フィルタを適用した結果の断面です。左図ではゴースト・セルが無いために、多くの界面が誤って抽出されていることが判ります。一方で右図では、D3 フィルタをまず最初に適用することで、その不具合が修正されています。

3.7 ジョブ・サイズにデータ・サイズを合わせる

ParaView **サーバとして、幾つのプロセスを使用すべきでしょうか。** これは多くの重要な影響を含む、普遍的な質問と言えます。そしてまた、非常に難しい質問でもあります。それぞれのプロセスにどのようなハードウエアが対応付けられているか、どの程度の大きさのデータを処理しようとしているか、どのような型のデータを処理しようとしているか、どのような種類の可視化操作を行おうとしているか、そしてユーザ自身の我慢強さなどの要因によって、その回答は大きく左右されます。

したがって、確固たる回答はありません。しかしながら、幾つかの経験則はあります。

構造データ (画像データ、直線格子、構造格子) に対しては、少なくとも 2,000 万セルにつき 1 つのプロセッサが与えられるようにしてください。もしさらにプロセッサを割り当てられるなら、500 から 1,000 万セルにつき 1 プロセッサが与えられれば、通常は充分です。

非構造データ (ポリ・データ、非構造格子) に対しては、少なくとも 100 万セルに対し 1 プロセッサが与えられるようにしてください。もしさらにプロセッサを割り当てられるなら、25 万から 50 万セルにつき 1 プロセッサが与えられれば、通常は充分です。

前述のように、これらは経験則に過ぎず、絶対的法則ではありません。常にデータ量に対してどの程度のプロセッサが適切かを評価し、実験するように努めるべきです。そしてもちろん、読み込みたいデータの規模が、ユーザが利用可能な計算機資源の限界を拡大する時が来る可能性は、常にあります。このような状況になれば、データ量の爆発的増大を確実に回避し、データを確実に間引きたくなることでしょう。

3.8 データ量の爆発的増大の回避

ParaView の有するパイプライン・モデルは、試行錯誤による可視化にはとても便利です。コンポーネント間の連携が緩やかであるため、型どおりでない可視化の構築のためのとても柔軟なフレームワークとなっています。また、パイプライン構造のため、設定を素早く容易に追い込むことが出来るようになっています。

この連携方法の欠点は、メモリ使用量が比較的多くなることです。パイプラインの各段階でそれぞれがデータを重複して保持するためです。ParaView は可能な時は常に、パイプラインの各段階がメモリ内の同一の領域のデータを参照するよう、データの**浅いコピー**を行います。しかしながら、新たなデータを作成したり、データの値やトポロジを変更するようなフィルタは全て、それらの結果のための新たなメモリを確保せざるを得ません。ParaView が非常に大きなメッシュに対してフィルタを適用する場合には、フィルタの使用法が適切でなければ即座に利用可能なメモリを全て使い尽くしてしまいます。したがって、大規模なデータセットを可視化する際には、フィルタのメモリ所要量を理解しておくことが重要です。

ただし、以下の助言は**非常に大規模なデータを取扱っていながら、使用可能なメモリが少なくなっている時のためのみ**であることに注意してください。メモリが尽きる心配がなければ、以下の助言は全て無視してください。

構造データを扱っているときは、どのフィルタがデータを非構造データに変更するかを知っておくことは絶対的に重要です。なぜなら、非構造データは、トポロジを明示的に記述する必要があるため、構造データよりもセルあたりのメモリ使用量が大幅に増大するからです。ParaView には、トポロジを何らかの方法で変更するフィルタが数多く存在し、それらのフィルタは生成されるあらゆる種類のトポロジを扱えるデータセットが非構造データのみであるため、データを非構造格子として書き出します。以下にリストされたフィルタは、入力とおおむね等しい新たな非構造格

子トポロジを出力に書き出します。これらのフィルタは、**絶対に**構造データに対しては使用するべきではなく、非構造データに対してのみ注意しながら使用するべきです。

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

技術的には、Ribbon および Tube フィルタもこのリストに含まれます。しかしながら、それらはポリ・データ中の 1 次元のセルにのみ作用するため、入力データは通常小規模で、ほとんど影響はありません。

次の同様な一連のフィルタもまた、非構造格子を出力しますが、一般的にはそのデータ量 (格子数) をいくらか削減します。ただし、このデータ量削減は多くの場合、非構造格子への変換によるオーバーヘッドを補う程ではないことに注意して下さい。また多くの場合、このデータ量の削減は (負荷分散の点では) あまり上手くバランスしないことに注意して下さい。したがって、これらのフィルタは非構造データに対しては注意深く、また構造データに対しては非常に注意深く使用する必要があります。

