It's been a hot, hot summer so far in the Northeast U.S., and there has been no exception at Kitware which is why we've been bringing our own heat in the form of major releases for our most popular toolkits: VTK and ParaView!

The team has been busy creating new blog posts and videos to support these and other toolkit releases. We encourage you to follow along with our updates and progress by subscribing to our blog (kitware.com/blog), liking us on Facebook, or following @Kitware on Twitter.

We're always looking to learn more about what the community is doing with our open-source tools. If you’re interested in contributing to a future issue of the Source, please contact editor@kitware.com for more details on the process. Writing for the Source is an excellent way to showcase your open-source project and present it to some of the top technical minds and developers in the world.

**Editor’s Note**

Recent Releases .......................................................... 1
What’s New in VTK 6? ....................................................... 3
RTK: The Reconstruction Toolkit ....................................... 8
ClimatePipes: User-Friendly Data Access, Manipulation, Analysis & Visualization of Community Climate Models ...... 10
Avogadro 2 and Open Chemistry ......................................... 14
Kitware News ................................................................ 19

**Recent Releases**

**PARAVIEW 4.0.1 RELEASED**

ParaView 4.0.1 was released in mid-June, marking the first major change in version number since 2007. This release also marks a milestone in the gradual progression of the user interface and framework, and addresses more than 130 issues since the release of ParaView 3. ParaView 4.0.1 is based on VTK 6.0, and includes all related fixes and enhancements that went into that release.

Major changes included in the ParaView 4.0.1 release were highly-requested on ParaView user voice - so thank you for making your voice heard, as well as for your contributions to this release!

Major updates and new additions in ParaView 4.0.1 include enhancements for visualizing multi-block datasets, updates to the Properties panel, support for creating extended annotations with mathtext, the ability to export vector graphics, a completely redesigned ParaViewWeb, updated Co-Processing modules, an improved color legend, and consistent online documentation.

It’s now easier to interact with multi-block datasets as it has been made easier to change display parameters such as visibility, color, and opacity. The Multi-Block Inspector enables users to specify such properties on a per-block basis. Block properties may also be changed from the context menu in the #d view. The Properties panel has also undergone some more changes in 4.0.1, including further cleanup, bug-fixes, and enhancements and the ability to group and create widgets based on their functionality.
A new feature included in this release is the ability to add mathematical markup to annotation text. ParaView can now support equation rendering support in the matplotlib package to generate mathematical equations in selected fields. To use this new feature, simply encase the applicable text in dollar signs: $....$; anything contained within these symbols will be treated as mathtext. Another often-requested feature that is now included in ParaView 4.0.1 is support for vector graphics. It’s now possible to export scenes from 3D views and charts, such as eps, pdf, or svg vector graphics. All text and annotations such as scalar bars, cube-axes, and curves in charts are exported as vector graphics, thus ensuring crisp reproduction for publications.

ParaViewWeb has been completely redesigned to use the latest HTML5 based technologies, such as WebSocket and WebGL. This enables communication with a ParaView server running on a remote visualization node or cluster using a lightweight JavaScript APT, which can make it easy to embed interactive 3D visualization component.

Co-Processing support has undergone several usability improvements to enable users to build adapters externally, and the Python co-processing modules and scripts were cleaned up. More information on this update will be featured in the upcoming Catalyst/Co-Processing User’s Guide.

Lastly, major improvements have been made to the Color Legend. In addition to general updates, the scalar bar can now accept MathText for the title, subtitle, and annotation text; Greek letters and other math symbols can also be included in the legend. This requires Python support with matplotlib available, which is included in the distributed binaries. Colormaps can also be marked to indicate whether their domains are ratio or interval-scaled, or categorical. Categorical colormaps are not shown as a continuous range of colors but as a discrete set of color swatches; text accompanying each value can be updated in the Annotations tab of the color legend editor.

As always, we appreciate and encourage your feedback. Stay tuned to the Kitware Blog for more details on Catalyst and ParaViewWeb, and for information on upcoming releases!

**VTK 6.0.0 RELEASE**

In late June, the VTK community released VTK 6.0, the first major release since 5.0 in December 2005! This release includes significant refactoring of VTK’s pipeline and build system. Pipeline changes have cleanly separated the Algorithm, DataObject, and Executive classes. Build system changes automate and simplify intra-library build time dependency analysis. Together these changes facilitate the addition of new features to and removal of unwanted features from VTK.

In addition to a number of low-level changes, there are also a number of new features in VTK 6 that are more readily-apparent to users of applications built with VTK; many of these new additions have also been included as part of ParaView 4. This includes the ability to map values to discrete color sets; vtkScalarstoColors and subclasses new support an IndexedLookup mode where only values that exactly match an annotation are assigned a color.

VTK’s text rendering capabilities have also been greatly extended. Text can be output to vector-graphics friendly raw Bezier curves, system fonts may be accessed through the FreeTypeFontConfig module, and mathematical equations can be rendered with the matplotlib Python package. The charting functionalities have seen a number of changes as well, including the addition of 3D charts and seamless transitions between 2D and 3D chart projections.

In regards to filters, there is a new polyhedron surface mode where users draw arbitrarily-shaped regions on the screen and extraction filters return all the surface elements within. The serial and parallel particle tracing filters, which track particles in time-varying vector fields, now run faster and more efficiently. Lastly, there are new Adaptive Mesh Resolution and HyperTree data types and filters included that are designed for visualization of large, multi-resolution scientific datasets. There is also a new vtkDeformPointSet filter included in this release.

