Available online at www.sciencedirect.com
ScienceDirect
SoftwareX 1–2 (2015) 9–12
SoftwareX
www.elsevier.com/locate/softx

# The Visualization Toolkit (VTK): Rewriting the rendering code for modern graphics cards

Marcus D. Hanwell*, Kenneth M. Martin, Aashish Chaudhary, Lisa S. Avila

*Kitware, Inc., 28 Corporate Drive, Clifton Park, NY 12065, USA*

Received 2 March 2015; received in revised form 10 April 2015; accepted 13 April 2015

**Abstract**

The Visualization Toolkit (VTK) is an open source, permissively licensed, cross-platform toolkit for scientific data processing, visualization, and data analysis. It is over two decades old, originally developed for a very different graphics card architecture. Modern graphics cards feature fully programmable, highly parallelized architectures with large core counts. VTK's rendering code was rewritten to take advantage of modern graphics cards, maintaining most of the toolkit's programming interfaces. This offers the opportunity to compare the performance of old and new rendering code on the same systems/cards. Significant improvements in rendering speeds and memory footprints mean that scientific data can be visualized in greater detail than ever before. The widespread use of VTK means that these improvements will reap significant benefits.


*Keywords:* Visualization; Toolkit; Data analysis; Scientific data

## Code metadata

| | |
|---|---|
| Current code version | v6.2.0 |
| Permanent link to code/repository used for this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-15-00004 |
| Legal Code License | 3-clause BSD |
| Code versioning system used | git |
| Software code languages, tools, and services used | C++, Python, MPI, OpenGL |
| Compilation requirements, operating environments & dependencies | C++ compiler, OpenGL 2.1+, Windows, Mac OS X, Linux and experimental Android/iOS support |
| If available Link to developer documentation/manual | http://www.vtk.org/doc/release/6.2/html/ |
| Support email for questions | vtkusers@vtk.org |

## Software metadata

| | |
|---|---|
| Current software version | v6.2.0 |
| Permanent link to executables of this version | http://www.vtk.org/files/release/6.2/vtkpython-6.2.0-Windows-64bit.exe —Windows 64 bit<br>http://www.vtk.org/files/release/6.2/vtkpython-6.2.0-Darwin-64bit.dmg —Mac OS X<br>http://www.vtk.org/files/release/6.2/vtkpython-6.2.0-Linux-64bit.tar.gz —Linux 64 bit<br>http://www.vtk.org/files/release/6.2/VTK-6.2.0.tar.gz —source code tarball |
| Legal Software License | 3-clause BSD |
| Computing platforms/Operating Systems | Linux, OS X, Microsoft Windows, Unix-like, Android, iOS |
| Installation requirements & dependencies | OpenGL 2.1+, Windows, Mac OS X, Linux and experimental Android/iOS support |
| If available, link to user manual—if formally published include a reference to the publication in the reference list | http://www.vtk.org/doc/release/6.2/html/ |
| Support email for questions | vtkusers@vtk.org |

* Corresponding author.
*E-mail address:* marcus.hanwell@kitware.com (M.D. Hanwell).

http://dx.doi.org/10.1016/j.softx.2015.04.001

## 1. Motivation and significance

VTK is an open source, permissively licensed (3-clause BSD), cross-platform toolkit for scientific data processing, visualization, and data analysis. It has been developed to offer reproducible visualization and data analysis pipelines for a range of scientific data, across domains such as medical imaging, chemistry, cosmological data, computational fluid dynamics, and finite element analysis. It implements a number of visualization modalities that include 3D polygonal, glyphing, volume rendering as well as 2D charting/rendering capabilities. These capabilities are developed primarily in C++, and automatically wrapped in Python, Java, and Tcl to offer advanced data processing, visualization, and analysis to scientists directly or through applications built upon the libraries.

