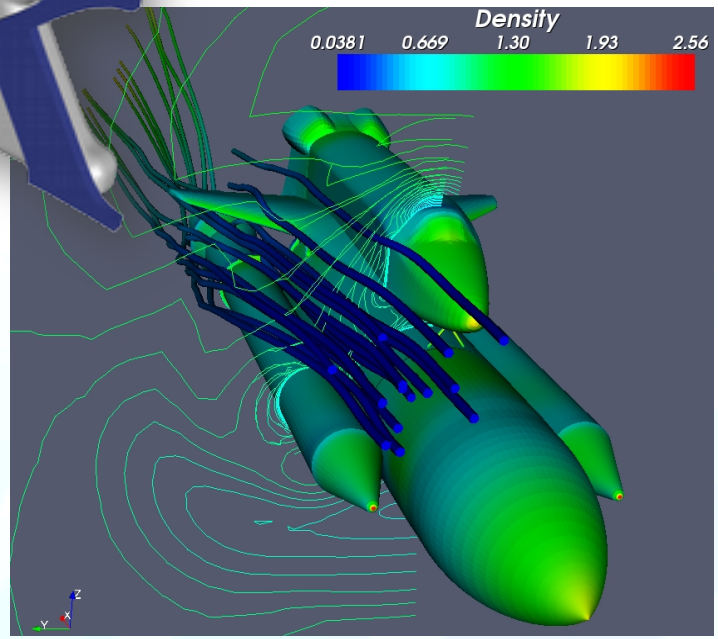

Kitware
eyeing DATA
utkarsh.ayachit
understanding VTK data model

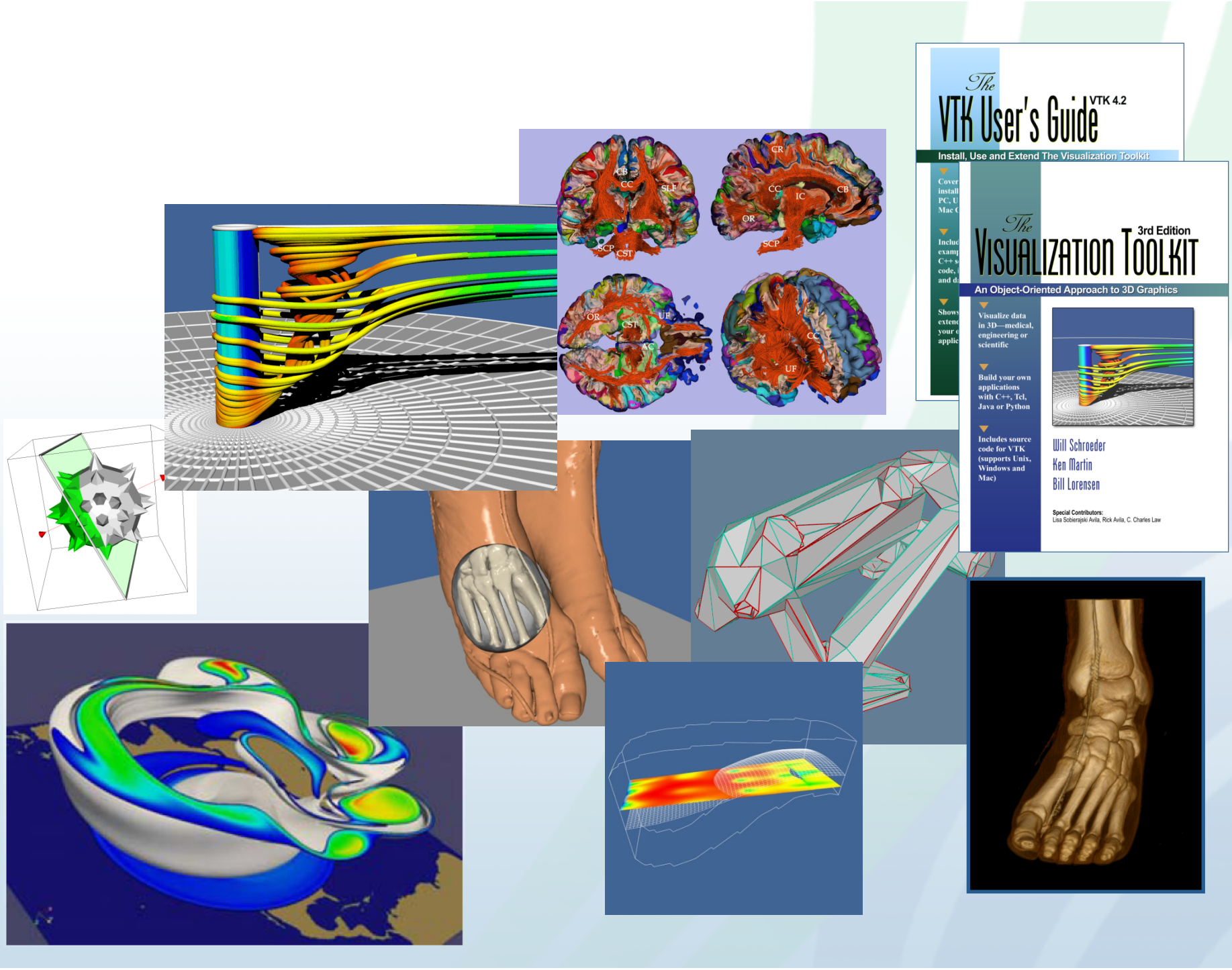

http://s192.photobucket.com/albums/z202/vendettachrncls/movie-shows/?action=view&current=matrixnumbersandletters.jpg&