Lastly, VTK 6 can leverage modern Mesa 3D OpenGL capabilities as it is no longer limited to the GL version 1.1 support whenever Mesa is in use. This extends VTK’s rendering capabilities, especially in HPC applications where rendering is typically done off-screen on supercomputers that lack graphics hardware.

To learn more about VTK 6 and to see a full list of changes, fixes, and enhancements made in this release, please visit vtk.org. To contribute to VTK, please join the mailing list!
**ITK 4.4.0**

In early June the Insight Toolkit (ITK) team released ITK 4.4.0. Major updates in this release included the addition of two new ImageIO modules: MINC and SCIFIO. The MINC ImageIO module provides a bridge to the MINC file format, a powerful medical imaging format used by the MINC library. The new MINC module has Image IO and Transform IO capabilities and is considered experimental; it is enabled by turning Module_ITKIOMINC ON in CMake’s configuration.

A new SCIFIO module has also been included; SCIFIO is a refactoring of the Bio-Formats library, and the module provides access to a wide range of file formats encountered in life science/microscopy environments. This experimental feature is available Remote module; to enable it, set the Fetch_SCIFIO CMake variable to ON. Many improvements and fixes are ongoing for this module; Git master is recommended for those interested in this module.

A number of improvements were also made to DCMTK ImageIO, but this module remains experimental and disabled by default. To try DCMTK ImageIO support, turn on the CMake option Module_ITKIODCM. On Unix platforms, the supporting DCMTK library will automatically be built as a CMake ExternalProject; on Windows, it must be built independently of the ITK build system. Then, specify the location to the external build after setting the CMake option ITK_USE_SYSTEM_DCMTN ON.

ITK 4.4.0 also includes a number of performance improvements to QuadEdgeMesh processing and memory copy functions. A new ImageScanlineIterator is available. This iterator can be used in many of the cases where an ImageRegionIterator is applied, but it has better performance. An API design change was also started to improve the implied functionality of the GetObject methods. The non-const “GetObjectMacro” has been replaced with a “GetModifiableObjectMacro” which will result in a GetModifiableXXX method instead of a GetXXX method.

In regards to the build system in ITK 4.4.0, the minimum CMake required version was bumped to 2.8.8 on Windows and 2.8.5 otherwise. This may be increased even higher in the near future to ensure that all CMake features used operate correctly.

Known assertion failures occur when compiled under Visual Studio in the Debug configuration. This will be fixed in 4.4.1. If this has a significant negative impact on your work, please consider contributing a Nightly dashboard build under this configuration. As previously scheduled, Visual Studio 2005 is no longer supported. However, support for building against system third party libraries continues to improve, and further improvements are expected for version 4.5.

For more information on ITK 4.4.0, please visit itk.org or contact the insight-developers@kitware.com mailing list.

**OPEN CHEMISTRY BETA RELEASE**

In April, the Open Chemistry team (openchemistry.org) announced the beta release of the suite of cross-platform, open-source tools and libraries: Avogadro 2, MoleQueue, and MongoChem. These three desktop applications can be used independently or together.

Avogadro 2 is a rewrite of Avogadro, the advanced molecular editor and visualization platform, and addresses many of the previous limitations in the rendering code, scalability, scriptability, and increased flexibility; these changes have enabled the development team to effectively address current and upcoming challenges in computational chemistry and related fields. This update also included the relicensing of Avogadro to the three-clause BSD license.

MoleQueue is a new application developed to execute computational chemistry codes locally and remotely. Rather than adding this functionality to Avogadro 2, it has been developed as a standalone, system-tray resident application that runs a graphical application and a local server. MoleQueue also provides a Qt 4-based client library that can easily be integrated into Qt applications for a familiar signal-slot-based API for job submission, monitoring, and retrieval.

MongoChem is another new application developed for Open Chemistry that leverages MongoDB, VTK, and AvogadroLibs to provide chemical informatics on the desktop. It aims to address the need for researchers and groups to be able to effectively store, index, and retrieve relevant chemical data. MongoChem supports the use of a central database server where all data can be housed, and enables the significant feature-set of MongoDB to be applied, such as sharding, replication, and efficient storage of large data files.

All the Open Chemistry tools can be used together via a JSON-RPC 2.0-based API. To learn more about the Open Chemistry suite of tools and how they can be applied to your project, please visit openchemistry.org. Following the mantra of “release early, release often,” the team encourages community feedback as they prepare for the first major release of Open Chemistry. As such, Avogadro 2 and MoleQueue both had 0.6 releases in July.

---

For more information on ITK 4.4.0, please visit itk.org or contact the insight-developers@kitware.com mailing list.
VTK 5.10.0 was tagged on May 12, 2012, with a bug fix release made on October 15. For many months, we worked on the modularization of VTK using a set of tools to automate the file moves and maintain a testing tree that was parallel to the current master. A lot of that work was documented in a recent blog post about “VTK Modularization and Modernization,” with the largest motivation being to update the VTK build system to facilitate our growth and evolution as one of the world’s most powerful and extensive visualization libraries. Once the modularization was merged into the main tree, we continued working on many new features and removing deprecated APIs, code, including the VTK 4 compatibility API.

**NEW FEATURES**

Major new features in VTK 6 include enhanced support for patched AMR, which comprises updated readers for Enzo and Flash format files and demand-driven filters for re-sampling and slicing AMR datasets. The latter changes enable users to visualize complex AMR simulation data while accessing only the blocks that are necessary and in only as much detail as necessary to display the details of interest. As part of this work, we’ve developed filters that will automatically generate ghost layers in distributed structured datasets in case they are not present in the input.

