

## Visualization with VTK



Erik Vidholm [erik@cb.uu.se](mailto:erik@cb.uu.se)  
2004-09-20

---

---

---

---

---

---

---

---

## Outline

- What is VTK?
- What can it be used for?
- How do I actually use it?

---

---

---

---

---

---

---

---

## VTK – The Visualization ToolKit

- Open source, freely available software for 3D computer graphics, image processing, and visualization
- Managed by Kitware Inc.
- Strictly object-oriented design (C++)
- High-level of abstraction
- Use C++, Tcl/Tk, Python, Java

---

---

---

---

---

---

---

---

## True visualization system

- Visualization techniques for visualizing
  - Scalar fields
  - Vector fields
  - Tensor fields
- Polygon reduction
- Mesh smoothing
- Image processing
- Your own algorithms

---

---

---

---

---

---

---

---

## Additional features

- Parallel support (message passing, multithreading)
- Stereo support
- Integrates easily with Motif, Qt, Tcl/Tk, Python/Tk, X11, Windows, ...
- Event handling
- 3D widgets

---

---

---

---

---

---

---

---

## 3D graphics

- Surface rendering
- Volume rendering
  - Ray casting
  - Texture mapping (2D)
  - Volume pro support
- Lights and cameras
- Textures
- Save render window to .png, .jpg, ... (useful for movie creation)

---

---

---

---

---

---

---

---

## Visualization

- Data types
  - Polygonal data
    - points, lines, polygons, triangle strips
  - Images and volumes
  - Structured grids (FD)
  - Unstructured grids (FE)

---

---

---

---

---

---

---

---

## Visualization continued

- Scalar algorithms
  - Iso-contouring
  - Color mapping
- Vector algorithms
  - Hedgehogs
  - Streamlines / streamtubes
- Tensor algorithms
  - Tensor ellipsoids

---

---

---

---

---

---

---

---

## Imaging

- Supports streaming => huge datasets
- `vtkImageToImageFilter`
  - Diffusion
  - High-pass / Low-pass (Fourier)
  - Convolution
  - Gradient (magnitude)
  - Distance map
  - Morphology
  - Skeletons

---

---

---

---

---

---

---

---

## Summary +

- Free and open source
- Create graphics/visualization applications fairly fast
- Object oriented - easy to derive new classes
- Build applications using "interpretive" languages Tcl, Python, and Java
- Many (state of the art) algorithms
- Heavily tested in real-world applications
- Large user base provides decent support
- Commercial support and consulting available

---

---

---

---

---

---

---

---

## Summary -

- Not a super-fast graphics engine due to portability and C++ dynamic binding – you need a decent workstation
- Very large class hierarchy => learning threshold might be steep

---

---

---

---

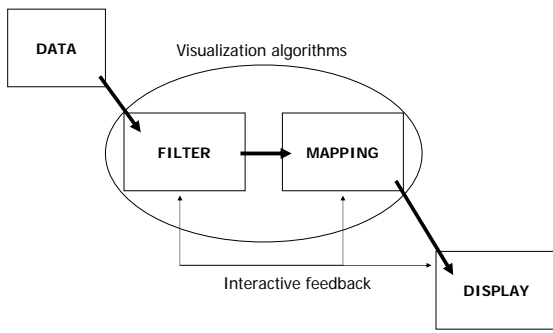
---

---

---

---

## The visualization pipeline



---

---

---

---

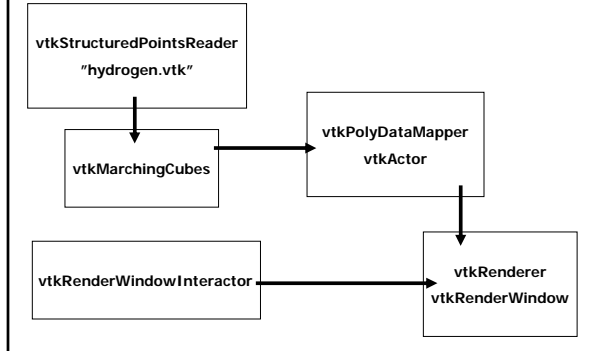
---

---

---

---

## The visualization pipeline - example




---

---

---

---

---

---

---

---

## Objects

- Data objects
  - vtkPolyData
  - vtkImageData
- Process objects
  - Source objects (vtkReader, vtkSphereSource)
  - Filter objects (vtkContourFilter)
  - Mapper objects (vtkPolyDataMapper)

---

---

---

---

---

---

---

---

## Example – Vector field visualization

```

vtkStructuredGridReader reader
reader SetFileName "office.binary.vtk"

# Create source for streamtubes
vtkPointSource seeds
seeds SetRadius 0.15
eval seeds SetCenter 0.1 2.1 0.5
seeds SetNumberOfPoints 6
vtkRungeKutta4 integ
vtkStreamLine streamer
streamer SetInput [reader GetOutput]
streamer SetSource [seeds GetOutput]
streamer SetMaximumPropagationTime 500
streamer SetStepLength 0.5
streamer SetIntegrationStepLength 0.05
streamer SetIntegrationDirectionToIntegrateBothDirections
streamer SetIntegrator integ
...
  