elegance of structure
http://www.cepolina.com/tensile_structure_cone_column.html


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


Density
0.0381
0.669
1.30
1.93
2.56

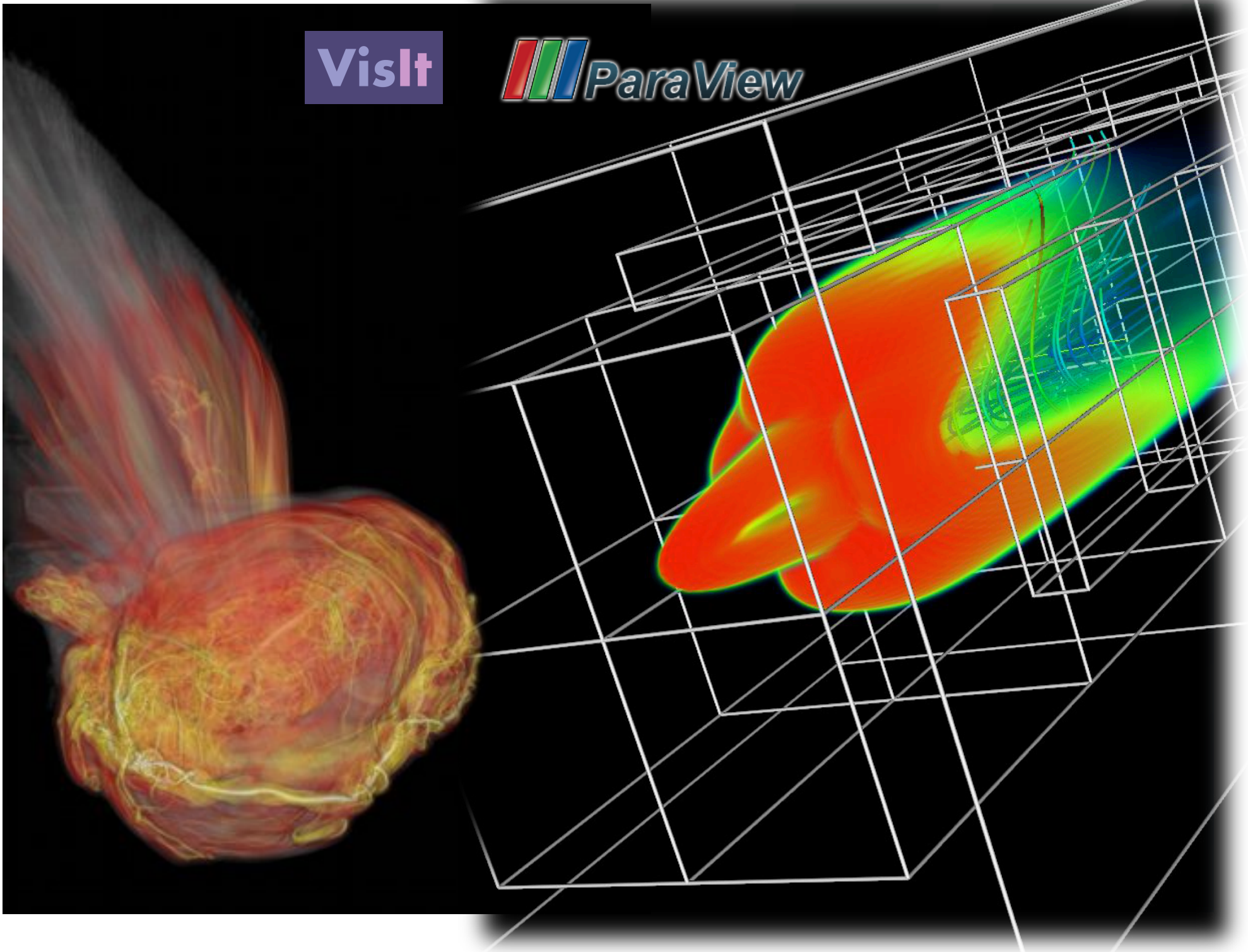
Kitware

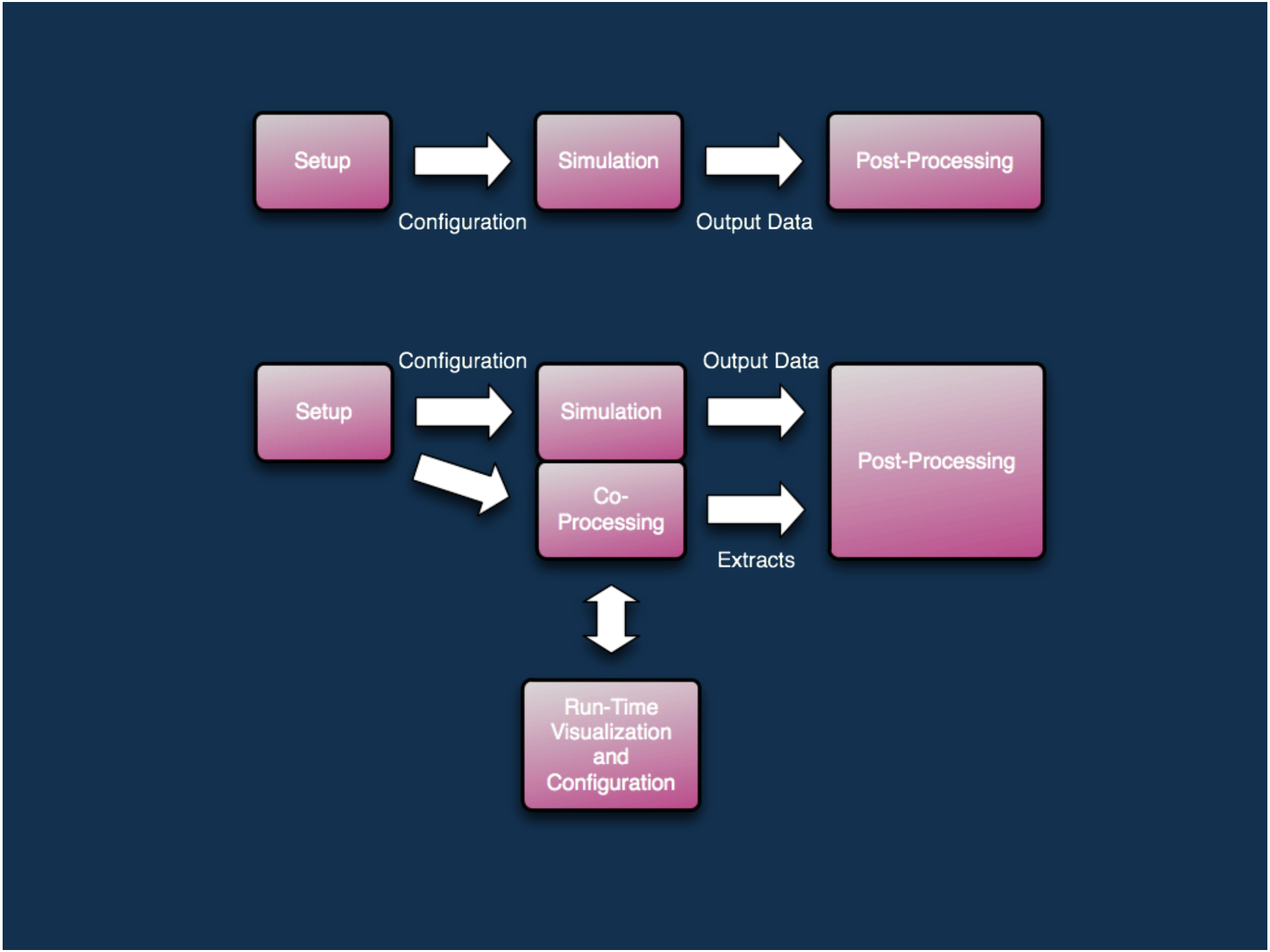

The
VTK User's Guide
VTK 4.2
Install, Use and Extend The Visualization Toolkit
The
Visualization Toolkit
3rd Edition
An Object-Oriented Approach to 3D Graphics
Visualize data in 3D—medical, engineering or scientific
Build your own applications with C++, Tcl, Java or Python
Includes source code for VTK (supports Unix, Windows and Mac)
