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 http://www.vtk.org/files/release/6.2/vtkpython-6.2.0-Darwin-64bit.dmg —Mac OS X http://www.vtk.org/files/release/6.2/vtkpython-6.2.0-Linux-64bit.tar.gz —Linux 64 bit 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.

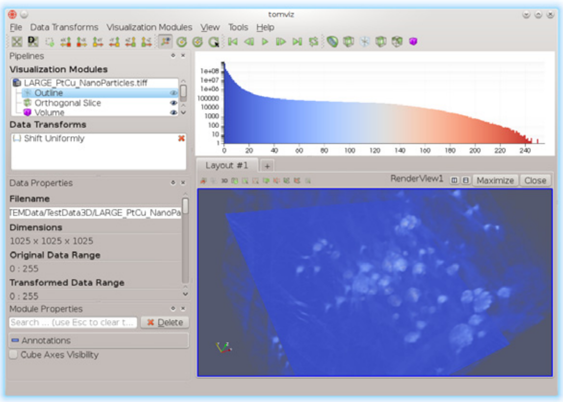
The legacy rendering code is in a group of implementation modules collectively called "OpenGL", whereas the new rendering code described is a drop-in replacement set of implementation modules collectively called "OpenGL2". This code sits at the end of the data pipeline, and is principally responsible 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



Filename
TEMData/TestData3D/LARGE_PtCu_NanoPa
Dimensions
1025 x 1025 x 1025
Original Data Range
0 : 255
Transformed Data Range
0 : 255

Fig. 1. The tomviz application that uses VTK and ParaView to build an environment for electron microscope tomography applications in materials science.

projects across different domains including one-off codes. These programs span a large range of scientific domains, and due to the permissive open-source licensing it is difficult to know of all the areas in which VTK has been used. Many proprietary applications make use of VTK, including desktop, web, and mobile applications across a number of industries.

The benefits of the rendering code rewrite have yet to be realized, but they are likely to be significant. The ParaView and 3D Slicer applications can already be built with the new rendering code, as can tomviz (shown in Fig. 1), and CMB. The RGG project is making use of rendering code only available in the "OpenGL2", and is probably the first application to require the new rendering code.

4. Impact

The impact of the VTK project has been significant over its 20 year history, finding uses across a large number of domains in scientific data visualization, analysis and related areas. The creation of the project led to a successful spin-off company, Kitware, Inc., which was founded in 1998—some 17 years of operation. The company was founded around the VTK project, and later went on to develop a number of new projects. It has enjoyed year-on-year growth, is privately owned, and has been profitable since creation. VTK remains a flagship project, along with ParaView that provides a turnkey desktop application leveraging VTK functionality.

The VTK project has had over 100,000 commits made by over 250 contributors according to Open HUB's analysis of source history [17]. A large number of commits and code were written by core developers, but the raw contributor numbers make it clear that many contributions have been contributed by members of the open source community (with two of the top five all-time contributors being external contributors). VTK has also taken part in the Google Summer of Code three times, encouraging students to propose summer projects (funded by Google) mentored by VTK experts.

The VTK project, and others that came after, have been developed using a "Platform Strategy", where Kitware experts develop powerful software platforms, using liberal licensing models to promote shared ownership of the platform with partners in national laboratories, universities, government, and industry. A software services model focuses on the creation of new functionality, and exploring challenging areas in scientific data visualization and analysis while delivering state-of-the-art software platforms.

Table 1
First frame render times, and relative speedup.

Triangles (M)	OpenGL 1 (s)	OpenGL 2 (s)	Speedup
1	0.423	0.046	9.1
5	2.356	0.219	10.7
20	9.659	0.802	12.0
30	14.459	1.167	12.4

Table 2
Subsequent frame render times (average of 50), and relative speedup.

Triangles (M)	OpenGL 1 (s)	OpenGL 2 (s)	Speedup
1	0.0003	0.0008	0.5
5	0.506	0.003	200.0
20	2.010	0.007	282.3
30	3.005	0.011	286.2

The rendering rewrite is not yet complete, but it is set to have enormous impacts in the coming years. Rendering performance was becoming a major bottleneck in VTK, with scientific data in virtually all areas growing in size rapidly. The graphics card has changed significantly, with the OpenGL programming interfaces undergoing significant change. It has moved from a fixed function system, to a dynamic, programmable, highly parallel mathematical processor with core counts far outstripping the main system.

The rewrite has led to significant speedups for a number of rendering scenes, and it has also reduced the memory footprint when rendering identical geometries. This means that geometries that were simply too large to load/render previously are now possible. It also means that there is a significant reduction in rendering time for equivalent data, offering the possibility of applying more advanced rendering techniques previously not possible if interactivity were to be maintained.