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

上のリストの項目と同様に、Extract Subset は構造データセットに対してデータ量削減を行いますが、構造データを出力します。したがって、新たなデータが作成されるという注意点は同様ですが、非構造データに変換されるとの心配は不要です。

以下の一連のフィルタもまた非構造データを出力しますが、(3 次元から 2 次元へのような) データの次元の削減を行いますので、出力はずっと小さくなります。し

たがって、これらのフィルタは通常、非構造データに対しても使用することができ、構造データに対しても多少の注意を払えば充分です。

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

これらのフィルタはデータのコネクティビティを全く変更しません。そのかわり、データにフィールド配列のみを付加します。既に存在するデータに対しては浅いコピーが行われます。これらのフィルタは通常、どのようなデータに対しても使用することができます。

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

以下の最後の一連のフィルタは、データを出力に全く追加しない (全ての出力データは浅いコピーによって得られる) か、付加されるデータは入力データの量に依存しないフィルタです。これらはどんな状況でも、まず問題なく使用することができます (ただし、処理時間は要するかもしれません)。

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

上記の分類には上手くあてはまらない、特殊なフィルタも存在します。それらのフィルタの幾つか (今のところ Temporal Interpolator と Particle Tracer) は、データが時間によってどのように変化するかに基づいて処理を行います。したがって、これらのフィルタは 2 ステップ、またはそれ以上の時刻ステップを読込む必要があり、メモリ上で必要なデータの量が倍、またはそれ以上となる可能性があります。また、時間軸方向の処理を行うフィルタの幾つか、例えば Temporal Statistics や、時間軸に沿ってデータをプロットするようなフィルタは、全てのデータをディスクから反復的に読込む必要があります。したがって、たとえ余分なメモリを使用しなくても、非実用的なほど長い処理時間を要する可能性があります。

Programmable Filter もまた、分類が不可能な特殊なケースです。このフィルタはプログラミングされたとおりの処理を行いますので、これらの分類のいずれにもあてはまる可能性があります。

3.9 データを間引く

大規模なデータを扱う際には、可能な限りデータを間引くことが明らかに最良の策であり、それも早い段階で行うほど、良いといえます。ほとんどの大規模データは 3 次元の形状として読込まれますが、所要の形状はそのデータの (表面などの) 面であることがしばしばあります。面データの所要メモリは通常、その元となる立体データよりも大幅に小さいため、早い段階で面データに変換するのが最良といえます。ひとたびそのようにすれば、他のフィルタを比較的安全に適用することができます。

非常に一般的な可視化の操作としては、Contour フィルタを使用して立体データから等値面を抽出することが挙げられます。Contour フィルタは通常、入力より

ずっと小さなデータ量の形状を出力します。したがって、いかなる状況であれ、もし Contour フィルタを使用するのであれば、早い段階で適用するべきです。とはいえ、Contour フィルタは大量のデータを生成する可能性もありますので、設定には注意して下さい。大量の等値面を生成する値を指定すれば、当然ながらそのような状況は起こり得ます。データ中の等値面を生成する値の上下に、ノイズのような高周波の変動が存在すれば、それも大量の不整形な面を生成する原因となります。

立体データの内部を見るもう一つの方法は、Slice フィルタを適用することです。Slice フィルタは、立体データを面によって薄切りにして、立体内の面が切断する部分のデータを見えるようにします。もし大規模データの中で興味深い特徴を有する位置が既に判っていれば、薄切りにするのがそのデータを見る良い方法です。

もしもデータについて**あらかじめ与えられた**情報がほとんどなく、しかも全データセットに対するメモリ量や処理時間を必要とすることなくデータを調べたいのであれば、Extract Subset フィルタを使用してデータの一部の領域を抽出、および間引くことができます。間引かれたデータの量を元のデータよりも大幅に小さくすることは可能でありながら、間引かれたデータは良好に負荷分散されているはずです。もちろん、間引くことによって細かなデータの変化は失われる可能性があり、必要なデータの特徴を発見したら完全なデータセットに戻って可視化を行うべきであることは注意して下さい。

