Information Visualization in VTK

The Titan Project
## Tutorial Outline

- **Introduction and Motivation (15 minutes)**
  - Project Background and Scope
  - Use Case: Cyber Defense

- **VTK InfoVis Data Structures (30 minutes)**
  - Data Structures
    - Tables
    - Trees
    - Graphs
  - Database Access
  - Table, Tree and Graph Readers

- **VTK InfoVis Components (45 minutes)**
  - Data Conversions
  - Graph/Tree Layout Strategies
  - Qt Model Adapters
  - Views/Displays
  - Selection
  - Geographic Visualization

- **Short Break**

- **Analysis Capabilities (50 minutes)**
  - Graph Algorithms
  - Statistics
  - MATLAB interface

- **Algebraic Methods (20 minutes)**

- **Building Applications (30 minutes)**
  - Script Language Support
    - C++
    - Java
    - .Net/Com Interfaces

- **OverView (30 minutes)**
  - Application
  - Plugins
Tutorial Outline

• Introduction and Motivation (15 minutes)
  • Project Background and Scope
  • Use Case: Cyber Defense

• VTK InfoVis Data Structures (30 minutes)
  • Data Structures
    • Tables
    • Trees
    • Graphs
  • Database Access
  • Table, Tree and Graph Readers

• VTK InfoVis Components (45 minutes)
  • Data Conversions
  • Graph/Tree Layout Strategies
  • Qt Model Adapters
  • Views/Displays
  • Selection
  • Geographic Visualization

• Short Break

• Analysis Capabilities (50 minutes)
  • Graph Algorithms
  • Statistics
  • MATLAB interface

• Algebraic Methods (20 minutes)

• Building Applications (30 minutes)
  • Script Language Support
    • C++
    • Java
    • .Net/Com Interfaces

• OverView (30 minutes)
  • Application
  • Plugins
Introduction / Motivation

**What is Titan?** Led by Sandia National Laboratories, in collaboration with Kitware and Indiana University, the Titan Informatics Project is a substantial expansion of VTK to support informatics and analysis.

**Why Titan?**
- Focused on Algorithms (Graph, Statistics, Algebraic Methods)
- Open, Flexible and Extensible
- Based on Scalable Architecture (*some work done, more coming*)

**How do I use it?** In the same way that scientific visualization applications are built with VTK, you can now build information visualization and analysis applications with Titan.
Introduction / Motivation

How do I use it? Alternatively, you can use the general-purpose OverView client to deploy Titan components ...
Project Scope

Data Structures
- Table
- Tree
- DAG
- Directed Graph
- Undirected Graph
- Sparse N-way Array
- Dense N-way Array

Database Drivers
- MySQL
- Postgres
- Oracle
- SQLite
- ODBC
- Netezza

MTGL Algorithms
- Community Finder
- ST Search
- CSG Search
- Temporal Search

BGL Graph Algorithms
- Breadth First Search
- Connected Components
- Biconnected Components
- Brandes Centrality

PBGL Integration

MATLAB Integration

Statistics Algorithms
- Descriptive
- Order
- Correlative
- Contingency

Linear-Time Graph Layouts
- GSpace
- Hierarchical
- Clustered
- Three tree-based variants

Multiple View Types
- Render (3D)
- Graph
- Hierarchical Graph
- Tree
- Treemap
- Georeferenced

Multiple Platforms / Languages
- Windows, Linux, OSX, HPC
- Write components in C++
- Use with C++, Python, TCL, Java
- Use as OverView “plugins”

Readers
- Dimacs
- DOT
- GXL
- Chaco
- XML
- Tulip
- DelimitedText
- FixedWidth
- ISI, RIS

Multidimensional Analysis
- TPP / PARAFAC
Prototype Application

Network Analysis and Cyber Defense: Using network packet captures to detect and track exfiltration events across political boundaries.

Network Grand Challenge

Demonstration of Prototype 1
Tutorial Outline

- Introduction and Motivation (15 minutes)
  - Project Background and Scope
  - Use Case: Cyber Defense
- VTK InfoVis Data Structures (30 minutes)
  - Data Structures
    - Tables
    - Trees
    - Graphs
  - Database Access
  - Table, Tree and Graph Readers
- Short Break
- Analysis Capabilities (50 minutes)
  - Graph Algorithms
  - Statistics
  - MATLAB interface
- Algebraic Methods (20 minutes)
- Building Applications (30 minutes)
  - Script Language Support
  - C++
  - Java
  - .Net/Com Interfaces
- OverView (30 minutes)
  - Application
  - Plugins
VTK Pipeline (Sidebar)

- Demand-driven
- Extensible, component design
- Shallow copy of data
Data Structures

- vtkDataObject
  - vtkArray
  - vtkGraph
    - vtkDirectedGraph
      - vtkDirectedAcyclicGraph
      - vtkTree
    - vtkMutableDirectedGraph
  - vtkTable
    - vtkMutableUndirectedGraph
    - vtkUndirectedGraph
      - vtkMutableUndirectedGraph
### vtkTable

<table>
<thead>
<tr>
<th>“ID”</th>
<th>“Name”</th>
<th>“Age”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bob</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>Ann</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>Sue</td>
<td>72</td>
</tr>
<tr>
<td>4</td>
<td>Bill</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Joe</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>Jill</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>Rick</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>Luis</td>
<td>68</td>
</tr>
</tbody>
</table>
Creating a vtkTable

```cpp
t = vtkTable::New();
col1 = vtkIntArray::New();
col1->SetName("ID");
col1->InsertNextValue(0);
col1->InsertNextValue(1);
t->AddColumn(col1);

col2 = vtkStringArray::New();
col2->SetName("Name");
col2->InsertNextValue("a");
col2->InsertNextValue("b");
t->AddColumn(col2);
```
vtkGraph and Subclasses