Will Schroeder
Ken Martin
Bill Lorensen
Special Contributors:
Lisa Sobierajski Avila, Rick Avila, C. Charles Law
CB
CC
SLF
SCP
CST
CR
IC
OR
UF
AC

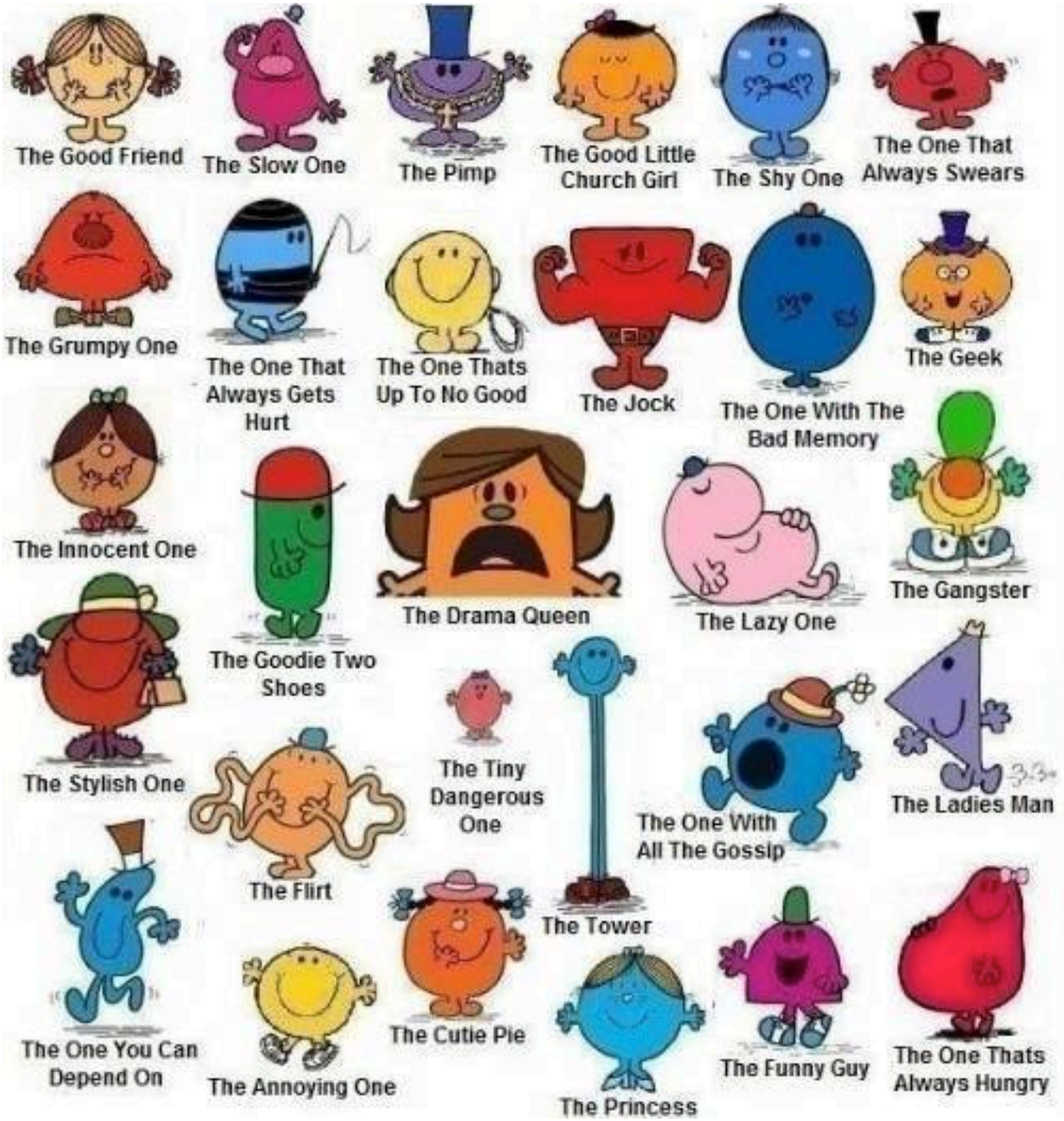
# DO-I-CARE-O-METER

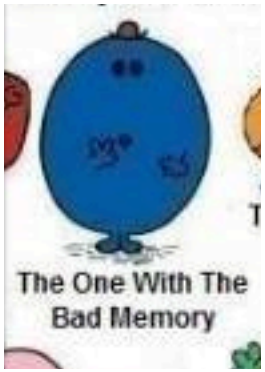

http://deliat.wordpress.com/category/thoughts-and-things-from-wonderland/

deliat.wordpress.com


VisIt
ParaView


Setup
Configuration
Simulation
Output Data
Post-Processing
Setup
Configuration
Simulation
Co-
Processing
Output Data
Post-Processing
Extracts
Run-Time
Visualization
and
Configuration