We’ve also optimized particle tracing in time-varying vector fields in this release. The vtkParticleTracer, vtkStreaklineFilter, and vtkParticlePathFilter classes, along with their parallel counterparts, have been rewritten to use the pipeline more effectively and to cache and reuse previously-computed results when possible. As part of this work the VTK pipeline’s treatment of time-varying data was refactored, shifting more of the responsibility from the executives and datasets into vtkMultipleTimeStepAlgorithm.

It is also now possible to smoothly and continuously warp arbitrarily complex polygonal meshes, and other datasets that are represented by explicit point coordinates, using the method of mean value coordinates and the vtkDeformPointSet filter.

The charts saw several enhancements, such as the introduction of the scatter plot matrix support which features seamless 2D-3D-2D transitions when animating between the different columns of the input data. The support added for 3D charts in the 2D chart API then led to the development of a new set of 3D charts supporting surfaces, points, and lines with reinforcement for fixed size charts where data can be panned and zoomed, and the charts rotated. Several new selection modes and interactions were added, including support for polygonal selection in addition to the standard rectangular selection box.

Vector graphics export to formats such as postscript, SVG, and PDF has been improved following a rewrite of the vtkGL2PSExporter class. Several related improvements to text rendering have been made in this release as well; text can now be rendered to a vtkPath object as a raw Bezier curve, for example; system fonts may now be used in VTK renderings by enabling the Rendering/FreeTypeFontConfig module on platforms that support fontconfig; and mathematical equations can be rendered using the matplotlib Python package. With this release, enclosing latex-style markups between dollar signs will produce pretty-printed equations in text actors and charts when the Rendering/Matplotlib module is enabled and matplotlib is installed.

The generalized mechanism by which VTK calls into Python mentioned above is itself a new feature in this release. This code can be found in the UtilitiesPython module. The interface to Python was promoted from ParaView, where it forms the basis of ParaView’s Python Programmable Filter. Likewise, the XDMF file format, a meta-format for describing HDF5 data, has been promoted from ParaView into VTK for the first time as part of this 6.0 release. This type of code reuse is simplified because of VTK’s new modular build system, which is described more later in this article.
VTK can now map values into discrete color sets. 'vtkScalarsToColors' and subclasses now support 'IndexedLookup' mode where only values that exactly match an annotation are assigned a color. Similarly, the 'vtkScalarBarActor' has been refactored to enable drawing of color swatches for lookup tables when the "indexed lookup" feature is turned on, and draw a "NaN" swatch to indicate the color used for not-a-number and infinite values. Annotations on the ScalarBarActor also make use of the new text rendering capabilities mentioned previously.

Figure 2: Indexed lookup color map.

For the first time, VTK also has a polyhedron surface selection mode. This extends VTK’s ability to select surface elements underneath a rectangular region of the screen with a variation that allows the user to draw arbitrarily shaped regions on the screen and return all the surface elements within.

Figure 3: Selecting odd-shaped features in VTK.

This release also marks a first foray into GPGPU filtering within VTK. The AcceleratorsPiston module is a module that interfaces to LANL’s Piston library. Piston is a library of visualization operators that are built on top of NVidia’s Thrust parallel processing library. The new VTK module is a set of filters that transport data from VTK into Piston, orchestrate the execution of Piston operators to process data on the GPU, render the data directly on the GPU, and then optionally bring the results back into VTK for further processing on the CPU.

MODULARIZATION
Modularization involved rewriting large pieces of the build system, taking advantage of many new features in CMake, moving almost all of the source code into new modules, and automating the addition and removal of modules as much as possible. We went from a set of 19 statically-defined kits with hardwired dependencies and wrapping code, to 160 modules where almost every aspect of the modules were processed automatically.

Two files were used to script the modularization of the VTK source tree: a Python file written by the Boost community, and a manifest file that describes the file moves, deletions, creations, and patches. This allowed us to get most of the new build system working without undue interruption to the normal development process, with some projects even using the modularized source before it was committed to master. On April 9, 2012, Brad King, Chris Harris, and I gathered in Brad’s office and pressed the button, running the modularization scripts one final time to create a sequence of commits that moved all of the source files, patched the source, and introduced the new build system.

SIMPLE MODULE NAMING SCHEME
We decided early on that we wanted to use a simple naming scheme and source tree layout. All source code is two levels deep, with a top-level grouping (one word) and a second level (one or more words), which means that it is very simple to go from module name to source code location and back again; for example, vtkCommonDataModel is in Common/DataModel and Rendering/Core code is exposed in the vtkRenderingCore module. The tests now all contain the name of the module they are located in and the language the test is written in, again making it much simpler to go from the name of a test to a location in the source tree. All names are camel-cased, with a lower case ‘vtk’ prefix.

ANATOMY OF A MODULE
The new VTK build system processes modules in two passes. The first scans the source tree for files named module.cmake which are two directories deep; when found, the files are included, which causes a CMake macro called vtk_module to run. This macro is used to declare the name of the module, its dependencies, group memberships, and any other special metadata. We kept the syntax as simple as possible, and a standard macro can be defined in just a few lines. A more complex module, such as vtkRenderingCore, takes a few extra lines.
BUILD TIME OPTIONS SHOULD NOT CHANGE API/MODULES

VTK had increasingly difficult-to-follow logic where the kits would be built differently depending upon what was available on the system. This led to the rendering kit linking to Qt, the Qt kit linking to QtWebKit, if available; and other behaviors that were not always desirable. In VTK 6, we decided to disallow options that changed API/linking in modules, with one-or-two exceptions where it was hard to avoid. The Qt kit was split into several modules, including a vtkGUIStuffQtWebKit, which contains the webkit-dependent code. Things such as vtkRenderingOpenGL had to make exceptions to account for different windowing systems, OpenGL libraries, etc.