立体データの一部を取り出すことのできるフィルタは幾つかあります。Clip 、Threshold 、Extract Selection、そして Extract Subset はいずれも、何らかの基準によってセルを抽出することができます。しかしながら、抽出されたセルが上手く負荷分散されている可能性はほとんど無く、全くセルが取り除かれないプロセスも存在する可能性に注意して下さい。また、Extract Subset 以外のこれらのフィルタは全て、構造データ型を非構造格子に変換します。したがってこれらは、抽出されたセルが元のデータより少なくとも一桁以上少なくなるのでない限り、使用するべきではありません。

可能であれば、3 次元データを抽出するフィルタを 2 次元の面を抽出するフィルタで置き換えてみて下さい。例えば、データ中のある平面を調べたいのであれば、Clip フィルタよりも Slice フィルタを使用して下さい。もしある範囲内の値を含むセルの領域の位置を調べるのであれば、Threshold フィルタを使って全てのセルを抽出するよりも、Contour フィルタを使ってその範囲の両端となる等値面を生成するようにしてみてください。ただし、このように代わりのフィルタを使用すると、下流側のフィルタに影響する可能性があることに注意して下さい。例えば、Threshold フィルタの後に Histogram フィルタを実行するのは、ほぼ同等の Contour フィルタの後に実行するのとは全く異なった結果となります。

3.10 レンダリング

レンダリングとは、データから実際に目にする画像を生成する処理のことです。データと効率的にインタラクションする能力は、レンダリングのスピードに大きく依存します。コンピュータ・ゲーム市場に牽引された3次元処理のハードウエア・アクセラレーションの進歩によって、高価でないコンピュータでさえ、3次元画像を高速にレンダリングできるようになりました。しかしながらもちろん、レンダリングの速度はレンダリングされるデータの量に比例します。データが大きくなるほど、レンダリング処理は必然的に遅くなります。

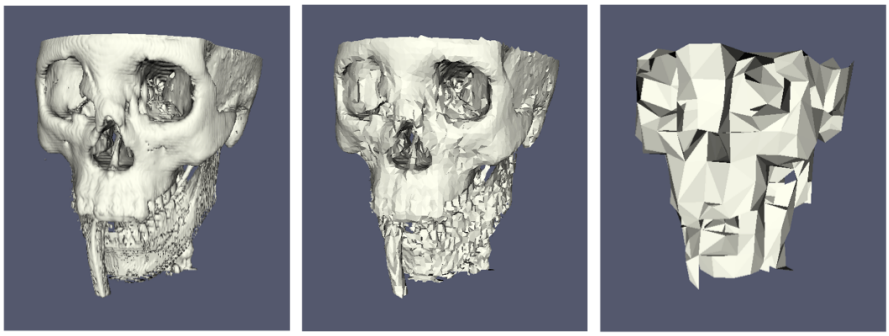
可視化セッションが確実にインタラクティブに実行されるため、ParaViewは2つのモードをサポートしており、それらのモードは必要に応じて自動的に切り替えられます。**スティル・レンダー**と呼ばれる1つ目のモードでは、データは可能な限りのディテールを保持してレンダリングされます。このレンダリング・モードでは、確実にデータの全てが正確に表現されます。**インタラクティブ・レンダー**と呼ばれる2つ目のモードでは、正確さよりも速度が優先されます。このレンダリング・モードでは、データの大きさにかかわらず、高速なレンダリングが行われるよう配慮されます。

マウスによる回転、パン、ズームなどの、3Dビューにおけるインタラクションを行っている間は、ParaViewはインタラクティブ・レンダーを行います。これはインタラクションの間、これらの機能を実用的に保つため、高いフレームレートが必要であるのと、インタラクションの間それぞれのフレームはすぐに次のフレームで置き換えられるため、このモードにおいては精細なディテールはあまり重要でないためです。3Dビューにおけるインタラクションが行われていない時は、ParaViewはデータの全てのディテールが明らかになるよう、スティル・レンダーを使用します。3Dビューでマウスをドラッグしてデータを動かすと、マウスを動かしている間は概略のみがレンダリングされますが、マウスボタンを放すとすぐに完全なディテールまで表現されるのがお判り頂けるでしょう。

