VisKo: Supporting Declarative Visualization Requests

Nicholas Del Rio
Outline

- Visualization Requests and Pipelines
- Visualization Query Language
- Query Answering
- Query Language Evaluation
I want to visualize my gridded time data as a set of 35 isosurfaces and rotated 45 degrees on Z axis.
I want to visualize my gridded time data as a set of 35 isosurfaces and rotated 45 degrees on Z axis.

Visualization packages require procedural type specifications.
Visualization Pipelines

• Toolkits require that users compose pipelines

```c++
vtkImageReader rdr = new vtkImageReader();
rdr.SetFileName(inputDatasetFilePath);
rdr.SetDataScalarTypeToUnsignedShort();
rdr.SetDataByteOrderToLittleEndian();
rdr.SetFileDimensionality(3);
rdr.SetDataOrigin(0,0,0);
rdr.SetDataSpacing(1,1,1);
rdr.SetDataExtent(0,230,0,25,0,68);
rdr.SetNumberOfScalarComponents(1);
rdr.FileLowerLeftOn();
rdr.Update();

vtkContourFilter contours = new vtkContourFilter();
contours.SetInput(rdr.GetOutput());
contours.GenerateValues(35,0.0,9000.0);

vtkPolyDataMapper contMapper = new vtkPolyDataMapper();
contMapper.SetInput(contours.GetOutput());
contMapper.SetScalarRange(0.0,9000.0);
contActor.RotateX(105);

vtkRenderer ren1 = new vtkRenderer();
ren1.AddActor(contActor);
ren1.AddActor2D(outlineActor);
ren1.SetBackground(1,1,1);

vtkRenderWindow renWin = new vtkRenderWindow();
renWin.SetOffScreenRendering(1);
renWin.AddRenderer(ren1);
renWin.SetSize(300,300);
renWin.Render();

vtkJPEGWriter img = new vtkJPEGWriter();
img.SetInputConnection(renWin.GetOutputPort());
img.SetFileName(outputDatasetFilePath);
img.SetQuality(100);
```
Goal

• Enable scientists to declaratively request for toolkit-backed visualizations

• Objectives:
  • define a language that can be used to request for visualizations
  • develop an approach for translating requests in our language into toolkit-backed pipelines
Approach

SELECT isosurfaces IN-VIEWER
FROM http://somedata.3d

Mimics human reasoning when constructing pipelines
Visualization Query

- A *visualization query* is a visualization request in a machine readable form.

```
<table>
<thead>
<tr>
<th>View &amp; Viewer</th>
<th>SELECT isosurfaces IN-VIEWER firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data URL, Format &amp; Type</td>
<td>FROM <a href="http://trust.utep.edu/HolesCode/time.3d">http://trust.utep.edu/HolesCode/time.3d</a></td>
</tr>
<tr>
<td></td>
<td>FORMAT binaryFloatArray</td>
</tr>
<tr>
<td></td>
<td>TYPE griddedTime</td>
</tr>
<tr>
<td>Parameter Bindings</td>
<td>WHERE zRotation = 45 AND</td>
</tr>
<tr>
<td></td>
<td>numContours = 35</td>
</tr>
</tbody>
</table>
```
Challenges

1. Modeling the visualization pipeline space
   • look to existing efforts (e.g., Data State, Data Flow)
   • leverage Semantic Web

2. Verifying that our query language is easier to use than pipeline based languages
   • perform user study
Past Visualization Models

• **Data State Visualization Model (Chi 1998)**
  - Models different states of data as they are transformed from view to value
  - In-state vs. transformers
  - No further refinement of in-state or transformer (e.g., mapper or viewer)

• **Visualization Ontology (Brodlie 2004)**
  - Models tasks, pipelines, at very high level (e.g., techniques, data)
  - No use of format
  - No classification of operators
  - No rules to describe how to compose tasks or services
Our Pipeline Model

• Our 3-level semantic visualization model:
  • expands on data state model
  • defines a hierarchy of toolkit operators (e.g., viewer, transformer, mapper)
  • defines a set of rules that describes how operators can be chained together