Points | VertexData | EdgeData
--- | --- | ---

Adjacency Lists

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Vertex ID | Edge ID
--- | ---
vtkMutableDirectedGraph* g = vtkMutableDirectedGraph::New();

vtkIntArray* vertId = vtkIntArray::New();
vertId->SetName("id");
g->GetVertexData()->AddArray(vertId);

for (vtkIdType v = 0; v < 10; ++v)
{
  g->AddVertex();
  vertId->InsertNextValue(v);
}

for (vtkIdType e = 0; e < 10; ++e)
{
  g->AddEdge(e, (e+1)%10);
}
Creating a Tree

```cpp
vtkMutableDirectedGraph* g = vtkMutableDirectedGraph::New();

vtkIdType root = g->AddVertex();
for (vtkIdType v = 0; v < 5; ++v)
{
    g->AddChild(root);
}

vtkTree* t = vtkTree::New(); t->ShallowCopy(g);
g->Delete();
```
vtkArray and Subclasses

Provides a flexible hierarchy of arbitrary-dimension arrays, including sparse and dense storage, efficient access, and support for custom array types.

There is SO much interesting stuff that we made a separate section. 😊

Using vtkArray with tensor decomposition methods such as PARAFAC:
Database Access In VTK

vtkSQLDatabase

vtkTable
,vtkTable
,vtkTable
,vtkTable

Database
#include <vtkSQLDatabase.h>
vtkSQLDatabase *db = vtkSQLDatabase::CreateFromURL("mysql://username@dbserver.domain.com/databasename");
bool openStatus = db->Open("mypassword");

OR

#include <vtkMySQLDatabase.h>
vtkMySQLDatabase *db = vtkMySQLDatabase::New();
db->SetUserName("username");
db->SetHostName("dbserver.domain.com");
db->SetDataBaseName("databasename");
db->Open("password");
vtkSQLQuery *query = db->GetQueryInstance();
query->SetQuery("SELECT name FROM employees WHERE salary > 100000");
Bool status = query->Execute();

while (query->NextRow())
{
    cout << query->DataValue(0).ToString() << " is making too much money, hire a new PhD."
}

query->Delete();
vtkRowQueryToTable *tableReader = vtkRowQueryToTable::New();
vtkSQLQuery *query = db->GetQueryInstance();

query->SetQuery("SELECT name, salary, age FROM employees");
tableReader->SetQuery(query);

tableReader->Update(); // will execute query and read the results
    // into a vtkTable

tableReader->Delete();
query->Delete();
Available Database Drivers

- **SQLite**
  - Public domain - ships with VTK

- **PostgreSQL**
  - Depends on libpq (part of the Postgres distribution)

- **MySQL**
  - Depends on libmysqlclient (part of the MySQL distribution)

- **ODBC**
  - Depends on having ODBC libraries installed
    - Unix (incl. Mac): iODBC, unixodbc
    - Windows: MS ODBC
    - Also requires vendor-specific driver for your particular database

- **Add your own! It’s simple.**
  - Subclass vtkSQLDatabase, vtkSQLQuery and implement abstract methods
  - Add optional support to CreateFromURL()
Using vtkDatabase and vtkRowQueryToTable to hit a database, pull data, and create graphs.

VTK/Examples/Infovis/Python/database.py
Graph Layout Strategies Example

Adding new graph layouts to Titan is a snap!

VTK/Examples/Infovis/Python/database.py
Table and Tree Readers

- **vtkDelimitedTextReader**
  - Creates a vtkTable from delimited text files, including CSV.

- **vtkISIReader**
  - Creates a vtkTable from files in the ISI bibliographic citation format.
  - Reference: [http://isibasic.com/help/helpprn.html#dialog_export_format](http://isibasic.com/help/helpprn.html#dialog_export_format)

- **vtkRISReader**
  - Creates a vtkTable from files in the RIS bibliographic citation format.

- **vtkFixedWidthTextReader**
  - Creates a vtkTable from text files with fixed-width fields.

- **vtkXMLTreeReader**
  - Creates a vtkTree using the structure of any XML file.
  - XML elements, text, and attributes are mapped to vertex attributes in the output graph.
Graph Readers and Sources

- **vtkRandomGraphSource**
  - Creates a graph containing a random collection of vertices and edges.

- **vtkSQLGraphReader**
  - Creates a vtkGraph from a pair of SQL vertex and edge queries.

- **vtkDIMACSGraphReader**
  - Creates a vtkGraph from files in DIMACS format.

- **vtkChacoGraphReader**
  - Creates a vtkGraph from files in Chaco format.

- **vtkTulipReader**
  - Creates a vtkGraph from files in Tulip format.