The Good Friend
The Slow One
The Pimp
The Good Little Church Girl
The Shy One
The One That Always Swears
The Grumpy One
The One That Always Gets Hurt
The One Thats Up To No Good
The Jock
The One With The Bad Memory
The Geek
The Innocent One
The Goodie Two Shoes
The Drama Queen
The Lazy One
The Gangster
The Stylish One
The Tiny Dangerous One
The Tower
The One With All The Gossip
The Ladies Man
The Flirt
The One You Can Depend On
The Annoying One
The Cutie Pie
The Princess
The Funny Guy
The One Thats Always Hungry


The One With The
Bad Memory

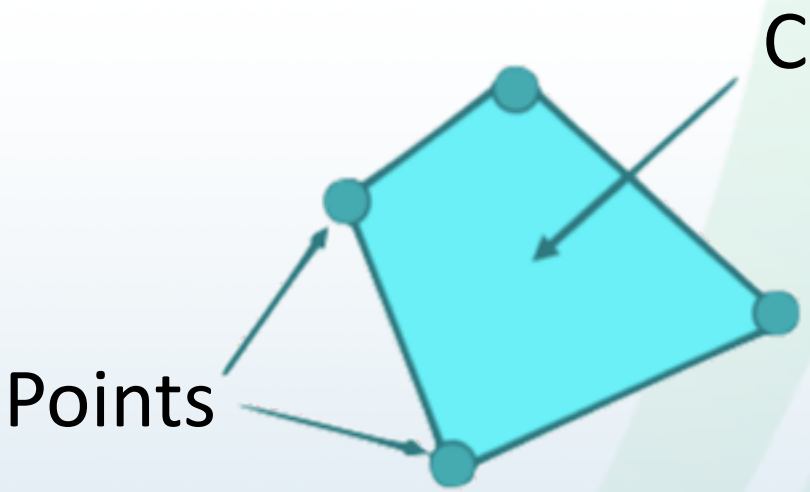

DIG IN
http://mychurchgraphics.com/slides.html

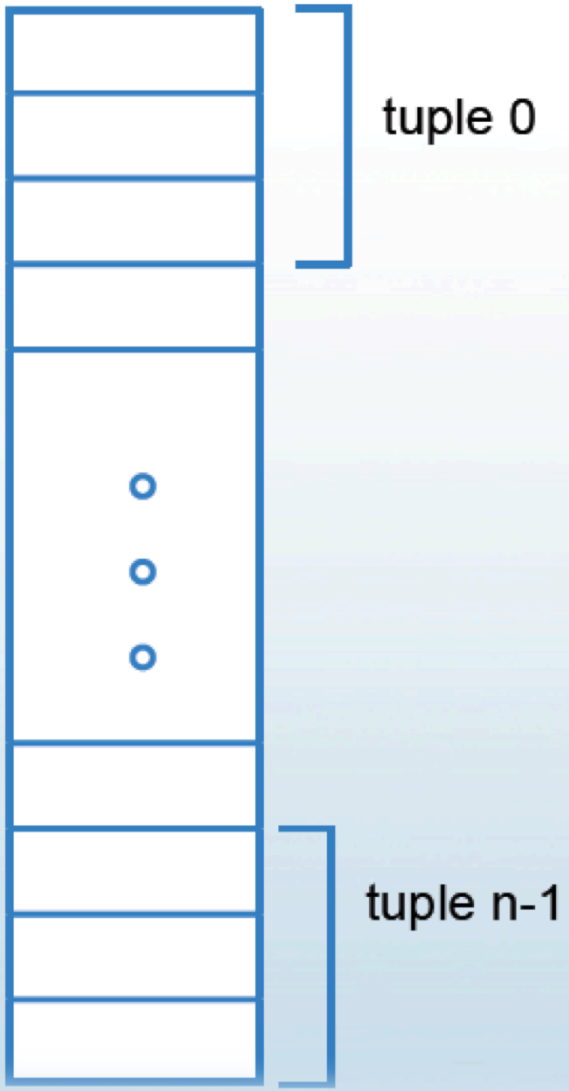
# VTK Layers

- Core
- Data Model
- Execution Model
- Components

# Execution Model


Sales DataBase
TimeKey
CustomerKey
ShipperKey
ProductKey
EmployeeKey
ItemFreight
ItemQuanity
ItemAll
Discount
Shipper
ShipperKey
ShipperID
ShipperName
Product
ProductKey
ProductID
CategoryID
Price
Employee
Title
Address
City
Region
PCode
Country
Phone
Mphone
Fax
Email
Data Model


Cell
Points

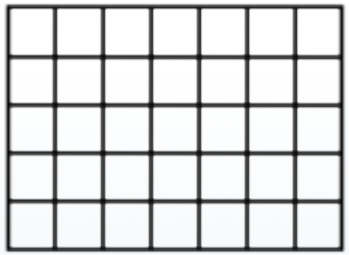



# vtkDataArray

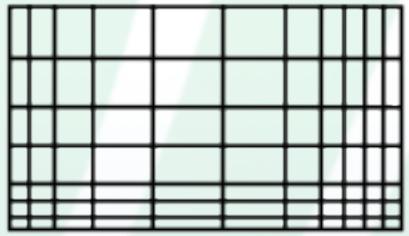
-array of *n* tuples
-each tuple has *m* components
-implemented as nxm C-array
-data-type determined by class