OPTIONAL USE OF THE OBJECT FACTORY

In previous VTK releases, all vtkObject-derived classes used the object factory in their static new method. While this allowed for a great deal of flexibility as any class could be overridden by dependent code, it also came at a great price in terms of run time cost for instantiation. With the release of VTK 6, the default vtkStandardNewMacro will return the class; the new vtkObjectFactoryNewMacro will always use the object factory; and the vtkAbstractObjectFactoryNewMacro will always use the object factory and potentially return NULL if no override was specified. The VTK_ALL_NEW_OBJECT_FACTORY option enables you to switch back to the old behavior at compile time by using the object factory for all vtkObject-derived classes.

OBJECT FACTORY INITIALIZATION

A big change in VTK 6 is the introduction of implementation modules. These move VTK from hardwiring object factory overrides in an ad-hoc fashion for each kit at compile time, to initializing the object factory overrides at link/runtime. We thought long and hard about how to reduce the pain as much as possible here, and came up with the solution in VTK 6.0 where you must state the implementation modules you rely upon and ensure your binary is compiled with the generated compiler definitions, as documented here.

The interface classes, such as vtkRenderWindow, are abstract and therefore it is not possible to return anything except NULL if their new method does not have a registered override. Linking to vtkRenderingOpenGL and adding the compiler definitions to your application’s build system will ensure that the object factory overrides are initialized whether you are linking to dynamic or static libraries. These overrides change highlighted areas where interface classes were inherently linked to OpenGL. Changes were also introduced to allow other back-ends, ultimately opening the door to alternative implementation modules. This pattern was also used to enhance database-related classes, where each implementation module registers the additional database backends (see vtkIOMySQL for example).

```
vtk_module(vtkRenderingCore
  GROUPS
  Rendering
  DEPENDS  vtkCommonExecutionModel
            vtkCommonTransforms
            vtkFiltersSources
            vtkFiltersGeneral
            vtkFiltersGeometry
            vtkFiltersExtraction
            vtkIOImage
            vtkIOXMLParser
  COMPILATION
  vtkUtilitiesMaterialLibrary
  TEST_DEPENDS
  vtkIOXML
  vtkTestingCore
  vtkTestingRendering
  vtkRenderingContext2D
)
```

The first argument is always the name of the module, which in this case is vtkRenderingCore. After that named arguments are used (all other arguments are optional), such as: GROUPS to specify membership in an option group; DEPENDS for public dependencies the module will link to; COMPILATION for modules that are needed at compile time but are not necessarily linked to; and TEST_DEPENDS that the tests depend upon in addition to the module dependencies. The dependencies are all recursive, and so depending on vtkCommonDataModel brings in vtkCommonCore, which is a dependency of vtkCommonDataModel for example.

There is an accompanying CMakelists.txt file in the same directory, which contains the standard logic for building a module. This file has access to everything a normal CMakelists.txt file has, in addition to some VTK-specific macros and functions that integrate with the dependency information specified in the module.cmake files. When adding a module here, it should be added with the same name as specified in the module.cmake file, using the vtk_module_library function. This will export module properties for the wrappers, ensure the headers are installed, and link to anything specified as a dependency of the module. Once this call has been made, a standard CMake target will be created and it can be used with all of the standard CMake commands, such as target_link_libraries and set_target_properties, if any special treatment is required.

The directory is added in the second pass, and in the correct order assuring all dependencies were already added, using a standard add_subdirectory call with variables set up correctly for the module; as a result, variables such as vtk-module will be set to the name of the module specified in module.cmake.
FINDING AND USING VTK 6

Any application using the standard OpenGL rendering in VTK will likely want to link to vtkRenderingOpenGL and vtkInteractionStyle. If using volume rendering, vtkRenderingVolumeOpenGL would also be required, and some of the text rendering used required vtkRenderingFreeTypeOpenGL.

If your application wants to take advantage of our integration with Qt then vtkGUISupportQt contains many of the standard classes, with vtkGUISupportQtOpenGL containing integration with QGLWidget and the Qt OpenGL library.

It is very important to ensure that you specify the components you want to use in the find_package call. If no components are specified, the VTK_LIBRARIES variable and the VTK_DEFINITIONS variables will be populated with all modules that were built. This is not usually what you want; you should instead specify the modules you will use in your application, and ensure that the compile definitions are used when compiling your application and that the libraries are linked to. A typical example would be one where you wish to find and link to the rendering code in VTK using the OpenGL implementation module. The CMake code in your application would look something like:

```cmake
cmake_minimum_required(VERSION 2.8.7 FATAL_ERROR)
project(vtkApplication)
find_package(VTK 6.0
  COMPONENTS vtkRenderingOpenGL vtkInteractionStyle
  NO_MODULE)
include(${VTK_USE_FILE})
add_executable(myApplication application.cxx)
target_link_libraries(myApplication
  ${VTK_LIBRARIES})
```

The inclusion of the VTK_USE_FILE is optional. If a user wishes to have more control, they may opt to replace it with the following calls (or similar):

```cmake
include_directories(${VTK_INCLUDE_DIRS})
set_property(DIRECTORY APPEND
  PROPERTY COMPILCE_DEFINITIONS ${VTK_DEFINITIONS})
```

The above is what the use file does, but it could be adapted so that include directories were set on the target directly, or the compiler definitions applied only to the application target in a more complex application. Including the use file sets several things at directory scope, which may not be desirable depending upon your build system and how your source tree is structured. It should also be noted that the aforementioned variables are populated locally with each invocation of the find_package command, and so subsequent calls could specify the components needed by another executable for example.

