

More VTK Details

Typical Graphics Interface Hierarchy

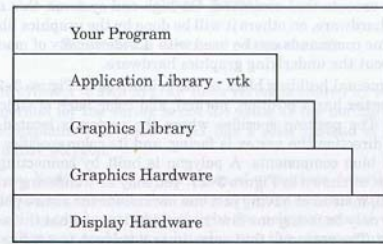
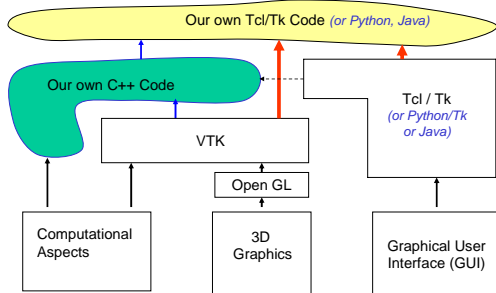


Figure 3-19 Typical graphics interface hierarchy.

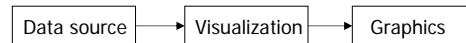
Multi-Platform Program Structure



VTK Visualization Pipeline

- Pipeline has two phases
 - Visualization phase
 - Processes up to and including mapping of data to geometry
 - Graphics phase
 - Creating the scene

Data Flow System: Pipeline execution



VTK Object Model

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

- Data Object: Represent *information* and *methods* to create, access, and delete the information.
example: the 3D point array $F(x,y,z)$
- Process Object: operates on (transforms) the data objects.
example: generating surfaces from $F(x,y,z)$

VTK Process Objects

vtkProcessObject (Source, filter, and mapper modules)

- Source
 - Interface to external data sources (readers) or generate data procedurally
- Filter
 - Transform the data input
- Mapper
 - Interface with the graphics model

VTK Process Objects

- `vtkProcessObject`
 - Filters that operate on data objects to produce new data object

ProcessObjects (aka "modules") are connected via code:

```
aProcess SetInput [anotherProcess GetOutput]
```

VtkDataObject or a subclass of it

THE UNIVERSITY OF
MEMPHIS

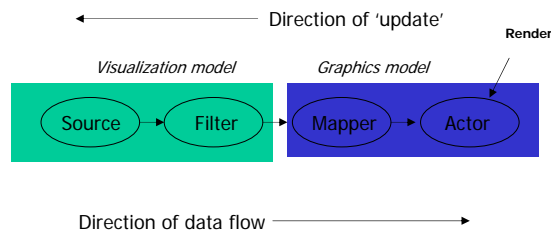
VTK Mapper Objects

- Transform and render geometry
- Interface between the visualization phase and the graphics phase
- Straddle the visualization and graphics phases
- `vtkMapper`
 - Geometric representation for an actor

THE UNIVERSITY OF
MEMPHIS

Pipeline Execution

Transform data into pictures



THE UNIVERSITY OF
MEMPHIS

Pipeline Execution

renWin Render

- "Lazy Evaluation" -- execute only when data is requested
- the `Update()` method must be called on a process-object (module) to cause it to do its job.
- generally in a pipeline, an `Update()` will be generated by the renderer to all Actors, and this will propagate back up to the source modules.

THE UNIVERSITY OF
MEMPHIS

VTK: Graphics Phase

- `vtkRenderWindow`
 - used to manage the interface between the graphics engine and your computer's windowing system
- `vtkRenderer`
 - coordinates the rendering process involving lights, cameras, and actors
- `vtkProp`
 - the stuff that appear in the scene *Props don't directly represent their geometry; instead they refer to `vtkMapper` objects, which are responsible for representing data
 - `vtkActor` is a subclasses of `vtkProp`
 - adds transformation matrix
 - Adds properties, position in world coordinates, etc.
- `vtkProperty`
 - controls the appearance of a `vtkProp`
 - color, lighting properties (diffuse, ambient, specular, etc.)
 - rendering representation (wireframe/surface/etc.)
- `vtkCamera`
 - Controls how 3D geometry is projected into the 2D image
- `vtkLight`
 - a source of light to illuminate a scene

THE UNIVERSITY OF
MEMPHIS

Visualization Process Example

Quadric function:

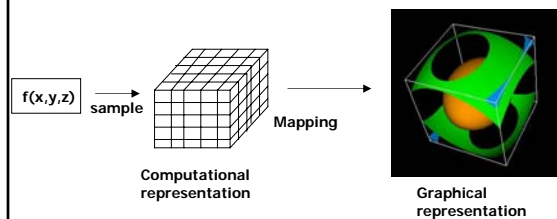
$$f(x,y,z) = a_0 x^2 + a_1 y^2 + a_2 z^2 + a_3 xy + a_4 yz + a_5 xz + a_6 x + a_7 y + a_8 z + a_9$$

Visualization Pipeline process:

- Generate source data $f(x,y,z)$
- Sample the function $f(x,y,z) \rightarrow$ a 3D array on a regular grid: $F[x][y][z] = f; (-1 \leq x,y,z \leq 1)$
- Generate a 3D surface corresponding to $f(x,y,z) = c$
- Display the surface