Kitware

vtkImageData

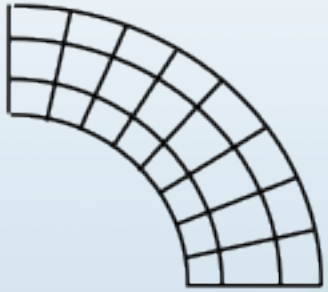


vtkRectilinearGrid

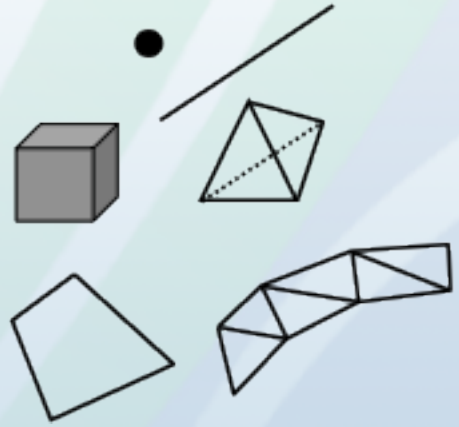


vtkPolyData

vtkStructuredGrid



vtkUnstructuredGrid

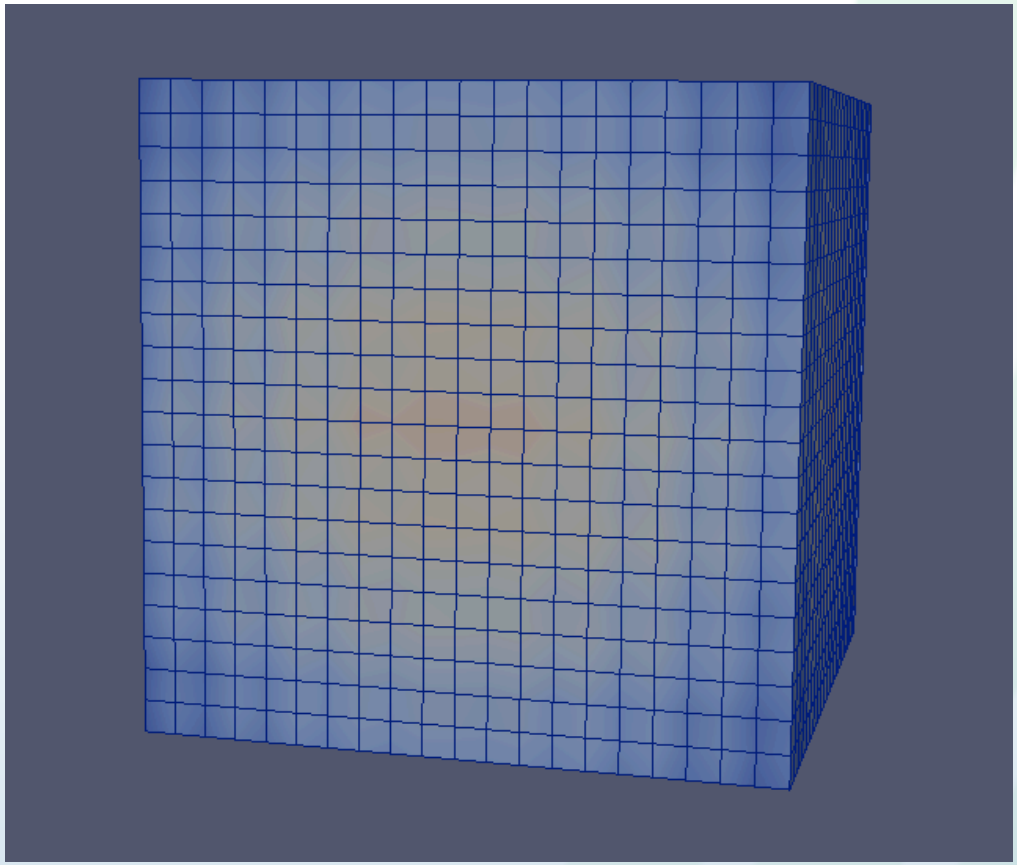


Kitware

# vtkDataSet

```
GetNumberOfPoints()
GetNumberOfCells()
GetPoint(…)
GetCell(…)
GetCellPoints(…)
GetPointCells(…)
GetCellNeighbors(…)
FindCell(…)
```