ACKNOWLEDGEMENTS

This release results from the effort of many different parties. Modularization was supported, principally, by the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research under award number DE-FC02-12ER26070, and by Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000. The final release work was sponsored primarily by NIH R01EB014955. As usual, the full list of sponsors and developers is long and diverse. Thanks go out to everyone who tried, contributed to, and funded the development of VTK 6.

Marcus D. Hanwell is a Technical Leader in the scientific computing team at Kitware, where he leads the Open Chemistry effort. He has a background in open source, open science, Physics, and Chemistry. He has worked in open-source for over a decade.

David E. DeMarle is a member of the R&D team at Kitware where he contributes to both ParaView and VTK. He frequently teaches Kitware’s professional development and training courses for these product applications.

DID YOU KNOW…?

Kitware’s Vimeo library is regularly updated with new videos and webinars to keep you up-to-date with the latest developments to our toolkits.

Visit vimeo.com/kitware to see what topics are available. Recent videos related this articles include:

**VTK 6.0 Release Webinar:** This 20-minute webinar highlights changes made in the VTK 6.0 release.

**Open-Source, Cross-Platform Charting with VTK:** This webinar teaches participants how to use VTK’s charting modules to plot and interact with non-spatially oriented datasets.

**ParaView 4.0 Release Webinar:** This short video shows what updates are new in ParaView 4.0, many of which are derived from VTK 6.
Tomographic reconstruction is at the core of medical imaging and has been an active field of research of the past fifty years. It aims at solving an inverse problem that requires the efficient implementation of algorithms based on mathematical solutions. A large part of the research is done by companies that not only protect their source code, but also the access to projection data. The cone-beam CT scanners installed in radiotherapy rooms are notable exceptions since access to raw X-ray transmission data is granted to the customers. This has triggered many developments in the past decade but, to our knowledge, there was no open-source software platform that would allow sharing of developments. The purpose of the Reconstruction Toolkit (RTK) is to answer this need.

PURPOSE
The RTK consortium aims at developing a community-supported open-source and cross-platform toolkit, RTK (http://www.openrtk.org), for fast tomographic reconstruction based on the Insight Toolkit (ITK). RTK is developed under the same Apache 2.0 license as ITK v4, approved by the Open Source Initiative. The RTK consortium is composed of CREATIS [1], the Massachusetts General Hospital [2], the Université Catholique de Louvain\footnote{3}, and IBA [4].

FEATURES
3D Circular Geometry: RTK has been developed for circular geometries, i.e., a source and a flat panel rotating along a circle. Deviations from this strict geometry can be described for each projection image by means of nine degrees of freedom: 3 coordinates for the point source position, 3 coordinates for the flat panel position, and 3 angles for the flat panel orientation. RTK provides classes to write and read the geometry in its own XML file format as well as readers for the geometry of commercial scanners, e.g., Elekta Synergy [5] and Varian OBI [6].

Input-Output: RTK uses the ITK factories for the input/ output of projection images. As for the geometry, we also provide new readers for Elekta Synergy and Varian OBI raw data (Figure 1). Simple conversion from raw data to linear attenuation is provided, and we aim to improve it in the future with calibration, beam hardening correction and scatter correction to reach quantitative CT, i.e., pixel values in Hounsfield units.

Forward- and Back-Projectors: At the core of tomographic reconstruction are forward- and back-projectors. They have been designed in a typical object-oriented fashion to allow testing of various implementations. GPU versions are also provided for faster reconstruction.
# Command lines used to generate these two images:
rtksimulatedgeometry --output geometry --nproj 720
rtkprojectgeometricphantom --geometry geometry \
--output proj.mha --phantomfile SheppLogan.txt \
--dimension 512
rtkfdk --path . --regexp proj.mha \
--geometry geometry \
--output fdk.mha --dimension 512 --spacing 0.5 \
--hardware cuda

## DEVELOPMENT TOOLS

RTK is developed in collaboration with Kitware and uses the software process suite: CMake, CDash, CTest, and CPack. The source repository is hosted on GitHub. The user's documentation is available on the main wiki and developer's documentation via Doxygen. Test and example data are hosted on MIDAS. The links to these tools are available on the RTK website.

## THIRD-PARTY SOFTWARE

**Insight Toolkit (ITK):** RTK is based on ITK [12] v3.20 or v4.3, which is its only mandatory dependency. Many ITK features are essential to the functioning of RTK: input-output factories, filters pipelining, streaming, mini-pipelines, multi-threading, smart pointers, FFTW wrapping, etc. Every RTK filter is meant to be ITK compliant and RTK can be thought of as an ITK module.