インタラクティブ・レンダーは速度と正確さの妥協の産物です。したがって、いつ、どのように粗いディテールが使用されるかについては、多くの設定が関係します。

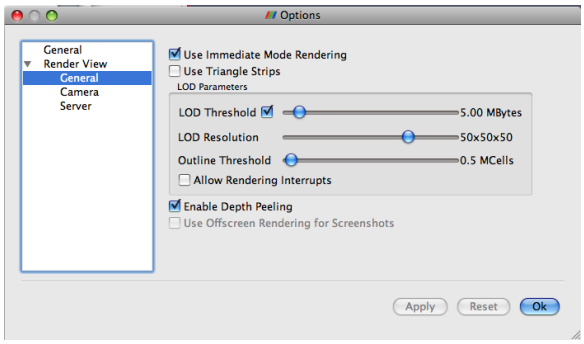
3.10.1 基本的な設定

最も重要なレンダリング設定の幾つかは、LODに関する設定です。インタラクティブなレンダリングの間、形状は低い**ディテールのレベル**(**LOD**)、すなわちより少ないポリゴンで表現された近似的な形状に置き換えられることがあります。



形状の近似化の解像度は、変更することができます。形状簡略化のアルゴリズムは、ポリゴンを粗い格子に沿って配置するように動作します。上の画像では、左の画像は完全な解像度です。中央の画像は 50^3 分割の格子による形状簡略化の結果で、右の画像は 10^3 分割の格子による形状簡略化の結果です。

3 次元レンダリングの設定は、メニューの Edit → Settings (Mac では ParaView → Preferences) で現れる設定ダイアログボックスにあります。ダイアログを開くと、基本的なレンダリング設定は Render View → General 以下にあります。



レンダリング性能に関する設定は、以下のような意味があります。

Use Immediate Mode Rendering (イミディエイト・モード・レンダリングを使用する) これをチェックすると、形状はグラフィックス・カードに送られてイミディエイト・レンダリングが行われます。チェックを外すと、形状はより効率的なレンダリングのためにディスプレイ・リストにまとめられます。通常はディスプレイ・リストの方が高速にレンダリングされますが、最初のフレームのレンダリングの間にディスプレイ・リストを作成する時間、およびそれを保持するメモリが必要となります。

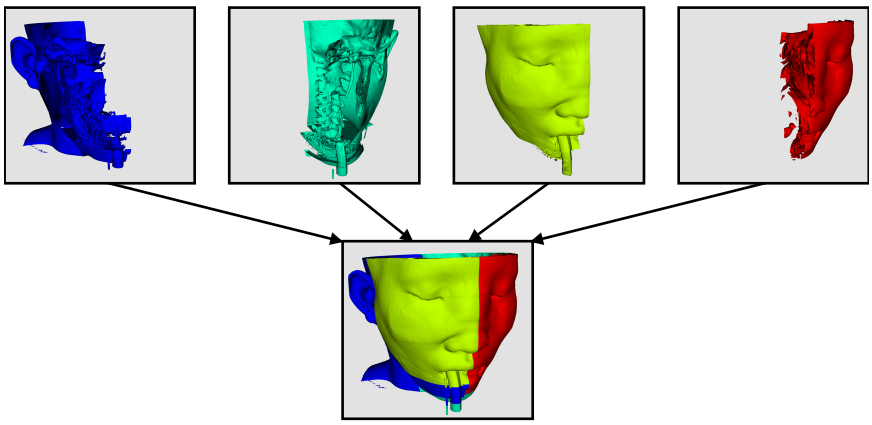
Use Triangle Strips (トライアングル・ストリップを使用する) チェックを外すと、データはポリ・データによって定義されたとおりにレンダリングされます。チェックすると、データはトライアングル・ストリップに変換されます。トライアングル・ストリップはグラフィックス・カードに効率的に転送され、より高速にレンダリングされることもありますが、通常はそうではありません。

LOD Threshold (LOD しきい値) どのような時に形状を簡略化された形状で置き換えるかを制御します。チェックボックスは、この機能自体をオンまたはオフにします。オンの場合は、スライダによってこの機能の動作するしきい値を与えます。形状のデータ量がこのしきい値未満であれば、その形状はレンダリングするのに充分小さいと見なされます。形状のデータ量がこのしきい値より大きければ、簡略化された形状がレンダリングに使用されます。