- **vtkGeoRandomGraphSource**
  - Creates a graph containing a random collection of geo-located vertices and edges.
Introduction and Motivation (15 minutes)
- Project Background and Scope
- Use Case: Cyber Defense

VTK InfoVis Data Structures (30 minutes)
- Data Structures
  - Tables
  - Trees
  - Graphs
- Database Access
- Table, Tree and Graph Readers

VTK InfoVis Components (45 minutes)
- Data Conversions
- Graph/Tree Layout Strategies
- Qt Model Adapters
- Views/Displays
- Selection
- Geographic Visualization

Analysis Capabilities (50 minutes)
- Graph Algorithms
- Statistics
- MATLAB interface

Algebraic Methods (20 minutes)

Building Applications (30 minutes)
- Script Language Support
  - C++
  - Java
  - .Net/Com Interfaces

OverView (30 minutes)
- Application
- Plugins
vtkTableToTree – Part II

<table>
<thead>
<tr>
<th>Author</th>
<th>Article</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a1</td>
<td>math</td>
</tr>
<tr>
<td>b</td>
<td>a1</td>
<td>math</td>
</tr>
<tr>
<td>c</td>
<td>a2</td>
<td>physics</td>
</tr>
<tr>
<td>d</td>
<td>a3</td>
<td>math</td>
</tr>
<tr>
<td>e</td>
<td>a4</td>
<td>physics</td>
</tr>
<tr>
<td>b</td>
<td>a4</td>
<td>physics</td>
</tr>
</tbody>
</table>

vtkTableToTree → vtkGroupLeafNodes → vtkGroupLeafNodes

Diagram:
- Root node
- Child nodes: a, b, c, d, e
- a has children m, a1, a1
- b has children p, a4
- c has children p, a2
- d has children m, a3
- e has children p, a4

Author topics:
- a: math
- b: math
- c: physics
- d: math
- e: physics

Diagram:
- Root
- a1: math
- a2: physics
- a3: math
- a4: physics

Diagram:
- vtkTableToTree
- vtkGroupLeafNodes
- vtkGroupLeafNodes
vtkTableToGraph – Part I

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>d</td>
</tr>
</tbody>
</table>

Diagram:

```
  a --- e --- c
  |     ^     |
  |     |     |
  d ------------- b
```
vtkTableToGraph – Part II

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Foo</td>
</tr>
<tr>
<td>b</td>
<td>Bar</td>
</tr>
<tr>
<td>c</td>
<td>Foo</td>
</tr>
<tr>
<td>d</td>
<td>Bar</td>
</tr>
<tr>
<td>e</td>
<td>Bar</td>
</tr>
<tr>
<td>b</td>
<td>Foo</td>
</tr>
</tbody>
</table>
### vtkTableToGraph – Part III

<table>
<thead>
<tr>
<th>Author</th>
<th>Article</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a1</td>
<td>math</td>
</tr>
<tr>
<td>b</td>
<td>a1</td>
<td>math</td>
</tr>
<tr>
<td>c</td>
<td>a2</td>
<td>physics</td>
</tr>
<tr>
<td>d</td>
<td>a3</td>
<td>math</td>
</tr>
<tr>
<td>e</td>
<td>a4</td>
<td>physics</td>
</tr>
<tr>
<td>b</td>
<td>a4</td>
<td>physics</td>
</tr>
</tbody>
</table>

- **Math**: A1, A3
- **Physics**: A2, A4

Diagram:
- Author: a, b, c, d, e
- Article: a1, a2, a3, a4
- Topic: Math, Physics
Table to Graph Example

Example that demonstrates the use of vtkTableToGraph filter.

VTK/Examples/Infovis/Python/database.py
Graph/Tree Layout Strategies

vtkDatabase
vtkQuery

vtkTable

vtkTableToGraph

vtkGraph

vtkGraphLayout
Layout Strategy

vtkGraph + Coordinates

Rest of Pipeline
Adding new graph layouts to Titan is a snap!

VTK/Examples/Infovis/Python/database.py
Making easy... easier...

Using `vtkSQLDatabaseGraphSource` which combines several classes to simplify pulling graphs from arbitrary databases.

VTK/Examples/Infovis/Python/database2.py
Qt Adapters

- **vtkQtAbstractModelAdapter**
  - Inherits from QAbstractItemModel, Qt’s generic item model for views
  - Provides common infrastructure for converting QModelIndex to VTK ids.
- **vtkQtTableModelAdapter**
  - Inherits from vtkQtAbstractModelAdapter
  - Adapts underlying vtkTable instance to a Qt model
- **vtkQtTreeModelAdapter**
  - Inherits from vtkQtAbstractModelAdapter
  - Adapts underlying vtkTree instance to a Qt model
- **QTableView, QTreeView**
  - Display a QAbstractItemModel
- **vtkQtTable/TreeView, vtkQtTable/TreeRepresentation**
  - Puts QTableView, QTreeView into VTK view/representation framework using the model adapter classes
  - Supports selection linking with other VTK views.
Qt Adapters C++ Example

Qt a reasonable model/view architecture for tables and trees
(*specifically shown are QTableView, QTreeView, QColumnView*).