<table>
<thead>
<tr>
<th>Filter Description</th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDK ramp filtering (512³ projection images)</td>
<td>83s (Vnl) 17s (FFTW)</td>
<td>3s (CuFFT)</td>
</tr>
<tr>
<td>FDK backprojection (512³ projection images in 512³ volume)</td>
<td>110s</td>
<td>22s</td>
</tr>
<tr>
<td>FDK ramp filtering (512³ volume in 512³ projection images)</td>
<td>84s</td>
<td>17s</td>
</tr>
</tbody>
</table>

*Table 1: Multi-threaded CPU vs. CUDA GPU timing of several RTK filters. The measurement excludes hard drive input/output but includes the required transfers between the RAM and the graphics card. The tests have been performed with 48 threads on a cluster node equipped with two Intel Xeon E5-2630L CPUs and one Nvidia Tesla M2090 GPU.*

**Gengetopt:** RTK also provides a set of command-line applications that use Gengetopt to define their options (Figure 2). Gengetopt [13] is a tool for writing command line options for parsing code for C programs. The code is generated from a configuration file, named with a .ggo extension in RTK. We have developed the CMake module file FindGengetopt.cmake to automatically create the gengetopt compilation rules. Note that Gengetopt source code is distributed along with RTK, without modification, to automatically compile it if the Gengetopt binary is not available in the system path.

**GPU: CUDA and Open CL:** RTK proposes the use of a CUDA [14] implementation of a few filters that are essential to fast tomographic reconstruction: forward- and back-projection and ramp filtering. Part of the code has been taken and adapted from Plastimatch [15] and NiftyRec [16], two other open-source tomographic packages. It is worth noting that ITK is not compatible with the CUDA compiler, nvcc so the CUDA code is in separate functions defined in .cu file, which are not members of itk::ImageToImageFilter daughters. RTK also proposes the use of an OpenCL [17] implementation of one filter, but so far less effort has been put into using OpenCL. Table 1 compares the processing of filters that exist in multi-threaded CPU and CUDA/GPU versions.

## FUTURE WORKS

### New Developments:
At this time, RTK still has a limited number of capabilities compared to the numerous algorithms available in the literature. We will be complementing it with new forward- and back-projectors, iterative reconstruction algorithms, etc. Other geometries could also be added, including the handling of parallel beams, which is currently being developed. Other modalities could also be considered, such as nuclear imaging.

### External Use:
RTK is a software toolkit that is meant to be used in other medical computing and imaging platforms. We have already developed a proton CT reconstruction algorithm that uses RTK [18]. We will also use RTK in Gate [19] to enhance its X-ray simulation capabilities.

### IBA Cone-Beam CT Scanner:
RTK has been supported by the IBA company, a part of the consortium, which is currently developing the first cone-beam CT scanner developed for proton therapy rooms, in partnership with the Université Catholique de Louvain. RTK will be used in the scanner software for the reconstruction and the 2D/3D registration.

## CONCLUSIONS
We have developed a cone-beam CT reconstruction platform based on ITK. We aim at extending the current set of features with new preprocessing and reconstruction algorithms to improve image quality. We also want to handle other geometries and modalities. RTK is open to new users and developers and those who have made a significant contribution to the platform can join the RTK consortium. You can join the project by visiting http://openrtk.org.
**Simon Rit** received his PhD from the Lumière University (Lyon, France) in 2007 on respiratory motion correction in cone-beam CT for image-guided radiotherapy. His 2-year post-doctoral fellowship on the same topic at the Netherlands Cancer Institute (Amsterdam) led to the clinical use of motion-compensated cone-beam CT. Since 2010, he is a CNRS research associate at CREATIS (CNRS 5220, INSERM U1044, INSA Lyon, Université de Lyon) and the Léon Bérard cancer center.

**Marc Vila Oliva** received his Master’s degree on image/signal processing from the Universitat Politècnica de Catalunya (Barcelona, Spain) in 2011. Since 2012, he is a research engineer at CREATIS (CNRS 5220, INSERM U1044, INSA Lyon, Université de Lyon) and the Léon Bérard cancer center.

**Julien Jomier** is currently directing Kitware’s European subsidiary in Lyon, France where he focuses on European business development. He holds a Master’s degree in electrical engineering and information processing from the ESCPE-Lyon in France, and another in computer science from the University of North Carolina at Chapel Hill.

**REFERENCES**


**MOTIVATION**

Observations unequivocally show that the global climate is changing, caused over the past 50 years primarily by human-induced emissions of heat-trapping gases. The dramatic effects of climate change include increases in water temperature, reduced frost days, a rise in sea level, and reduced snow cover. As the pace of climate change is expected to accelerate, the impact of these changes will resonate through a broad range of fields including public health, infrastructure, water resources, and many others.

Long-term coordinated planning, funding, and action are required to reduce the impact on ecosystems, infrastructure, economies, and the global population. The wide variety of fields impacted by climate change means access to climate data and resources can no longer be restricted to scientists and researchers; other stakeholders must be involved.

Unfortunately, widespread use of climate data in non-climate science communities is impeded by factors such as large data size, lack of adequate metadata, poor documentation, and lack of sufficient computational and visualization resources.

ClimatePipes (Figure 1) aims to address many of these challenges by creating a platform that provides state-of-the-art, user-friendly data access, analysis, and visualization for climate and other relevant geospatial datasets, making the simulation data available to non-researchers, decision-makers, and other stakeholders. The overarching goals of ClimatePipes are:

1. Enable users to explore real-world questions related to climate change.
2. Provide tools for data access, analysis, and visualization.
3. Facilitate collaboration by enabling users to share datasets, workflows, and visualization.