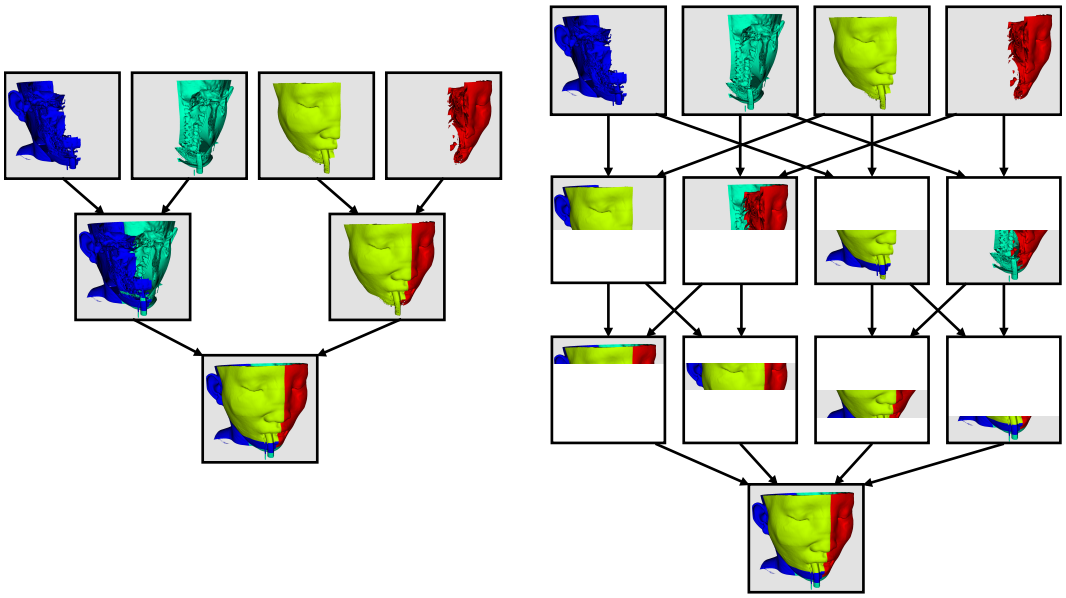
Allow Rendering Interrupts (レンダリングの中断を許可する) チェックされている場合、スティル・レンダーを中断してインタラクティブ・レンダーを行えるようになります。

3.10.2 基本的な並列レンダリング

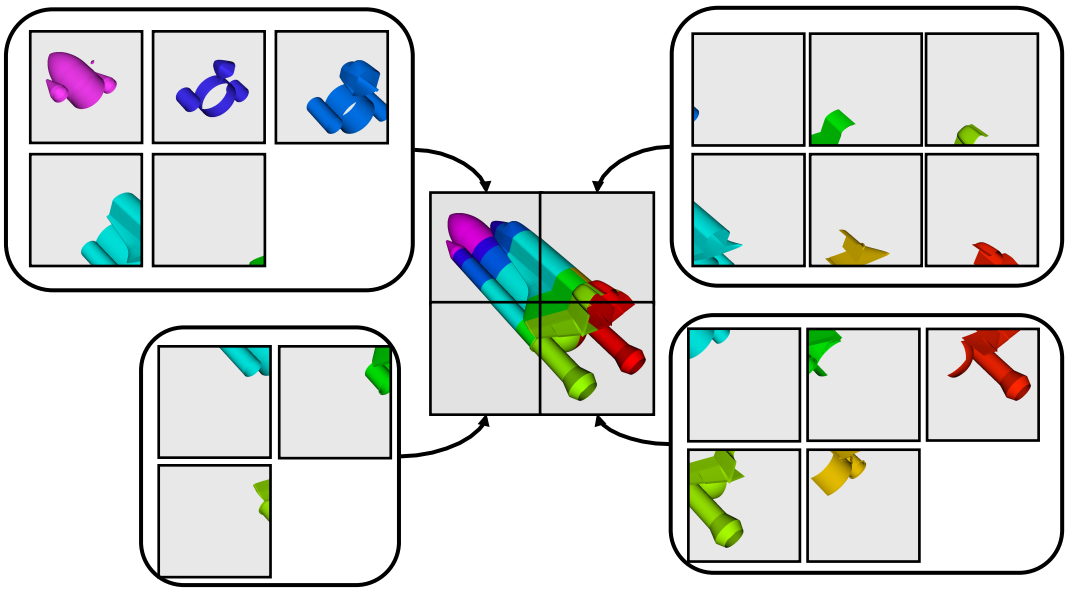
並列可視化を行う際には、レンダリングに至るまで、およびレンダリング自体の全てのプロセスにおいて、データが分割されているよう注意します。ParaView では、**IceT** と呼ばれる並列レンダリング・ライブラリが使用されます。IceT では、**ソート・ラスト**・アルゴリズムが、並列レンダリングに使用されます。この並列レンダリング・アルゴリズムによって、それぞれのプロセスはそれぞれに与えられた分割された形状を独立にレンダリングし、その結果の部分的な画像を**重畳**して最終的な画像を生成することができます。