Code “clips” from VTK/Examples/Infovis/Cxx/EasyView

```cpp
... this->XMLReader = vtkSmartPointer<vtkXMLTreeReader>::New();
this->TreeView = vtkSmartPointer<vtkQtTreeView>::New();

// Set widget for the tree view
this->TreeView->setItemView(this->ui->treeView);
...

// Create xml reader
this->XMLReader->setFileName(fileName.toAscii());
...

// Now hand off tree to the tree view
this->TreeView->setRepresentationFromInputConnection(
    this->XMLReader->GetOutputPort());
...```
Views

- **vtkView**: manages: “canvas”, interaction
- **vtkDataRepresentation**: manages: data, selection
  - uses input from **vtkDataObject**
  - uses selection from **vtkSelectionLink**

Specific views:
- **vtkQtChartView**: has a **QCanvas**
- **vtkQtItemView**: has a **QItemView**
- **vtkRenderView**: has a **vtkRenderWindow**

Sub-views of **vtkQtChartView**:
- **vtkQtTreeView**
- **vtkQtTableMapView**
- **vtkQtListView**

Sub-views of **vtkQtItemView**:
- **vtkGraphLayoutView**
- **vtkTreeMapView**
- **vtkGeoView**
- **vtkHierarchicalGraphView**
Views Tour

vtkGraphLayoutView

vtkGeoView

vtkHierarchicalGraphView

vtk TreeMapView

vtk QtTreeView

vtk QtTableView
Views Python Example

The VTK class hierarchy shown in a `vtkTreeMapView`, a `vtkGraphLayoutView` (with tree layout) and a `vtkHierarchicalGraphView`. The last view also pulls in a graph to show class inheritance relationships.

VTK/Examples/Infovis/Python/views.py
Linked Selection

- Extract selected data to highlight
- Map to local domain
- Pull from selection link
- Convert to shared selection type (pedigree IDs)
- Assign to selection link

`vtkView`

`vtkDataRepresentation`

`vtkSelectionLink`

`vtkSelection`

Pedigree ID
Index
Value
Frustum
Thresholds
This example demonstrates the use of `vtkSelectionLink` and `vtkSelectionSource`. Any vtk view can link it’s selection with any other view. `vtkSelections` are quite flexible and can be used in a variety of ways, here we select edges with high centrality.

VTK/Examples/Infovis/Python/selection.py
Geographic visualization

- Current features (in VTK now)
  - 3D vtkGeoView
  - Multi-resolution texture and geometry
  - Display placemarks with relationships (i.e. a geolocated graph)
  - Deep integration with other VTK views
    - Takes vtkDataObject input
    - Linked selection with other views
    - Easily embedded into larger applications

- Developing features
  - vtkGeoView2D
  - Multi-texturing overlay images with blending
  - More input sources
3D GeoView Python Example

Uses `vtkGeoView` and `vtkGeoRandomGraphSource`, linked with the same graph in a `vtkGraphLayoutView`.

VTK/Examples/Infovis/Python/geovis.py
GeoView Python Examples

Pulls data from the publicly available GTD (Global Terrorism Database) and uses vtkGeoView2D.

vtkSNL/Examples/Python/Infovis/gtd_geovis_2d.py
All projections from the open-source Proj.4 projection library are available to \texttt{vtkGeoView2D}.
Break Time!

Graph Algorithms, Statistics, and Algebraic Methods are next…
Tutorial Outline

- Introduction and Motivation (15 minutes)
  - Project Background and Scope
  - Use Case: Cyber Defense
- VTK InfoVis Data Structures (30 minutes)
  - Data Structures
    - Tables
    - Trees
    - Graphs
  - Database Access
  - Table, Tree and Graph Readers
- VTK InfoVis Components (45 minutes)
  - Data Conversions
  - Graph/Tree Layout Strategies
  - Qt Model Adapters
  - Views/Displays
  - Selection
  - Geographic Visualization

- Short Break

- Analysis Capabilities (50 minutes)
  - Graph Algorithms
  - Statistics
  - MATLAB interface

- Algebraic Methods (20 minutes)

- Building Applications (30 minutes)
  - Script Language Support
    - C++
    - Java
    - .Net/Com Interfaces

- OverView (30 minutes)
  - Application
  - Plugins
vtkBoostGraphAdapter.h implements the BGL graph concepts for vtkGraph.

vtkBoostGraphAdapter.h implements the BGL graph concepts for vtkGraph.

vtkBoostBreadthFirstSearch
vtkBoostBreadthFirstSearchTree
vtkBoostBiconnectedComponents
vtkBoostBrandesCentrality
vtkBoostConnectedComponents
vtkBoostKruskalMinimumSpanningTree
vtkBoostPrimMinimumSpanningTree

... add your own! 😊
BGL Python Examples

Running vtkBoostBreadthFirstSearch and coloring/labeling the vertices based on the distance from the seed point.

VTK/Examples/Infovis/Python/boost_bfs.py

