VTK: The Visualization Toolkit

Part II: Visualization Model

Han-Wei Shen
The Ohio State University
VTK Visualization Model

• Graphics and Visualization Model
  – Graphics model: rendering
  – Visualization model: transformation and representation

Data Flow System: Pipeline execution
The Role of Visualization Model

• Converting data from its original form into graphical form
• Deal with the issues of transformation and representation
  • Representation: internal data structures
  • Transformation: data to graphics

Example: stock price display

Internal (computational) representation:

Arrays of stock prices
Price[i] = 17100, 16950, 17073, 17050,...
Time[i] = 10, 10:30, 11, 11:30, 12, ...

Graphical Representation: x-y plot
3D Plot Example

Quadric function:

\[ \mathbf{f(x,y,z)} = a_0 x + a_1 y + a_2 z + a_3 xy + a_4 yz + a_5 xz + a_6 x^2 + a_7 y^2 + a_8 z^2 \]
3D Plot Example

Quadric function:

\[ f(x, y, z) = a_0 x + a_1 y + a_2 z + a_3 xy + a_4 yz + a_5 xz + a_6 x^2 + a_7 y^2 + a_8 z^2 \]

- Evaluate the function at a pre-determined x, y, z samples
3D Plot Example

Quadric function:

\[ f(x, y, z) = a_0 x + a_1 y + a_2 z + a_3 xy + a_4 yz + a_5 xz + a_6 x^2 + a_7 y^2 + a_8 z^2 \]

- Evaluate the function at a pre-determined \(x, y, z\) samples
- Store the function values into a 3D array

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Quadric function:

\[ f(x,y,z) = a_0 x + a_1 y + a_2 z + a_3 xy + a_4 yz + a_5 xz + a_6 x^2 + a_7 y^2 + a_8 z^2 \]

- Evaluate the function at a pre-determined \(x,y,z\) samples
- Store the function values into a 3D array
- Generate a 3D surface corresponding to \(f(x,y,z) = c\) (this is called an isosurface)
3D Plot Example

Quadric function:

\[ f(x,y,z) = a_0 x + a_1 y + a_2 z + a_3 xy + a_4 yz + a_5 xz + a_6 x^2 + a_7 y^2 + a_8 z^2 \]

- Evaluate the function at a pre-determined \( x, y, z \) samples
- Store the function values into a 3D array
- Generate a 3D surface corresponding to \( f(x,y,z) = c \) (this is called an isosurface)
- Display the surface

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>( Z )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Transformation (visualization)

Find all \((x,y,z)\)

\( f(x,y,z) = c \)
VTK Visualization Model

How is a visualization process described in VTK?

Visualization pipeline: It describes the flow of data from its original form to the final image output

Visualization Pipeline =
  Function Model + Object Model

• Function model: Transformation
  – steps to create visualization (how data flow through the system)
• Object model: objects that participate in the function model
  – processes and internal data structures
VTK Function Model

- Function model is to Illustrate the steps to create visualization.

```
Sample F(x,y,z)

Point Array

Extract Surface

Polygon

Display
```

- : process
- : data
- : data flow direction
VTK Object Model

Describe the objects that participate in the function model (visualization pipeline)
VTK Object Model

Describe the objects that participate in the functional model (visualization pipeline)

- Process Object: Operates on (transform) the data objects
  - Example: generating surfaces from $F(x,y,z)$
VTK Object Model

Describe which objects operate on the functional model (visualization pipeline)

• Process Object: Operates on (transform) the data objects
  - Example: generating surfaces from $F(x,y,z)$

• Data Object: Represent *information* and *methods* to create, access, and delete the information
  - Example: the 3D point array $F(x,y,z)$
Process Objects

• Source: Interface to external data sources or generate data procedurally

Visualization model

Source ➔ Filter ➔ Mapper

Graphics model

Actor ➔ Renderer
Process Objects

- **Source:** Interface to external data sources or generate data procedurally
- **Filter:** Transform the data input
Process Objects

- **Source**: Interface to external data sources or generate data procedurally
- **Filter**: Transform the data input
- **Mapper**: The sink of the functional model, interface with the graphics model
Process Objects (cont’d)

Sample $F(x,y,z)$ → Source

Point Array → Filter

Extract Surface → Mapper + Graphics Model Objects

Polygon → Display
Data Objects

Data Object: Dataset

- Structures: how the information is organized
  - Topology
  - Geometry

- Attributes: store the information we want
to visualize. e.g. function values
Data Objects: structures

• Topology:
  - Invariant under geometric transformation (rotation, translation, scaling etc)
  - Topological structures: Cells

• Geometry:
  - The instantiation of the topology
  - Geometric structures: cells with positions in 3D space
VTK Cell Types

(a) vertex
(b) Polyvertex
(c) line
(d) polyline
(e) triangle

(e) Quadrilateral
(e) Polygon
(f) Tetrahedron
(f) Hexahedron

And more
VTK Data Attributes

• The information stored at each corner of the cell
  – Scalars: temperature, pressure, etc
  – Vector: velocities
  – Normals: surface directions
  – Texture coordinates: graphics specific
  – Tensors: matrices
VTK Datasets

A Collection of structures and attributes

Structured Grid  Rectilinear Grid  Unstructured Points

Structured Grid  Polygonal Grid  Unstructured Grid
Pipeline Connections

• Type-Checking: the type of input data and the expected input to a filter should match, which can be checked at compile, link, or run time

• Multiplicity: a filter can have one or multiple output, and each output can have multiple fan-out, that is, connected to one or multiple downstream processes
Pipeline Execution

• A visualization pipeline is often executed multiple times, when the algorithm parameters are changed for example.

• How a visualization pipeline is executed is very much related to the performance

• A process object should re-execute only if changes occur to its input
  – Demand-driven approach: when the output of the process is needed
  – Event-driven approach: when changes to the process or to the input data
Pipeline Synchronization

- Pipeline synchronization – synchronization is needed between the processes in the visualization pipeline
  - *Explicit* execution: a centralized executive is used to track changes and coordinate network execution
  - *Implicit* execution:
    - each process propagate the execution request to its upstream processes, until the request reaches the source of the pipeline
    - The source process then examine its input and output to decide whether it should execute
    - Then recursively push the command for checking down to the pipeline

VTK uses this