An open source, freely available software system for 3D computer graphics, image processing, and visualization

Implemented as a C++ class library, with interpreted layers in Tcl/Tk, Java, Python

700 C++ classes, 350K lines of source code, 215K lines of wrapper code

Multi-platform implementation (Windows, Unix, MacOS)

Visualization Techniques: Scalar, Vector, Tensor, Texture, and Volumetric methods

Modeling Techniques: Implicit modelling, Polygon reduction/simplification, mesh smoothing, cutting, contouring, and Delaunay triangulation

Extensible system
VTK: Interaction and GUI

- Integrates seamlessly with other windowing systems, including Qt, FLTK, wxWindows, Tcl/Tk, Python, Java, etc.
- Supports a variety of interaction styles: trackball, joystick, customized interaction styles
- Implements a command/observer event handling mechanism; supports a sophisticated event model.
- Includes an extensive set of 3D widgets including point, line, plane, implicit plane, box, sphere, scalar bar, etc.
VTK: 3D Graphics Model

- **Surface Rendering**
- **Volume Rendering** (ray casting, texture-based, VolumePRO)
- Rendering Primitives: points, lines, polygons, triangle strips, volume
- Interactive Viewer/Renderer: **3D Widgets** for interaction
- **Attributes**: local reflection model, transparency, textures, flat/Gouraud shading
- **Lights**: infinite, spot
- **Cameras**: parallel/perspective projection
VTK: 3D Graphics Model (contd)

- Lights illuminate the scene
- Cameras define viewpoint
- Actors specify geometry/properties
- LOD actor (manual/automatic generation) for interactive rendering
- Assemblies group actors into arbitrary hierarchies
- Mappers define geometry/link into visualization pipeline
- Renderers coordinate lights, cameras, actors to create image
- Volumes: special actor with special properties
VTK: Visualization Data/Primitives

- **Data Types:** polygonal data, images/volumes (structured points), structured/unstructured grids, rectilinear grids

- **Cell Types:** vertex/poly-vertex, line/poly-line, triangle(strip), pixel/voxel, quadrilateral, tetrahedron, hexahedron, pyramid, wedge

- **Attributes:** scalars, vectors, tensors (3 × 3), normals, texture coords, field data
VTK: Visualization Data/Primitives
VTK: Visualization Algorithms (contd.)

- **Scalar Algorithms:** Color mapping, carpet-plots, iso-contours (marching and dividing cubes), thresholding
- **Vector Algorithms:** hedgehogs, streamlines, streampoints, stream surfaces, stream polygon
- **Tensor Algorithms:** tensor ellipsoids, glyphs, hyper streamlines
- **Modelling Algorithms**
VTK: Visualization Algorithms (contd)

- **Modeling Algorithms**
  - Primitives: sphere, cone, cube, line, plane, triangle strip generation
  - Implicit modeling, swept surfaces/volumes
  - Glyphs, cutting, clipping
  - Textures (boolean), texture thresholding
  - Decimation, 2D/3D Delaunay triangulation, mesh smoothing
  - Scattered point visualization, reconstruction, appending/merging/cleaning data

- **Data Interface:** Readers/Writers, Importers/Exporters
VTK: Example Visualizations

Plate 1 - “Hello Cone” example showing rendering window and Tcl/Tk interpreter.

Plates 2-4 - Isosurfaces from medical dataset; decimated and smoothed mesh; cut plane in structured grid.
VTK: Example Visualizations

**Plates 5 & 6** - Streamtubes in flow around post; visualization of quadric function.

**Plates 7-9** - Hyperstreamlines around point load; splatted multi-dimensional data; texture cutting to reveal inner structure of assembly.
VTK: Visualization Pipeline

- Visualization pipeline demand-driven data-flow with automatic network updates
- Reference counting to reduce memory requirements
- Sources, filters, mappers to start, process, and terminate network
- Network looping and feedback supported
- Strong type-checking to enforce filter connectivity
- Supports multiple input/multiple output filters
Implicit (Demand Driven) Execution

A process object executes only if its input or local parameters change.

- **Update:** Object requesting new output examines its input objects; recurs until source objects are reached.
- **Execution:** Starting from source objects, all objects in the path to the requesting object are executed.
- **Advantage:** Simplicity and locality, but inconvenient for parallel execution.
VTK: Imaging

Features:
- Uses cached, streaming pipeline, can operate on gigantic datasets.
- Imaging filters are multi-threaded for parallel execution.
- Fully integrated with 3D graphics/visualization pipeline.

Filter types: diffusion, Butterworth, dilation/erosion, convolution, magnitude, gradient, distance xform, FFT, histogram/threshold, Gaussian, Fourier, Sobel.