Running vtkBoostBrandesCentrality and coloring/labeling the edges and vertices based on centrality.

VTK/Examples/Infovis/Python/boost_centrality.py
Running `vtkBoostBrandesCentrality` and then `vtkBoostKruskalMinimumSpanningTree` to compute a ‘maximal’ spanning tree on high centrality edges.

VTK/Examples/Infovis/Python/boost_mst.py

Running the same boost algorithms as above on a more complicated graph and then using `vtkExtractSelectedGraph` to send the extracted MST to another view.

VTK/Examples/Infovis/Python/boost_mst_extract_graph.py
Now showing how the original graph and its computed ‘Maximal’ spanning tree can both be sent to vtkHierarchicalGraphView. The MST is used to drive the hierarchy and layout, the original graph edges are ‘bundled’ by using the hierarchy as control points.

VTK/Examples/Infovis/Python/boost_mst_with_hgv.py
vtkPBGLGraphAdapter.h implements the PBGL graph concepts for a vtkGraph (with associated vtkPBGLDistributedGraphHelper).

\[
\text{vtkGraph} \cdot \text{SetDistributedHelper(PBGL)};
\]

vtkPipeline \rightarrow \text{Any PBGL Algorithm} \rightarrow \text{vtkPipeline}:

- vtkPBGLShortestPaths
- vtkPBGLRMATGraphSource
- vtkPBGLMinimumSpanningTree
- vtkPBGLGraphSQLReader
- vtkPBGLConnectedComponents
- vtkPBGLVertexColoring
- vtkPBGLBreadthFirstSearch
- vtkPBGLRandomGraphSource
Currently in the “Hello World” stage:
Running a BFS on a random graph containing 50M vertices and 500M edges on 80 nodes.
Performing BFS on a random graph with 100K vertices and 100K edges in parallel, collecting the graph and viewing it in graph layout view.

VTK/Examples/Infovis/Cxx/ParallelBFS.cxx
PBGL Examples

The Enron email corpus graph, containing 75K email accounts and 2M email communications.

VTK/Parallel/Testing/Cxx/TestPBGLGraphSQLReader.cxx

Using a parallel pipeline to extract summary information of how people with different job titles interact.
Multi-Threaded Graph Library (MTGL) Adapter

vtkMTGLGraphAdapter.h implements the MTGL graph concepts for vtkGraph.

vtkPipeline \rightarrow \text{Any MTGL Algorithm} \rightarrow \text{vtkPipeline}

vtkMTGLCommunityFinder
vtkMTGLHierarchicalCommunityFinder
vtkMTGLSearchEdgeTime
vtkMTGLSearchSSSPDeltastepping
vtkMTGLSelectionFilterCSG
vtkMTGLSelectionFilterST

… list is growing…

Cray XMT: Massively multithreaded platform, great for graph algorithms. 😊
Running vtkMTGLCommunityFinding and coloring/labeling the vertices based on the community.

vtkSNL/Examples/Python/Infovis/mtgl_community.py

Running vtkBoostTemporalSearchFwd and coloring/labeling the edges and vertices based on earliest ‘reachability’.

vtkSNL/Examples/Python/Infovis/temporal_search_test.py
The statistics engines can be run in “Learn” (calculate model statistics from a data set) and/or “Assess” (given model statistics – from the same or another data set -- mark each datum) options.

Univariate statistics:
  Descriptive statistics:
  “Learn”: minimum, maximum, mean, variance, skewness, kurtosis (various estimators)
  “Assess”: mark with number or relative deviations (specified means and deviation)
  Order statistics:
  “Learn”: arbitrary quartiles (in particular, "5-point" statistics (quartiles) and box plots, deciles, percentiles, etc.), histogram.
  “Assess”: mark with deviation from specified box

Bivariate statistics:
  Correlative statistics:
  “Learn”: bivariate mean, variance/covariance matrix, Pearson regression
  “Assess”: mark with relative probability w.r.t. to specified means and covariance matrix
  Contingency statistics:
  “Learn”: contingency table and joint probabilities
  “Assess”: mark with conditional PDF values (X|Y and Y|X) and with joint PDF value.
  Also, calculate information entropies to decide which conditioning is the most informative.

Contingency statistics can be performed on any (categorical) type of variables; the other engines take numerical variables as inputs.
Descriptive, Order and Correlative Statistics

BTW: Linux Screenshot
Awesomeness is free 😊
Running contingency statistics on network transfers illuminates protocols going over non-standard network ports.

VTK/Examples/Infovis/Python/contingency_port_protocol.py
Demonstrates a conditional probability calculation $p(\text{port} \mid \text{protocol})$. 
MATLAB® Titan Toolbox *(work in progress)*

- The MATLAB® Titan Toolbox allows Titan functionality to be accessed from the MATLAB® command line.