上の図は大幅に簡略化したものです。IceT には、**2 分木法**や**バイナリ・スワップ法**のような、いくつかの段階を用いて処理を効率的にプロセス間に分割する、複数の並列化された画像重畳アルゴリズムが含まれています。



ソート・ラスト法による並列レンダリングの素晴らしい点は、その効率がレンダリングされるデータの量に全く依存しないことです。そのため、この手法は非常にスケーラブルで、大規模なデータに適しています。しかしながら、並列レンダリングのオーバーヘッドは画像中の画素数に対して線形に増加します。したがって、レンダリング設定の幾つかは画像の大きさに関するものです。

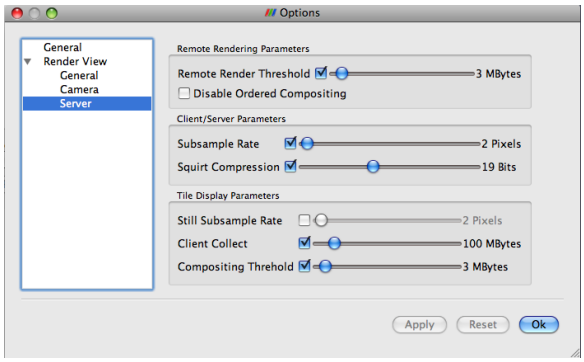


IceT は、タイリング表示されたディスプレイ (多数の並べられたモニタやプロジェクタによって構成される、大型・高解像度のディスプレイ) を駆動する能力があります。ソート・ラスト・アルゴリズムをタイリング・ディスプレイに使用するのは、重畳すべき画素数が非常に多くなるため、やや直感的ではありません。しかしながら IceT では、それぞれのプロセスに存在するデータの特定の局所性を利用して、必

要な重畳処理を大幅に低減するようになっています。この空間的な局所性は、データに対し D3 フィルタを適用することで強制することができます。

並列レンダリングにはオーバーヘッドが存在するため、ParaView ではいつでも並列レンダリングをオフにすることができます。並列レンダリングがオフの時は、形状データが実際に表示が行われる所に転送されます。明らかに、これはレンダリングするデータが小さいときにのみ行うべきです。

3.10.3 並列レンダリングの設定



他の 3 次元レンダリングの設定と同じように、並列レンダリングの設定も Edit → Settings (Mac では ParaView → Preferences) によって現れる設定ダイアログボックスに配置されています。ダイアログを開くと、並列レンダリングの設定は Render View → Server 以下にあります。それぞれの設定は、以下のような意味があります。

Remote Render Threshold (リモート・レンダリングを行うしきい値) このチェックボックスは、リモート・レンダリングをオンまたはオフにします。スライダによって、並列レンダリングを行うしきい値を変更することができます。形状のデータ量がこのしきい値未満であれば、形状データが表示が行われる所 (通常はクライアント) に転送されます。

Disable Ordered Compositing (順序付け重畳を行わない) ボリューム・レンダリングおよび透明なポリゴンが正しく表示されるには、順序付け重畳と呼ばれる特別な並列レンダリング・モードが必要です。しかしながら、このモードには追加の計算処理とメモリが必要です。このチェックボックスをオンにすると、順序付け重畳を行わなくなります。

Subsample Rate (間引き率) 並列レンダリングのオーバーヘッドは、生成された画像の大きさに比例します。したがって、インタラクティブなレンダリングは、画像を間引く割合を指定することで高速化することができます。このチェックボックスがチェックされている時は、インタラクティブ・レンダーでは低解像度の画像を生成し、表示される時にその画像を拡大するようになります。こ

の設定はインタラクティブ・レンダーの時のみ使用されます。スティル・レンダーの時は、常に完全な解像度の画像が使用されます。

Squirt Compression (Squirt 圧縮) 画像がサーバからクライアントに転送される前に、その画像を **SQUIRT** と呼ばれるアルゴリズムによって圧縮することができます。このチェックボックスでは、この SQUIRT 圧縮をオンまたはオフにします。圧縮をさらに効果的にするために、SQUIRT では圧縮前に画像の色深度を削減することができます。スライダによって、削減する色深度のビット数を指定します。デフォルトの 19 ビットでは、ほとんどのディスプレイではまず違いが判りません。スライダによる設定は、インタラクティブ・レンダーの時のみ有効です。スティル・レンダーの時は、常にフルカラーの色深度が使用されます。