It was initially developed by the three founders of the project, Ken Martin, Will Schroeder, and Bill Lorensen, to provide reusable examples accompanying a book on object oriented computer graphics programming [1,2]. VTK is among the oldest open source toolkits that is still actively developed, with over two decades of development by a large and distributed development team. The toolkit provides reusable software components for use in scientific data visualization and analysis. One of the primary aims of the project is to provide state-of-the-art implementations of visualization algorithms, such as marching cubes [3], that can be examined by the community, adapted, and reused in scientific data analysis and visualization.

For a number of years the rendering capabilities had lagged behind what was possible with the latest graphics cards. In order to take full advantage of them it required a significant software engineering investment to rewrite the rendering code to target modern programmable cards. The graphics programming interface has changed significantly in the last twenty years, with consolidation from multiple competing standards down to OpenGL [4] that is available across a number of platforms and operating systems, and DirectX [5] available for operating systems developed by Microsoft.

The VTK project has been reused in a large number of applications for scientific data visualization and analysis across a number of scientific domains. It has established best practices in the field using OpenGL as its primary rendering interface, and it has been the foundation of a number of research projects involving visualization and data analysis. The code has been developed in a portable fashion, and has been built on embedded systems, mobile phones, desktop/laptop computers, and supercomputers.

## 2. Software description

VTK is written primarily in portable C++ [6], using CMake [7] to build on multiple platforms. At its core there is a data pipeline that is connected together to form a working analysis pipeline. There are sources, such as data from file readers, the network, etc., filters that act upon the data, and sinks such as file writers or mappers that render data to the screen [8].

The permissive, OSI-approved 3-clause BSD license was chosen for its simplicity and because it promotes shared ownership of the code. It offers everyone the same access and opportunity to develop open or proprietary projects using it (in contrast to dual-licensing using "copyleft" as an inducement to purchase licenses). This promotes a service-based model for commercial activity, and promotes maximum reuse of the code in all settings/sectors.

The toolkit is composed of a number of software libraries, or 'modules', that encapsulate a related set of functionality, or bring a major dependency into the standard programming interface provided. The rendering code is separated into interface modules that provide the programming interfaces, common logic, etc., and the implementation modules that override interface classes with concrete implementations. The mechanism developed offers compile time and run time control of these overrides, but the primary mechanism employed is compile time.

The legacy rendering code is in a group of implementation modules collectively called "OpenGL", whereas the new rendering code described is a drop-in replacement set of implementation modules collectively called "OpenGL2". This code sits at the end of the data pipeline, and is principally responsible for mapping data from the pipeline to the screen. This part of the code is critical to interaction with scientific data, and often represents one of the major bottlenecks to understanding data. At some point in the future the default set of implementation modules will be switched to "OpenGL2".

A testing suite has been developed alongside VTK to verify visualizations are consistent across platforms, and as changes are merged into the main code base. These tests provide automatic verification of functionality, and provide pixel-to-pixel comparisons. Any differences are uploaded to a software quality dashboard, with a mismatch being considered a test failure. This suite of tests has played a central role in aiding the rewrite of the VTK rendering code, offering validation of the new rendering code across a number of operating systems and graphics cards/implementations.

Some of the major features of VTK include polygonal rendering of data, with options to color the surface based upon secondary parameters using a number of color mapping techniques. Volume rendering of data is also available, with an array of rendering options. There are a number of more specialized rendering modes, such as glyphing for scenes with repeats of the same geometry and impostor spheres/cylinders for chemical data. The 3D scenes are fully interactive, and there are a selection of 3D widgets to aid in interaction. There are also a number of 2D rendering classes providing support for interactive charts using the same data pipelines, and OpenGL for the rendering.

## 3. Illustrative examples

The VTK project has been reused in a number of projects, some examples of which include ParaView [9], VisIt [10], 3D Slicer [11], the Medical Imaging Interaction Toolkit (MITK) [12], Mayavi [13], Reactor Geometry Generator (RGG), Computational Model Builder (CMB), tomviz [14,15], MongoChem, Avogadro 2 [16], UV-CDAT, VisTrails, and many