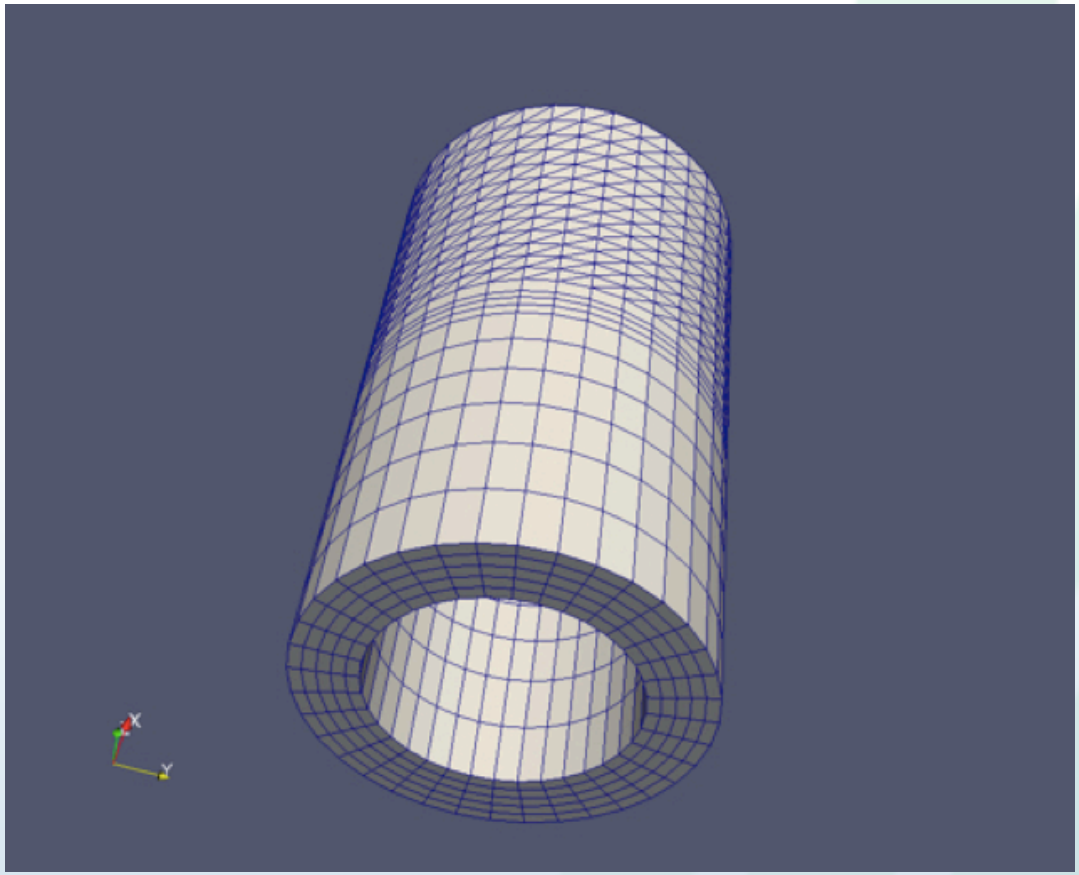
vtkImageData



-Dimensions
-Origin
-Spacing

Kitware

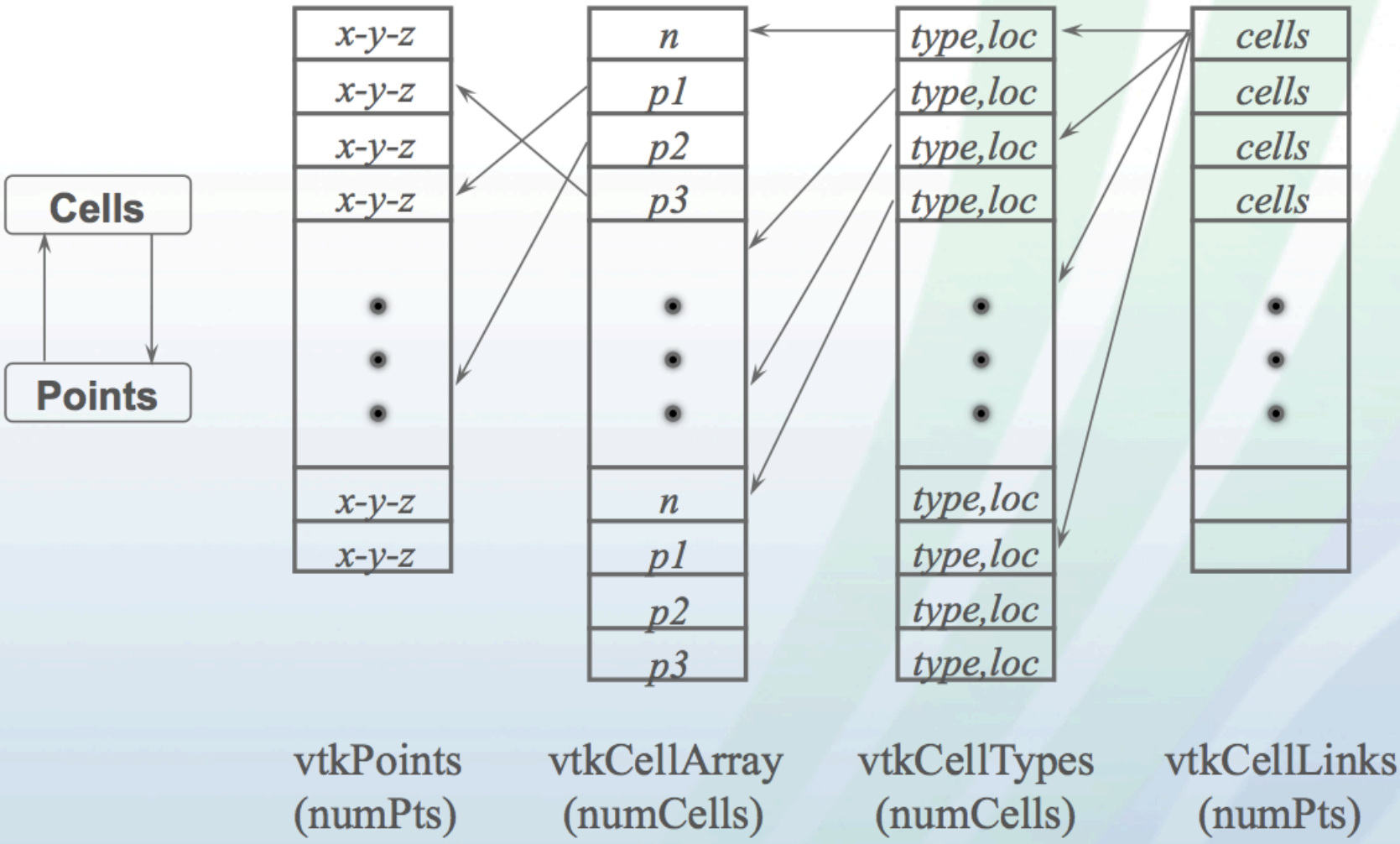
vtkUnstructuredData



(continued…)