THE UNIVERSITY OF
MEMPHIS

Visualization Process



THE UNIVERSITY OF MEMPHIS

Object-oriented viewpoint

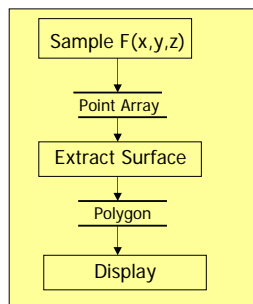
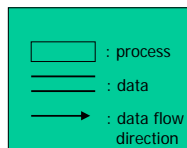
VTK Visualization Model

- Representation: **Object model**
internal data structures
- Transformation: **Functional model**
the steps to create visualization (how data flow through the visualization pipeline)

THE UNIVERSITY OF MEMPHIS

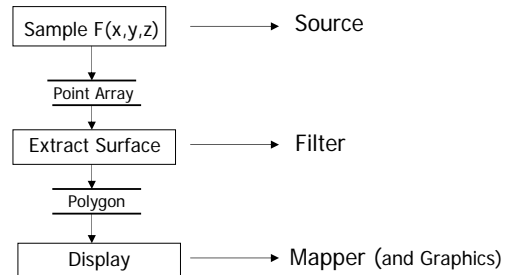
VTK Functional Model

Illustrate the steps to create visualization, also called *Visualization Pipeline* (visualization networks)



THE UNIVERSITY OF MEMPHIS

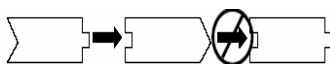
Process Objects (cont'd)



THE UNIVERSITY OF MEMPHIS

Pipeline Connections

- Type means the form of data process object accept as input or generate as output



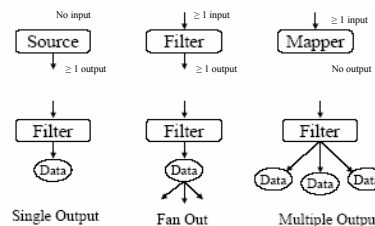
VTK is strongly typed (C/C++)

E.g., A sphere source may generate polygonal output
 A mapper may accept point geometric representation
 Output/Input of a connection must be type compatible

THE UNIVERSITY OF MEMPHIS

Pipeline Connections

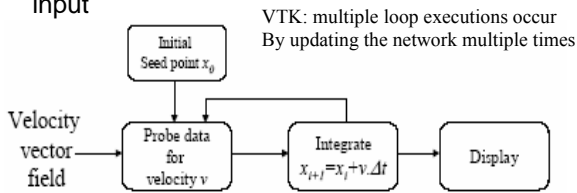
- Multiplicity refers to the number of inputs/outputs supported



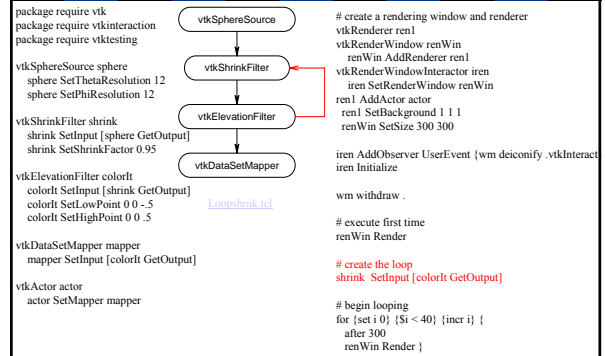
THE UNIVERSITY OF MEMPHIS

Network Loops

- Feedback loops allow us to direct the output of a process object upstream to affect its input



THE UNIVERSITY OF
MEMPHIS



THE UNIVERSITY OF
MEMPHIS

vtkShrinkFilter

- vtkShrinkFilter shrinks cells composing an arbitrary data set towards their centroid. The centroid of a cell is computed as the average position of the cell points. Shrinking results in disconnecting the cells from one another.

THE UNIVERSITY OF
MEMPHIS

vtkElevationFilter

- vtkElevationFilter is a filter to generate scalar values from a dataset. The scalar values lie within a user specified range, and are generated by computing a projection of each dataset point onto a line. The line can be oriented arbitrarily. A typical example is to generate scalars based on elevation or height above a plane.

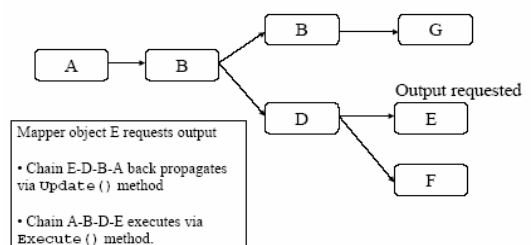
THE UNIVERSITY OF
MEMPHIS

Pipeline Execution

- VTK is demand driven
 - When output is requested by a mapper object, the network is re-executed, starting with the source object

THE UNIVERSITY OF
MEMPHIS

VTK is Demand-Driven



THE UNIVERSITY OF
MEMPHIS

Memory/Computation Tradeoffs

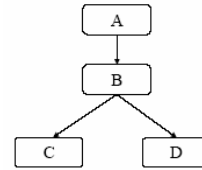
- Static Memory Model
 - Keep results in memory
 - Saves computation but memory intensive
- Dynamic Memory Model
 - Release memory
 - May need to re-execute computation
 - Saves memory but may require more computation

THE UNIVERSITY OF
MEMPHIS

Static/Dynamic Memory

VTK default uses static model

A executes
B executes
C executes
D executes



A executes
B executes
A releases memory
C executes
B releases memory
A re-executes
B re-executes
A releases memory
D executes
B releases memory

Static Model

Dynamic Model

THE UNIVERSITY OF
MEMPHIS