```matlab
>> g = graph;
>> for i=0:99 g.addvertex; end
>> for i=0:99 g.addedge(i,mod(i+1, 100)); g.addedge(i,mod(i+3, 100)); end

>> g.view;
>> spy(g.tomatrix);
```
MATLAB® Interface (work in progress)

- **vtkMatlabEngineFilter**
  - Interact with a MATLAB® engine process
  - Execute MATLAB® commands from VTK
  - Push VTK pipeline data to MATLAB®
  - Pull MATLAB® data into the VTK pipeline

- **vtkMatlabProgrammableFilter**
  - Call a compiled MATLAB® M-file function from VTK
  - Filter handles the necessary data conversions
  - Allows fast prototyping through M-file scripting
  - Requires MATLAB® MCC compiler product
MATLAB® Examples (work in progress)

Uses algorithm by Volchenkov and Blanchard (An algorithm generating random graphs with power law degree distributions, Physica A, Volume 315, Number 3, 1 December 2002, pp. 677-690) to produce four random graphs with a power law degree distribution on graph vertex out-degree.

A log plot of vertex out-degree (with blue circles) from the four random graphs shows the expected linear relationship between vertex fraction and outgoing vertex degree.

vtkSNL/Examples/MatlabTitanToolbox/powerlawgraph.m
Generate a random power law graph with 50 vertices.

Perform a Breadth First Search on the graph and display the resulting search tree.
Tutorial Outline

- Introduction and Motivation (15 minutes)
  - Project Background and Scope
  - Use Case: Cyber Defense

- VTK InfoVis Data Structures (30 minutes)
  - Data Structures
    - Tables
    - Trees
    - Graphs
  - Database Access
  - Table, Tree and Graph Readers

- VTK InfoVis Components (45 minutes)
  - Data Conversions
  - Graph/Tree Layout Strategies
  - Qt Model Adapters
  - Views/Displays
  - Selection
  - Geographic Visualization

- Short Break

- Analysis Capabilities (50 minutes)
  - Graph Algorithms
  - Statistics
  - MATLAB interface

- Algebraic Methods (20 minutes)

- Building Applications (30 minutes)
  - Script Language Support
    - C++
    - Java
    - .Net/Com Interfaces

- OverView (30 minutes)
  - Application
  - Plugins
Data Structures for Algebraic Methods

• What We Wanted To Do:
  • Integrate algebraic methods into the pipeline, including tensors.
    • Note: mathematical tensors, not tensor-fields!
    • SVD / LSA
    • PARAFAC / TUCKER / DEDICOM
  • Provide dense and sparse storage.
  • Efficiently represent graphs as adjacency matrices.
  • Possibly provide a future replacement for vtkAbstractArray and friends.

• What We Didn’t Want To Do:
  • Invent another linear algebra package.

• Caveats:
  • Currently, the array classes aren’t wrapped / aren’t accessible from scripting languages (because they’re class templates).
  • You can still use array algorithms for scripting.
• **vtkArray**: Generic arrays of unknown type.
• **vtkTypedArray**: Generic arrays of known type.
• **vtkDenseArray**: Contiguous-storage arrays of known type.
• **vtkSparseArray**: Coordinate-storage arrays of known type.
• **Custom Arrays**: The array API allows for "custom" arrays - imagine compressed-row storage, upper / lower diagonal storage, etc.
// Creating a dense array of 10 integers:
vtkDenseArray<vtkIdType>* array = vtkDenseArray<vtkIdType>::New();
array->Resize(10);
array->Fill(1);

// Creating a dense 10 x 20 matrix:
vtkDenseArray<double>* matrix = vtkDenseArray<double>::New();
matrix->Resize(10, 20);
matrix->Fill(0.0);

// Creating a sparse 10 x 20 x 30 x 40 tensor:
vtkArrayExtents extents;
Extents.SetDimensions(4);
extents[0] = 10;
extents[1] = 20;
extents[2] = 30;
extents[3] = 40;
vtkSparseArray<vtkIdType>* tensor = vtkSparseArray<vtkIdType>::New();
tensor->Resize(extents);
// Assign array value [5]:
array->SetValue(5, 42);

// Assign matrix value [4, 3]:
matrix->SetValue(4, 3, 1970);

// Assign tensor value [3, 7, 1, 2]:
vtkArrayCoordinates coordinates;
coordinates.SetDimensions(4);
coordinates[0] = 3;
coordinates[1] = 7;
coordinates[2] = 1;
coordinates[3] = 2;
tensor->SetValue(coordinates, 38);

// Access array value [5]:
array->GetValue(5);

// Access matrix value [4, 3]:
matrix->GetValue(4, 3);

// Access tensor value [3, 7, 1, 2]:
tensor->GetValue(coordinates);
Populating Sparse Arrays

- `vtkSparseArray<T>::AddValue()` appends unchecked values to a sparse array.
  - Note: it's up to the caller to avoid calling `AddValue()` more than once with the same set of coordinates!

- `vtkSparseArray<T>::ResizeToContents()` updates the array extents to match the current array contents.
  - Handy once you're done calling `AddValue()`.

- Example: `VTK/Examples/Array/Cxx/IdentityMatrix.cxx`
Array Value Iteration

- Iteration provides unordered access in O(1) time:
  - Eliminates the cost of lookups when getting / setting sparse array values.
  - Only visits non-null values in sparse arrays.
  - Provides a consistent interface across dense and sparse arrays.
  - Is completely dimension-independent.
- Example: VTK/Examples/Array/Cxx/ArrayIteration.cxx

```cpp
// Set the n-th value in an array:
void vtkTypedArray<T>::SetValueN(vtkIdType n, const T& value);

// Return the n-th value in an array:
const T& vtkTypedArray<T>::GetValueN(vtkIdType n);