Kitware

# Unstructured Data Structure




Kitware

# vtkDataSet

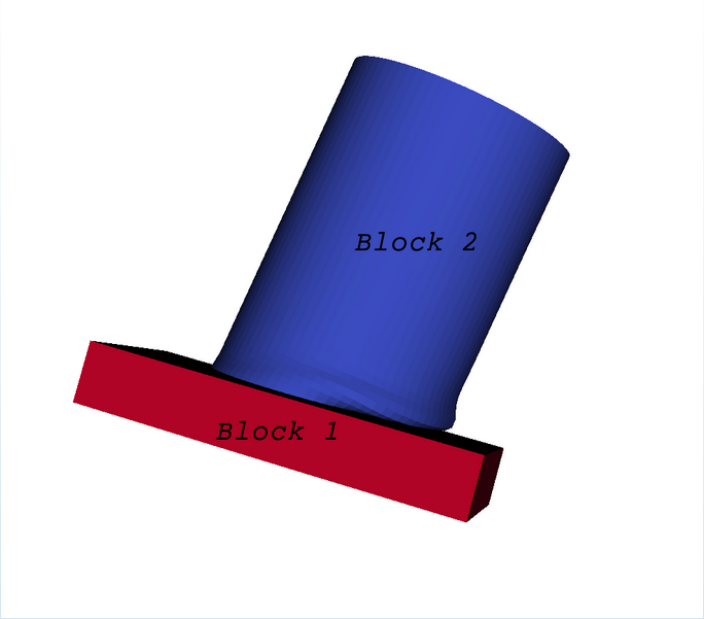
```
GetNumberOfPoints()
GetNumberOfCells()
GetPoint(…)
GetCell(…)
GetCellPoints(…)
GetPointCells(…)
GetCellNeighbors(…)
FindCell(…)
```

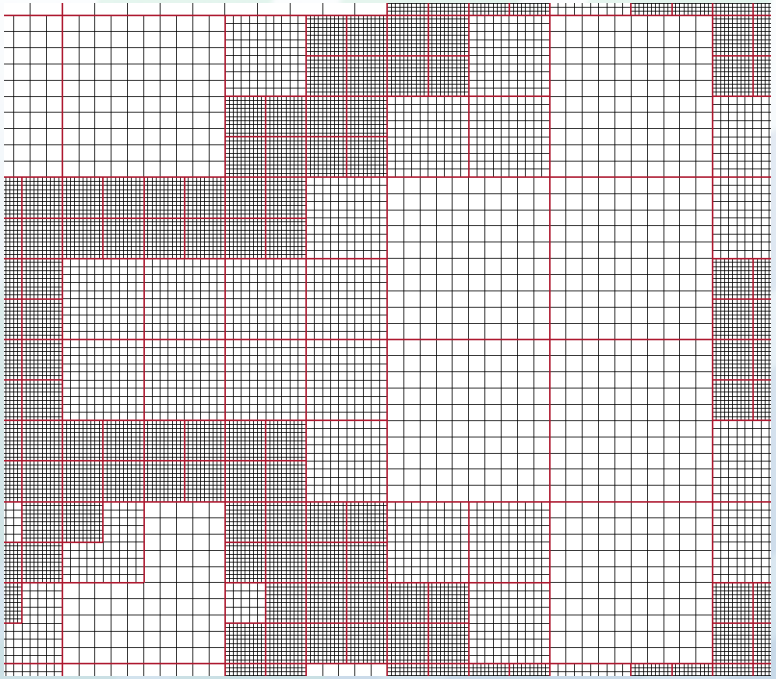
```
vtkIdType vtkImageData::FindCell(double x[3], ...)
{
  int idx[3];

    // Compute squared distance of point x from the boundary
   double dist2 = 0.0;

   ...

   for (int i=0; i<3; i++)
     {
     int minIdx = extent[i*2];
     int maxIdx = extent[i*2+1];
     int negSpacing = (spacing[i] < 0);
     double minBound = bounds[i*2 + negSpacing];
     double maxBound = bounds[i*2 + (1-negSpacing)];

     if ( idx[i] < minIdx )
       {
       idx[i] = minIdx;
       pcoords[i] = 0.0;
       double dist = x[i] - minBound;
       dist2 += dist*dist;
       }
     else if ( idx[i] >= maxIdx )
       {
       if (maxIdx == minIdx)
         {
         idx[i] = minIdx;
         pcoords[i] = 0.0;
         }
       else
         {
         idx[i] = maxIdx-1;
         pcoords[i] = 1.0;
         }
       double dist = x[i] - maxBound;
       dist2 += dist*dist;
       }
     }

   // Check squared distance against the tolerance
   if (dist2 > tol2)
     {
     return -1;
     }
    ...
  return this->ComputeCellId(idx);

}
```

```
vtkIdType vtkPointSet::FindCell(double x[3], ...)
{
  vtkIdType foundCell;


  // Check to see if the point is within the bounds of the data.  This is not
  // a strict check, but it is fast.
  double bounds[6];
  this->GetBounds(bounds);
  if (    (x[0] < bounds[0]) || (x[0] > bounds[1])
      || (x[1] < bounds[2]) || (x[1] > bounds[3])
      || (x[2] < bounds[4]) || (x[2] > bounds[5]) )
    {
    return -1;
    }

  if ( !this->Locator )
    {
    this->Locator = vtkPointLocator::New();
    this->Locator->Register(this);
    this->Locator->Delete();
    this->Locator->SetDataSet(this);
    this->Locator->BuildLocator();
    }


   ...



  // Now find the point closest to the coordinates given and search from the
  // adjacent cells.
  vtkIdType ptId = this->Locator->FindClosestPoint(x);
  if (ptId < 0) return -1;
  this->GetPointCells(ptId, cellIds);
  foundCell = FindCellWalk(this, x, gencell, cellIds,
                           tol2, subId, pcoords, weights,
                           visitedCells, ptIds, neighbors);
  if (foundCell >= 0) return foundCell;


  ...



  // Could not find the cell.
  return -1;
}
```