![Figure 1: ClimatePipes interface showing cloudiness dataset in a geospatial context.](image)
ClimatePipes uses a web-based application platform due to its widespread support on mainstream operating systems, ease-of-use, and inherent collaboration support. The front-end of ClimatePipes uses HTML5 (WebGL, CSS3) to deliver state-of-the-art visualization and to provide a best-in-class user experience. The back-end of the ClimatePipes is built using the Visualization Toolkit (VTK), Climate Data Analysis Tools (CDAT), and other climate and geospatial data processing tools such as GDAL and PROJ4. In the next section, we will provide some more detail on the technology and relevant tools used for ClimatePipes infrastructure.

PROGRESS IN THE FIRST YEAR
We are currently one year into the project and much has already been achieved. The first year has concentrated on building the infrastructure and components that will provide the building blocks of the ClimatePipes platform. In this year, the focus has been on creating and improving the essential climate and geospatial data libraries, and evaluating state-of-the-art web-visualization and analysis techniques. Here are some of the highlights of the past year work.

KEY CLIMATEPIPES TECHNOLOGIES
We chose Python as our server-side language using CherryPy (http://www.cherrypy.org/) as the web server. CherryPy allows developers to build web applications in much the same way they would build any other object-oriented Python program. This results in smaller source code developed in less time. Although we are currently using CherryPy, we have designed our modules in such a way that in the future we could easily move to another Python-based web framework, providing the flexibility to evolve in the future. Using Python as the back-end language allowed us to leverage existing toolkits such as VTK and CDAT, which provide APIs in Python. On the front-end we are using HTML5, particularly WebGL for geo-spatial visualization. JQuery (http://jquery.com/) and Bootstrap (http://twitter.github.io/bootstrap/) are being used as two supporting frameworks to provide a consistent interactive cross-browser experience.

GEO-SPATIAL (GEO) VISUALIZATION LIBRARY USING WEBGL
In recent years, WebGL has gained popularity and wide support from different vendors and browsers. WebGL is a cross-platform, royalty-free web standard for a low-level 3D graphics API based on OpenGL ES interfaces. We chose WebGL as the API to render climate and other geo-spatial datasets because of its scalability, support for 2D and 3D formats, and accelerated graphics performance. We have developed a geospatial visualization library, named Geo, using WebGL and JavaScript. Geo provides support for rendering geometry (points, lines, polygons) and image data types. The library implements various features required for interactions such as picking, visibility toggling, and animations (see Figure 2).

Figure 2: Visualization of time-varying historical climate data via animations in ClimatePipes (powered by the Geo library).

VISUALIZATION OF CLIMATE DATASET IN A GEOSPATIAL CONTEXT
We have developed an infrastructure for providing high-resolution imagery as the geospatial context for climate data visualizations. The Tile-Based Geospatial Information System (TBGIS) supports panning, different levels of zoom, rotation in one axis (X-axis between 0-to-60 degrees), and different tiles sources. By panning and changing the zoom-level, users can easily find points of interest in the map. The rotation gives the possibility of 3D plots over the map. In addition, with tile sources, users may opt between different sources in order to have different visual representations of the map. Currently, it supports three tile sources: OpenStreetMap, MapQuest Map, and MapQuest Satellite; however, the implementation supports ability to add more new tile sources if needed. The algorithm calculates the number of tiles necessary to cover the entire canvas, and based on the zoom-level and the specified position (longitude and
latitude), it calculates and downloads the tiles around that region. Previously-downloaded tiles are cached to improve the rendering performance of the system. Figure 3 shows the tiles downloaded and rendered by the ClimatePipes system for a given position and zoom-level. Artificial spacing between tiles is introduced for clear visual separation.

DATA INTEGRATION WITH ESGF

One of the main features of ClimatePipes is to enable users to combine datasets from different sources. The Earth System Grid Federation (ESGF) is one such source. ESGF is a distributed data archive used to host climate datasets and associated metadata. We have integrated with ESGF by providing a query interface to enable searching of climate datasets in ESGF using a RESTful search API. The search is performed by the ClimatePipes back-end. The result set returned by ESGF can potentially be very large and the total search time can be significant. A streaming approach is used to maintain the interactivity, and to avoid having users wait for the entire search to be completed. The XML document containing the search result from ESGF is parsed and as each catalogue is received the result is streamed up to the client so users see the first documents very quickly; they can then interact with this data as the rest streams in.

DATA PROCESSING FOR VISUALIZATION AND ANALYSIS

Climate data in ClimatePipes is accessed through a VTK pipeline via a ClimatePipes VTK service module that exists within the CherryPy web server instance. This module contains a static pipeline that begins with a vtkNetCDFCFReader source. The reader opens and parses a given NetCDF CF convention format file, and produces VTK data structures corresponding to a chosen time step and variable selection. The client-side GUI interface is provided with a list of available files via a MongoDB database interface to a catalogue of available files with their relevant meta-data including temporal and spatial domains and the set of available attribute arrays.

The rest of the pipeline contains filters that prepare the data for transmission to the client. Currently, the VTK pipeline constructed is rather simple, but in the future it can be extended to do more server side data processing.

To deliver data to the client, we chose to use the simple and well-known GeoJSON format (http://www.geojson.org/geojson-spec.html). We then developed a server-side VTK writer and a client-side JavaScript GeoJSON importer. On the server-side, the new vtkGeoJSON writer instance sits at the end of the VTK pipeline and converts vtkPolyData into an in-memory GeoJSON representation that is suitable for delivery over the network.

On the client-side, a geojsonReader instance is created in response to a JQuery request to the server for a particular data file. The reader parses the data stream so that it may be visualized and otherwise processed on the client.