Still Subsample Rate (スティル・レンダーの間引き率) タイリング・ディスプレイは色々な用途に使われます。例えば、小規模な共同作業や大規模なプレゼンテーションなどです。大規模なプレゼンテーションでは、観衆がディスプレイの全ての画素を見ることはまずありません。そのような場合には、このオプションによってスティル・レンダーの画像を間引いて、レンダリングの時間を大幅に短縮することができます。

Client Collect (クライアントに集約する) タイリング・ディスプレイ・モードでは、並列レンダリングによる画像はデスクトップでなく、タイリングされたディスプレイに送られます。したがって、クライアントは全てのデータをローカルにレンダリングしなければなりません。この設定では、クライアントに送られるデータ量の上限を指定します。データが指定されたしきい値より大きければ、クライアントにはデータが収まるバウンディング・ボックスのみが表示されます。

Compositing Threshold (重畳を行うしきい値) 重畳を行うしきい値とは、タイリング・ディスプレイに対するリモート・レンダリングを行うしきい値と同等のものです。ただし、重畳を行うことによる得失は異なります。タイリングされたディスプレイは解像度が高いために重畳のオーバーヘッドは大きく、そのかわり通常は高速なネットワーク内で行われるため、形状の集約のオーバーヘッドは小さくなります。したがって、重畳を行うしきい値をリモート・レンダリングを行うしきい値よりも高くすると、効果のあることがあります。形状のデータ量がこの重畳を行うしきい値より小さければ、表示される形状が全てのレンダリング・ノードに送られ、それぞれのレンダリング・ノードはそれぞれが受け持つタイリング・ディスプレイの部分に直接レンダリングされます。

3.10.4 大規模データのための設定

デフォルトのレンダリング設定は、ほとんどのユーザに適しています。しかしながら、非常に大規模なデータを扱う時は、設定を追い込むことが有効であることがあります。最適な設定はデータおよびParaViewが実行されるハードウエアによって変わりますが、以下のような助言に従うと良いでしょう。

1. Use Immediate Mode Renderingをオンにして、Use Triangle Stripsをオフにして下さい。これらの設定は両方とも、効率的にレンダリング可能な構造に変換することが目的です。しかしながら、データが大規模なときは大抵、処理とメモリのオーバーヘッドがその効率化に見合いません。実際、メモリの限界が拡大されれば、この設定によって性能は低下します。

2. LOD Thresholdを**オフ**にしてみてください。大規模データにおいては、形状の簡略化には長い時間がかかり、データに極端な曲がり部や特殊なコネクティビティがあれば、良い結果が得られないことがあります。LODによって性能が改善されるのであれば、LOD Resolutionの設定スライダを一番右 $(10 \times 10 \times 10)$ に動かしてみて下さい。

3. 常にリモート・レンダリングをオンにしておいてください (Remote Render Thresholdの隣のチェックボックスで切り替えられます)。リモート・レンダリングによってサーバ全体のレンダリング能力が使われ、クライアントに画像が転送されます。リモート・レンダリングがオフであれば、形状データがクライアントに転送されます。大規模なデータに対しては、形状データを転送するよりも画像を転送する方が必ず高速です。

4. 間引き処理をオンにして、Subsample Rateを必要に応じて調整して下さい。画像の重畳に時間がかかる場合は、クライアントとサーバの間の接続が狭帯域であるか、非常に大きな画像をレンダリングしているときには間引き率を大きくすると、インタラクティブ・レンダリングの性能が大幅に改善されることがあります。

5. Squirt Compressionがオンであることを確認して下さい。この設定はデスクトップへの画像表示性能に大きな影響があります。また、この設定によって引き起こされる画質低下は最小限であり、かつ影響を受けるのはインタラクティブ・レンダリング時のみです。

第4章　さらに情報を得るには

このチュートリアルに参加頂き、ありがとうございました。ParaViewを使って、大規模データの可視化を始めるのに充分なだけ勉強されたことでしょう。以下には、さらに情報を得るための情報源を挙げます。

The ParaView Guideは、ParaViewと一緒に持っておくのにとても良い情報源です。ここで学んだ以外の多くの使用法や、多くの機能に関するより詳細な説明が得られます。

Amy Henderson Squillacote. *The ParaView Guide*. Kitware, Inc., 2008. ISBN-10 1-930934-21-1.