# vtkCompositeDataSet

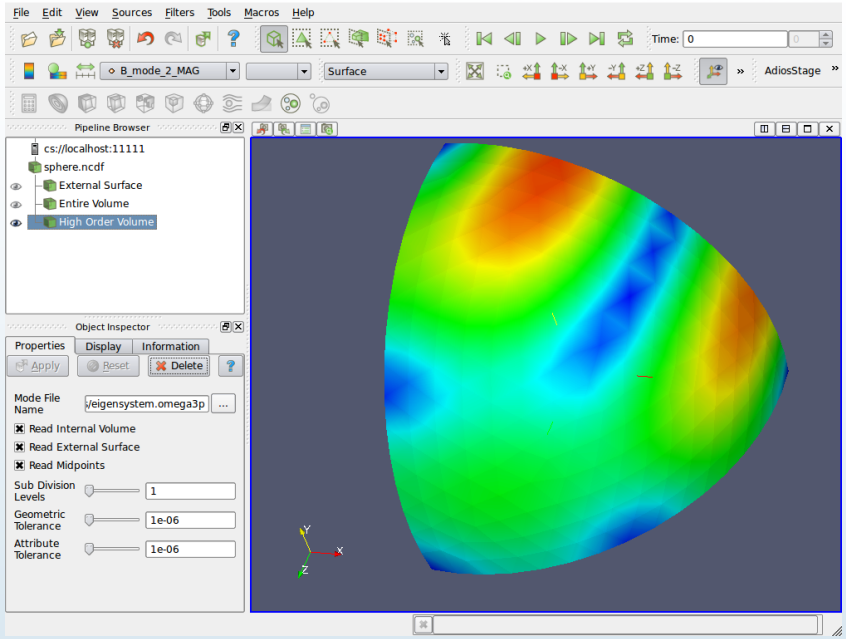
-vtkMultiBlockDataSet
-vtkNonOverlappingAMR
-vtkOverlappingAMR

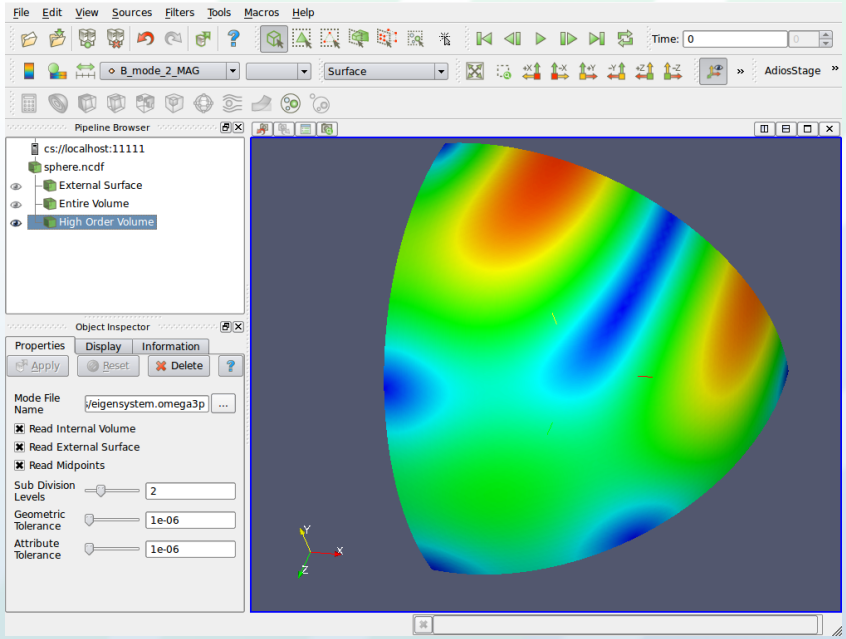


Kitware

# vtkGenericDataSet

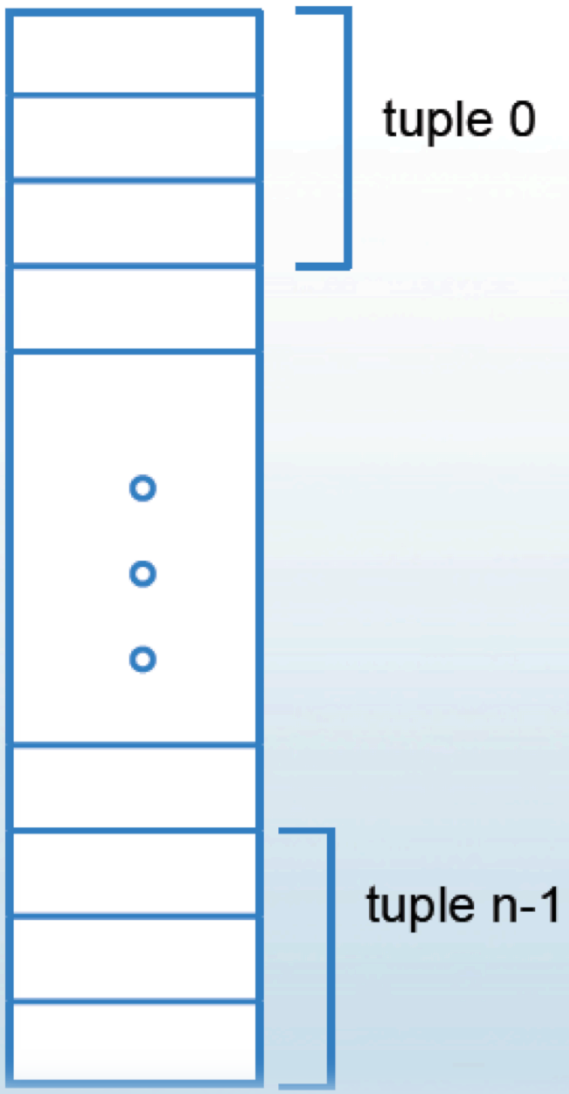



Kitware

# Challenges / Ongoing Work

- Memory
  - Adaptable layout
  - Sharing data / Avoiding copy
  - Meta-data > Heavy-data




# vtkDataArray

-array of *n* tuples
-each tuple has *m* components
-implemented as nxm C-array
-data-type determined by class

*-what about mxn arrays?*

Kitware

# Challenges / Ongoing Work

-Memory
    -Adaptable layout
    -Sharing data / Avoiding copy
    -Meta-data > Heavy-data

# Challenges / Ongoing Work

-Memory
    -Adaptable layout
    -Sharing data / Avoiding copy
    -Meta-data > Heavy-data

Kitware

*that's all, folks!*

Kitware