// Return the coordinates for the n-th value in an array:
void vtkArray::GetCoordinatesN(vtkIdType n, vtkArrayCoordinates& coordinates);
```
Array Sources

- **vtkDiagonalMatrixSource**
  - Produces sparse or dense matrices of arbitrary size, with user-assigned values for the diagonal, superdiagonal, and subdiagonal, e.g:

- **vtkBoostRandomSparseArraySource**
  - Produces sparse matrices with arbitrary size and number of dimensions.
  - Separate controls for random values and random sparsity pattern.

- **vtkTableToSparseArray**
  - Converts a vtkTable containing coordinates & values into a sparse array of arbitrary dimensions.
Array Algorithms

- **vtkAdjacencyMatrixToEdgeTable**
  - Converts a dense matrix into a vtkTable suitable for use with vtkTableToGraph.
  - Dimension labels in the input matrix are mapped to column names in the output table.

- **vtkArrayVectorNorm**
  - Computes an L-norm for each column-vector in a sparse double matrix.

- **vtkCosineSimilarity**
  - Treats each row or column in a matrix as a vector, and computes the dot-product similarity between each pair of vectors, producing a vtkTable suitable for use with vtkTableToGraph as output.
More Array Algorithms

- **vtkBoostLogWeighting**
  - Replaces each value $p$ in an array with the natural logarithm of $p+1$.
  - Good example of a filter that works with any array, containing any number of dimensions.

- **vtkMatricizeArray**
  - Converts sparse double arrays of arbitrary dimension to sparse matrices.
  - For example, an $i \times j \times k$ tensor can be converted into an $i \times jk$, $j \times ik$, or $ij \times k$ matrix.

- **vtkNormalizeMatrixVectors**
  - Normalizes either row vectors or column vectors in a matrix.
  - Good example of a filter that works efficiently with both sparse and dense input matrices.
  - Good example of a filter that works with either row or column vectors.

- **vtkTransposeMatrix**
  - Does what it says …
// Create a matrix with non-zero super- and sub-diagonals
vtkDiagonalMatrixSource* source =
    vtkDiagonalMatrixSource::New();
source->SetExtents(10);
source->SetDiagonal(0);
source->SetSuperDiagonal(1);
source->SetSubDiagonal(2);
// Convert an adjacency matrix to an edge table
vtkAdjacencyMatrixToEdgeTable* edges =
    vtkAdjacencyMatrixToEdgeTable::New();
edges->SetInputConnection(source->GetOutputPort());
// Convert the edge table to a graph
vtkTableToGraph* graph = vtkTableToGraph::New();
graph->SetInputConnection(edges->GetOutputPort());
graph->AddLinkVertex("rows", "eid", false);
graph->AddLinkVertex("columns", "eid", false);
graph->AddLinkEdge("rows", "columns");
Tensor Reduction

- vtkTableToArray
- vtkTppParafac
- vtkExtractFactoredArray
- vtkExtractFactoredArray
- vtkExtractFactoredArray
Term-term Similarity Graphs

vtkExtractFactoredArray → vtkLogWeighting

vtkNormalizeMatrixVectors → vtkCosineSimilarity → vtkTableToGraph → vtkGraphView
Concept-term Similarity Graphs

vtkExtractFactoredArray

vtkAdjacencyMatrixToEdgeTable → vtkTableToGraph → vtkGraphView
Concept-Activity Over Time

vtkExtractFactoredArray → vtkArrayToTable → vtkQtChartView
Tutorial Outline

• Introduction and Motivation (15 minutes)
  • Project Background and Scope
  • Use Case: Cyber Defense

• VTK InfoVis Data Structures (30 minutes)
  • Data Structures
    • Tables
    • Trees
    • Graphs
  • Database Access
  • Table, Tree and Graph Readers

• VTK InfoVis Components (45 minutes)
  • Data Conversions
  • Graph/Tree Layout Strategies
  • Qt Model Adapters
  • Views/Displays
  • Selection
  • Geographic Visualization

• Short Break

• Analysis Capabilities (50 minutes)
  • Graph Algorithms
  • Statistics
  • MATLAB interface

• Algebraic Methods (20 minutes)
  • Building Applications (30 minutes)
    • Script Language Support
      • C++
      • Java
      • .Net/Com Interfaces

  • OverView (30 minutes)
    • Application
    • Plugins
package require vtk

tkRandomGraphSource src

tkGraphLayoutView view
view AddRepresentationFromInputConnection
    [src GetOutputPort]

tkRenderWindow window
view SetupRenderWindowWindow window
    [window GetInteractor] Start

wm withdraw .
from vtk import *

source = vtkRandomGraphSource()

view = vtkGraphLayoutView()
view.AddRepresentationFromInputConnection(source.GetOutputPort())

window = vtkRenderWindow()
window.SetSize(600, 600)
view.SetupRenderWindow(window)
window.GetInteractor().Start()
Qt has model/view classes for tables and trees (specifically shown are \textit{QTableView}, \textit{QTreeView}, \textit{QColumnView}).

Code “clips” from VTK/Examples/Infovis/Cxx/EasyView

...  
this->XMLReader = \texttt{vtkSmartPointer<vtkXMLTreeReader>::New();}
this->TreeView = \texttt{vtkSmartPointer<vtkQtTreeView>::New();}

\hspace{1cm} // Set widget for the tree view
this->TreeView->setItemView(this->ui->treeView);
...

\hspace{1cm} // Create xml reader
this->XMLReader->setFileName( fileName.toAscii() );
...

\hspace{1cm} // Now hand off tree to the tree view
this->TreeView->setRepresentationFromInputConnection(  
this->XMLReader->GetOutputPort());
...
VTK/Examples/Infovis/Java/Focus.java

Display all data, along with focused selection using breadth first search. Uses Java Swing components.
.Net Example Application: Wikipedia Browsing (C#)

http://www.kitware.com/products/activiz.html

Application for browsing wikipedia connectivity using C# wrappers. Uses Windows GUI components.
using Kitware.VTK;

private void addLinks(Kitware.VTK.vtkMutableDirectedGraph g, string lookupValue, int hops)
{
    // Fetch XML from Wikipedia
    System.Net.HttpWebRequest webRequest = ...
    ... // Parse XML to get links to other articles
    // If the new vertex is not already there add it
    int v = label.LookupValue(substring);
    if (v < 0)
    {
        v = g.AddVertex();
        label.InsertNextValue(substring);
    }
}
Olympic Medals (Visual Basic embedded in Excel)

vtkRenderWindow COM ActiveX control shows connections between Countries, Athletes, and Events at Beijing 2008.
Private Sub CommandButton1_Click()

    ' Create a vtkTable by looking up Excel cells
    ...

    ' Use vtkTableToGraph to make a graph
    ...

    Set win = vtkRenderWindowControl1.GetRenderWindow

    Set v = New vtkGraphLayoutView
    v.AddRepresentationFromInput cat.GetOutput
    v.SetupRenderWindow win
    v.SetLayoutStrategyToSimple2D
    v.SetVertexLabelArrayName "value"
    v.VertexLabelVisibilityOn
    v.SetVertexColorArrayName "category"
    v.ColorVerticesOn
    v.Update

    End Sub
Tutorial Outline

• Introduction and Motivation (15 minutes)
  • Project Background and Scope
  • Use Case: Cyber Defense

• VTK InfoVis Data Structures (30 minutes)
  • Data Structures
    • Tables
    • Trees
    • Graphs
  • Database Access
  • Table, Tree and Graph Readers

• VTK InfoVis Components (45 minutes)
  • Data Conversions
  • Graph/Tree Layout Strategies
  • Qt Model Adapters
  • Views/Displays
  • Selection
  • Geographic Visualization

• Short Break

• Analysis Capabilities (50 minutes)
  • Graph Algorithms
  • Statistics
  • MATLAB interface

• Algebraic Methods (20 minutes)

• Building Applications (30 minutes)
  • Script Language Support
    • C++
    • Java
    • .Net/Com Interfaces

• OverView (30 minutes)
  • Application
  • Plugins
OverView - A general-purpose informatics tool
OverView - based on the ParaView architecture
OverView Plugins

- Extends the collection of readers, writers, and filters at runtime.
- Shared libraries containing new filters are dynamically-linked into the working-set at runtime.
- Plugins can be loaded automatically at startup from known locations, locations specified via environment variable (PV_PLUGIN_PATH) or manually loaded via the plugin manager GUI:
Plugin Types - Readers & Writers
Plugin Types - General Filters
Plugin Types - Custom Toolbars

Useful for automating setup of a complex pipeline, an arbitrary view configuration, etc.
Plugin Types - Custom Panels

Provides a filter-specific user interface panel - useful with complex filters where the auto-generated GUI is insufficient.
Plugin Types - Custom Views
Plugin Technical Challenges

Node 0
- vtkParallelReader
- vtkFooFilter
- vtkBarFilter
- ...
- vtkParallelRenderStuff

Node 1
- vtkParallelReader
- vtkFooFilter
- vtkBarFilter
- ...
- vtkParallelRenderStuff

Node 2
- vtkParallelReader
- vtkFooFilter
- vtkBarFilter
- ...
- vtkParallelRenderStuff

Node N
- vtkParallelReader
- vtkFooFilter
- vtkBarFilter
- ...
- vtkParallelRenderStuff

Client
"Server Manager XML" is used to "wrap" a Titan filter so it can be used in OverView:

```xml
<ServerManagerConfiguration>
  <ProxyGroup name="filters">
    <SourceProxy name="FooFilter" class="vtkFooFilter">
      <InputProperty name="Input" ... />
      <IntVectorProperty name="FooCount" ... />
    </SourceProxy>
    <!-- More proxies in this group ... -->
  </ProxyGroup>
  <!-- More groups in this plugin ... -->
</ServerManagerConfiguration>
```

- The XML is linked into the plugin binary, and parsed by OverView when the plugin is loaded.
- The XML is used to auto-generate a graphical user interface for the plugin.
<ServerManagerConfiguration>
  <ProxyGroup name="sources">
    <SourceProxy name="TulipReader" class="vtkTulipReader">
      <StringVectorProperty name="FileName"
        command="SetFileName" number_of_elements="1">
        <FileListDomain name="files"/>
      </StringVectorProperty>
      <Hints>
        <View type="ClientGraphView"/>
      </Hints>
    </SourceProxy>
  </ProxyGroup>
</ServerManagerConfiguration>
Useful Plugin References

- "Advanced ParaView Visualization" Tutorial, tomorrow!

- General Plugin Information
  - [http://paraview.org/Wiki/Plugin_HowTo](http://paraview.org/Wiki/Plugin_HowTo)

- Server Manager XML
  - The ParaView/Servers/ServerManager/Resources/ directory.

- Sample Plugins
  - The ParaView/Plugins directory.
  - The ParaView/Examples/Plugins directory.
Interested in Using?
- Kitware Wiki: www.kitware.com/InfovisWiki

Source code: Download the VTK repository (instructions at www.vtk.org).

Questions/Issues: vtkusers@vtk.org

Interested in Contributing?
We are actively pursuing collaborators to join Sandia, Kitware and Indiana University in our efforts to grow and refine the capabilities.

Contacts:
- Brian Wylie (bnwylie@sandia.gov)
- Timothy Shead (tshead@sandia.gov)
- Jeff Baumes (jeff.baumes@kitware.com)
End

Questions/Comments?
Qt Interface

QTreeView

QVTKWidget

QTableView
Motivation: Project Goals

**Unified Toolkit:** Scientific Visualization and Information Visualization together at last!

**Scalable Toolkit:** Sandia’s use of VTK/ParaView has provided scalability on some of the world’s largest simulation results (*Billions of cells/Terabytes on disk*).

**Flexible Toolkit:** Component based pipeline architecture provides a development model that allows expansion, agility and domain specific application construction.
The Titan toolkit can be used by any ‘front-end’ GUI (TCL/TK, Java, Python and Qt).

Specific adapters have to be written for vtkTree and vtkTable that implement the `QtAbstractItemModel` interface.

```
vtkQtTreeItemModelAdapter vtkQtTableItemModelAdapter
```