ParaView Wikiは、ParaViewをセットアップして使用するのに役立つ情報が満載です。特に、並列ParaViewサーバをインストールしたい方は、そのためのビルドおよびインストールのページを必ずご参照ください。

http://www.paraview.org/Wiki/ParaView

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

可視化や、ParaViewで利用可能なフィルタに関する詳細事項をさらに学ぶことに興味がある場合は、以下の可視化に関するテキストを入手することをご一考下さい。

Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit*. Kitware, Inc., fourth edition, 2006. ISBN 1-930934-19-X.

ParaViewをカスタマイズしようとされているのであれば、上記の本およびウエブ・ページに多くの情報があります。ParaViewが依存する可視化ライブラリのVTK、GUIライブラリのQtの使用法に関する情報については、下記の本が参考となります。

Kitware Inc. *The VTK Users Guide*. Kitware, Inc., 2006.

Jasmin Blanchette and Mark Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall, 2006. ISBN 0-13-187249-4 (邦訳: 杵渕 聡、杉田 研治 訳「入門 Qt 4 プログラミング」オライリー・ジャパン、2007年、ISBN 978-4-87311-344-9).

もし並列可視化の設計や VTK パイプラインの機能に興味があるのであれば、以下の技術論文があります。

James Ahrens, Charles Law, Will Schroeder, Ken Martin, and Michael Papka. “A Parallel Approach for Efficiently Visualizing Extremely Large, Time-Varying Datasets.” Technical Report #LAUR-00-1620, Los Alamos National Laboratory, 2000.

James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, and Michael Papka. “Large-Scale Data Visualization Using Parallel Data Streaming.” *IEEE Computer Graphics and Applications*, 21(4): 34–41, July/August 2001.

Andy Cedilnik, Berk Geveci, Kenneth Moreland, James Ahrens, and Jean Farve. “Remote Large Data Visualization in the ParaView Framework.” *Eurographics Parallel Graphics and Visualization 2006*, pg. 163–170, May 2006.

James P. Ahrens, Nehal Desai, Patrick S. McCormic, Ken Martin, and Jonathan Woodring. “A Modular, Extensible Visualization System Architecture for Culled, Prioritized Data Streaming.” *Visualization and Data Analysis 2007, Proceedings of SPIE-IS&T Electronic Imaging*, pg 64950I-1–12, January 2007.

John Biddiscombe, Berk Geveci, Ken Martin, Kenneth Moreland, and David Thompson. “Time Dependent Processing in a Parallel Pipeline Architecture.” *IEEE Visualization 2007*. October 2007.

もし ParaView の並列レンダリングのアルゴリズムや構造に興味があるのであれば、それらに関する技術論文もまた多数あります。

Kenneth Moreland, Brian Wylie, and Constantine Pavlakos. “Sort-Last Parallel Rendering for Viewing Extremely Large Data Sets on Tile Displays.” *Proceedings of IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, pg. 85–92, October 2001.

Kenneth Moreland and David Thompson. “From Cluster to Wall with VTK.” *Proceddings of IEEE 2003 Symposium on Parallel and Large-Data Visualization and Graphics*, pg. 25–31, October 2003.

Kenneth Moreland, Lisa Avila, and Lee Ann Fisk. “Parallel Unstructured Volume Rendering in ParaView.” *Visualization and Data Analysis 2007, Proceedings of SPIE-IS&T Electronic Imaging*, pg. 64950F-1–12, January 2007.

謝辞

Amy Squillacote 氏には、このチュートリアルに教材を提供して下さったことに感謝いたします。そしてもちろん、Kitware、サンディア、CSimSoft の各位には、多大なる労力によって ParaView を現在のようにして下さったことに感謝申し上げます。

サンディアは、DE-AC04-94AL85000 の契約に基づいて、米国エネルギー省国家核安全保障局のために、ロッキード・マーチン系列のサンディア・コーポレーションによって運営される、多数のプロジェクトからなる研究所です。

日本語版 謝辞

このドキュメントは、タイトルページに記した著者らによる、“Large Scale Visualization with ParaView: Supercomputing 2008 Tutorial” の日本語訳です。原著者各位には、日本語への翻訳、およびその配布を快諾くださったのみならず、原文の LaTeX ソース及び画像ファイル一式を提供頂きました。感謝いたします。

索 引