Some sample benchmarks were taken on a 64 bit Linux system, and are representative of results seen on all operating systems that support both rendering implementations. Tables 1 and 2 show the average time to first render, and an average subsequent render taken over a number of frames. This code used to benchmark is located at 'Utilities/Benchmarks/GLBenchmarking.cxx' in the source tree, and './bin/GLBenchmarking –start 8 –end 14' was used to obtain timings in this range. They show that initial render times are an order of magnitude faster, and subsequent render times are two orders of magnitude faster. This represents the best case, for a single large triangular mesh, but most rendering scenarios have benefited from significant speedups.

The core rendering programming interfaces in VTK have not changed, and so for many applications it is possible to simply rebuild them with the new rendering code without any significant porting effort. In these cases existing codes will benefit from these improvements, with the VTK development team providing guidance on how to port existing codes that are more complex. The ParaView application is one of the more complex VTK-based applications, and its transition to the new rendering code has highlighted a number of issues that were fixed.

5. Conclusions

The modern graphics card has changed significantly, and rewriting the rendering code to take advantage of advances in modern graphics rendering techniques has yielded significant performance improvements. The VTK project provides a reusable software framework that can be used across a number of scientific domains, with proven open source and commercial applications. The improvements to the rendering not only offer faster rendering of larger data, but free up precious computational resources to perform additional analyses, or to perform increasingly complex visualizations of data to aid in understanding. The open source, cross platform code base offers a platform for verified, reproducible, open science that can be reviewed by the wider community.

Acknowledgments

We would like to acknowledge the National Institutes of Health for sponsoring this work under the grant NIH R01EB014955 “Accelerating Community-Driven Medical Innovation with VTK”. We would also like to take the opportunity to thank the community that has developed, supported, and promoted VTK over its lifetime.

References

[1] Schroeder W, Martin K, Lorensen B. An object oriented approach to 3d graphics. 4th ed. Kitware, Inc.; 2004.
[2] Geveci B, Schroeder W. VTK. In: Brown A, Wilson G, editors. The architecture of open source applications. 1st ed. 2012. lulu.com.
[3] Lorensen WE, Cline HE. Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comput Graph 1987;21:163–9.
[4] Shreiner D, Group TKOAW. OpenGL programming guide: the official guide to learning opengl, versions 3.0 and 3.1. Addison-Wesley Professional; 2009.
[5] Perez A, Royer D. Advanced 3-D game programming with directx 7.0. 1st ed. Plano, TX, USA: Wordware Publishing Inc.; 2000.
[6] Stroustrup B. The C++ programming language. 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2000.
[7] Martin K, Hoffman B. Mastering CMake. 5th ed. Kitware, Inc.; 2010.
[8] The VTK user’s guide. 11th ed. Kitware, Inc.; 2010.
[9] Henderson A, Ahrens J, Law C. The ParaView guide. 2004.
[10] Childs H, Brugger E, Whitlock B, Meredith J, Ahern S, Pugmire D, Biagas K, Miller M, Harrison C, Weber GH, Krishnan H, Fogal T, Sanderson A, Garth C, Bethel EW, Camp D, Rübel O, Durant M, Favre JM, Navrátil P. VisIt: An end-user tool for visualizing and analyzing very large data. In: High performance visualization–enabling extreme-scale scientific insight. 2012. p. 357–72.
[11] Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J, Fillion-Robin J-C, Pujol S, Bauer C, Jennings D, Fennessy F, Sonka M, Buatti J, Aylward S, Miller J, Pieper S, Kikinis R. 3d slicer as an image computing platform for the quantitative imaging network. Mag Reson Imaging 2012;30:1323–41.
[12] The Medical Imaging Interaction Toolkit (MITK), Online. 2015. URL http://mitk.org/.
[13] Mayavi, Online. 2015. http://code.enthought.com/projects/mayavi/.
[14] tomviz web site, Online. 2014. URL http://tomviz.org/.
[15] Hanwell MD, Ayachit U, Hovden R, Muller DA, Maynard R, Boeckel B. tomviz 0.4.0, 2014. URL http://dx.doi.org/10.5281/zenodo.12723.
[16] Hanwell MD, Curtis DE, Lonie DC, Vandermeersch T, Zurek E, Hutchison GR. Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. J Cheminformatics 2012;4.
[17] The Visualization Toolkit Open Source Project on Open HUB, Online. 2015. https://www.openhub.net/p/vtk.