**Visualization**

- **Visko-Views** (views and properties)
- **Visko-Operator** (operator function + composition rules)
- **Visko-Service** (services, parameters, and types)

**Service Composition**

- Built on Earth Science Information Partners (ESIP) Data types
- Separation of conceptual description from implementation (Brodlie 2004)
Automated Pipeline Composition

We can compose pipelines using the information from both queries and our model.
Knowledge about Toolkits

Knowledge about toolkit operators

Dataset

Information from Query

Binary Float Array

hasFormat

vtk Poly Data

vtk Image Data

operatesOn

transformsTo

vtk Contour Filter

mapsTo

HasView

Isosurfaces view
The Knowledge is Linked

Both query and operator reference “isosurfaces”

Information from Query

Knowledge about toolkit operators
Can Be Transformed To?

Dataset Information from Query

Binary Float Array

hasView

Isosurfaces view

hasView

mapsTo

vtk Contour Filter

vtk Poly Data

operatesOn

transformsTo

vtk Image Data

hasFormat

Contour Filter

mapsTo

vtk Poly Data

Knowledge about toolkit operators
Backtrack Proofs to Identify Pipelines

1. Analyze proofs of \( \text{canBeTransformedTo} \)
2. Identify a transformation path from one format to another
3. Query for operators that output each format

A visualization

Pipeline

\[
\text{CBTTo}(a,b) \rightarrow \text{canBeTransformedTo}(a, b) \rightarrow \text{CBTTo}(fmt1, fmt2, fmt3) \rightarrow \text{CTTo}(fmt2, fmt3) \rightarrow \text{CBTTo}(fmt1, fmt3, fmt4) \rightarrow \text{CTTo}(fmt3, fmt4) \rightarrow \text{CBTTo}(fmt1, fmt4)
\]

\[
\text{fmt1} \rightarrow \text{fmt2} \rightarrow \text{fmt3} \rightarrow \text{fmt4}
\]
Evaluating Our Query Language

• Compare our query language to pipeline languages
  • Is our query language more readable than pipeline code
  • Is our query language more writeable than pipeline code

```java
vtkImageReader rdr = new vtkImageReader();
rdr.SetFileName(inputDatasetFilePath);
rdr.SetDataScalarTypeToUnsignedShort();
rdr.SetDataByteOrderToLittleEndian();
rdr.SetFileDimensionality(3);
rdr.SetDataOrigin(0,0,0);
rdr.SetDataSpacing(1,1,1);
rdr.SetDataExtent(0,230,0,25,0,68);
rdr.SetNumberOfScalarComponents(1);
rdr.FileLowerLeftOn();
rdr.Update();

vtkContourFilter contours = new vtkContourFilter();
contours.SetInput(rdr.GetOutput());
contours.GenerateValues(35,0.0,9000.0);

vtkPolyDataMapper contMapper = new vtkPolyDataMapper();
contMapper.SetInput(contours.GetOutput());
contMapper.SetScalarRange(0.0,9000.0);
```

```sql
SELECT visko:isosurfaces IN-VIEWER visko:firefox
FROM http://rio.cs.utep.edu/HolesCodeFullPML/vel.3d
  FORMAT formats:BINARYFLOATARRAY
  TYPE types:d2
WHERE params:xRotation = 104     AND
     params:contourMin = 0
     AND
     params:contourMin = 9000.0
```
Experiment Trial Types

Reading (Use 1)

Given
Data Description
Pipeline
Visualization Set

Required
Trial Type 1
identify

Writing (Use 2)

Given
Data Description
Visualization
Operator List

Required
Trial Type 2
compose
Pipeline

Trial Type 3

Given
Data Description
Query
Visualization Set

Required
Trial Type 4
identify
compose
Visualization Knowledge Base
Query

Pipeline (Condition 2)

Query (Condition 1)
Conclusions

• VisKo queries serve as a declarative means for generating visualizations

• VisKo *knows* how to translate queries into pipelines

• A user study is being conducted to evaluate the VisKo query language
  • Readable?
  • Writable?