```

~> `vtk officeTubes.tcl`

---

---

---

---

---

---

---

---

## The hydrogen example - Python

```
# File: isosurface.py
import vtk

# image reader
reader = vtk.vtkStructuredPointsReader()
reader.SetFileName("hydrogen.vtk")
reader.Update()

# bounding box
outline = vtk.vtkOutlineFilter()
outline.SetInput( reader.GetOutput() )
outlineMapper = vtk.vtkPolyDataMapper()
outlineMapper.SetInput( outline.GetOutput() )
outlineActor = vtk.vtkActor()
outlineActor.SetMapper( outlineMapper )
outlineActor.GetProperty().SetColor(0.0,0.0,1.0)
```

**Must call update to read!** →

**Pipeline connections** →

---

---

---

---

---

---

---

---

## Example continued

```
# iso surface
isosurface = vtk.vtkContourFilter()
isosurface.SetInput( reader.GetOutput() )
isosurface.SetValue( 0, .2 )
isosurfaceMapper = vtk.vtkPolyDataMapper()
isosurfaceMapper.SetInput( isosurface.GetOutput() )
isosurfaceMapper.SetColorModeToMapScalars()
isosurfaceActor = vtk.vtkActor()
isosurfaceActor.SetMapper( isosurfaceMapper )

# slice plane
plane = vtk.vtkImageDataGeometryFilter()
plane.SetInput( reader.GetOutput() )
planeMapper = vtk.vtkPolyDataMapper()
planeMapper.SetInput( plane.GetOutput() )
planeActor = vtk.vtkActor()
planeActor.SetMapper( planeMapper )
```

**vtkContourFilter chooses the appropriate method for the data set** →

---

---

---

---

---

---

---

---

## Example continued

```
# a colorbar
scalarBar = vtk.vtkScalarBarActor()
scalarBar.SetTitle("Iso value")

# renderer and render window
ren = vtk.vtkRenderer()
ren.SetBackground(.8, .8, .8)
renWin = vtk.vtkRenderWindow()
renWin.SetSize( 400, 400 )
renWin.AddRenderer( ren )
```

**Creates a legend from the data and a lookup table** →

---

---

---

---

---

---

---

---

## Example continued

The `RenderWindowInteractor` contains functions for mouse/keyboard interaction

```
# render window interactor
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow( renWin )

# add the actors
ren.AddActor( outlineActor )
ren.AddActor( isosurfaceActor )
ren.AddActor( planeActor )
ren.AddActor( scalarBar )

# this causes the pipeline to "execute"
renWin.Render()

# initialize and start the interactor
iren.Initialize()
iren.Start()
```

The `renWin.Render()` calls `Update()` on the renderer, which calls `Update()` for all its actors, which calls...

---

---

---

---

---

---

---

---

---

---

## Example result

```
~> vtkpython isosurface.py
```

---

---

---

---

---

---

---

---

---

---

## The VTK file format

A converter from raw data to the vtk file format is available on the webpage (cba)

```
# vtk DataFile Version 2.0
Hydrogen orbital
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 64 64 64
ORIGIN 32.5 32.5 32.5
SPACING 1.0 1.0 1.0
POINT_DATA 262144

SCALARS probability float
LOOKUP_TABLE default
0.0 0.0 0.01 0.01 .....
```

---

---

---

---

---

---

---

---

---

---

Next example - Slicer

`~ > vtk slicer.tcl`

---

---

---

---

---

---

---

---

Next example - Protein

`~ > vtk protein.tcl`

---

---

---

---

---

---

---

---

## VTK and C++

- Build with CMake and your favorite compiler
- CMake generates makefiles or project files for your environment
- Use the resulting file(s) to build your executable
- With C++ you have full control and can derive own classes, but you need to write many lines of code...

---

---

---

---

---

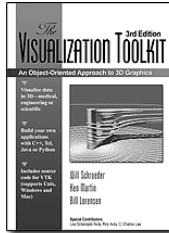
---

---

---

## VTK resources

- [www.vtk.org](http://www.vtk.org)
  - Download (source and binaries)
  - Documentation
  - Mailing lists
  - Links
  - FAQ, Search
- [www.kitware.com](http://www.kitware.com)
  - VTK Textbook
  - VTK User's guide
  - Mastering CMake



---

---

---

---

---

---

---

---

## VTK@CBA

- [www.cb.uu.se/~erik/vtk/](http://www.cb.uu.se/~erik/vtk/)
  - The examples shown today
  - Links
- Installed on the UNIX-system
- Installed on IPREN and TRIXIG

---

---

---

---

---

---

---

---

Questions?

---

---

---

---

---

---

---

---

Thanks for  
your  
attention!

---

---

---

---

---

---

---

---