To display the data on the client, we populate JavaScript data structures that are suitable for WebGL rendering. The data structures and other rendering infrastructure are defined in the ClimatePipes VGL rendering library component that has also been developed in this first phase of the project.
INTERACTIVE WEB EDITOR FOR CLIMATEPIPES

ClimatePipes has an interactive workflow editor intended to provide easy creation of task-specific workflows. The implementation of workflow-required development of two main components: the JavaScript front-end used to display and interact (build/edit) with the workflows, and a Python workflow engine on the back-end to execute the workflow. New modules are created in the JSON data structure used by the JavaScript front-end. The front end was created using HTML5, JavaScript, and CSS. To execute the workflow, we created a lightweight workflow engine in Python, which simply executes the modules in order and passes data along. Alternatively, VisTrails (http://www.vistrails.org/index.php/Main_Page) can be used to execute the workflow if provenance, caching, and other features are desired. These workflows are automatically generated for the user after selecting data and a visualization. Users can bring up the workflow view and edit the workflow to perform more advanced changes.

CONCLUSION

Going into the second year, we are planning on improving integration between different components of ClimatePipes to provide a robust, scalable system at the end of Summer as version 1.0 of the system. Our particular area of focus in the next few months will be:

Data Integration: We will continue to integrate data sources such as DataOne and other data sources hosted by NASA, NOAA, and USGS.

Analysis: We will provide several simple but extremely useful climate data analysis algorithms to make it easier for users to comprehend climate variability such as yearly and decadal averages.

Workflows and Data Processing: The challenge is working with large datasets while maintaining the interactive user experience. We have several fronts to work on here, including the use of cloud computing and data streaming.

ACKNOWLEDGMENTS

We are thankful to Dr. Berk Geveci of Kitware and Dr. Claudio Silva of NYU-Poly for their support and guidance. We also appreciate Dean Williams and Charles Doutriaux of Lawrence Livermore National Laboratory for providing climate domain expertise and integration efforts with ESGF and UV-CDAT.

The work is performed under DOE SBIR Phase II (DE-SC0006493).

Aashish Chaudhary is an R&D Engineer on the Scientific Computing team at Kitware. Prior to joining Kitware, he developed a graphics engine and open-source tools for information and geo-visualization. Some of his interests are software engineering, rendering, and visualization.

David E. DeMarle is a member of the R&D team at Kitware where he contributes to both ParaView and VTK. He frequently teaches Kitware’s professional development and training courses for these product applications.

Ben Burnett is a senior software developer in the vgc research group at NYU Poly. He received his M.S. in Computing from the University of Utah where he focused on Data Management and Analysis.

Chris Harris is an R&D Engineer at Kitware. His background includes middleware development at IBM, and working on highly-specialized, high performance, mission critical systems.

Wendel Bezerra Silva received both his bachelor and masters degree at Universidade de Fortaleza (UNIFOR), working with real-time rendering on mobile devices. Currently, he’s a Ph.D. student at Polytechnic Institute of NYU (Poly-NYU), working with information visualization, large data and visualization for the masses.
We have written about the Open Chemistry project in a previous Source article [1], and MoleQueue [2] in a more recent article. Now we will focus on Avogadro 2, and the Avogadro libraries being developed to support the Open Chemistry [3] project. The project recently made its first release [4], tagging version 0.5.0 of all projects on April 11, 2013. Avogadro 2 is a rewrite of the Avogadro codebase, with a focus on scaling to larger problems in chemistry, molecular modeling, materials science, and bioinformatics.

The Avogadro paper describes Avogadro 1.x [5], with a particular focus on the work leading up to and including the 1.0 releases. Another, more recent article describing linking NWChem and Avogadro with the syntax and semantics of the Chemical Markup Language [6] describes collaborative work on an end-to-end solution in open computational chemistry in collaboration with the developers of NWChem [7] and FoX [8]. The diagram below shows the three core Open Chemistry projects, along with the framework development. This article will focus on Avogadro 2 and the libraries that power it.

A BRIEF HISTORY OF AVOGADRO
The Avogadro project was founded in 2006, with development really ramping up in 2007 bolstered by a strong collaboration with KDE on the Kalzium [9] project and a Google Summer of Code project to bring a molecular editor to Kalzium. The Avogadro 1.0 release was made in 2009, and a new beta series with many changes (1.1.0) in 2012. Kitware received SBIR funding in 2011 to develop a molecular workbench, with part of the funding allocated to rewriting Avogadro so that it was well positioned to tackle major research problems over the coming years, along with a HPC integration application (MoleQueue), and a cheminformatics application (MongoChem). It remains an open, community project with most of the current development happening at Kitware. We would like to form stronger ties with the wider community, and have worked hard to make a simple, extensible API that others can take and use in their own area of research, teaching, development, etc.

WHY REWRITE AVOGADRO?
One obvious question is why we decided to rewrite Avogadro, rather than incrementally update and improve it. It is always more difficult than anticipated to rewrite, and this project has been no exception. As we developed Avogadro over the years, we learned a great deal and also faced many new problems as the project grew. One of the major issues was that many of the data structures in Avogadro were written with quantum mechanics calculations in mind, which should come as no surprise as the application was initially developed to facilitate GAMESS calculations (predominantly ab-initio methods on small molecules in vacuum).

From those beginnings we generalized the interface in many ways, but were hitting scaling issues and having problems using Avogadro with larger datasets. We also found that some of the interface decisions were not ideal, and as a group we wanted to tackle a larger set of application areas than Avogadro initially set out to address. With that in mind, we have written a new set of core classes, and recently obtained permission from almost all contributors to relicense the code in Avogadro from GPLv2+ to BSD. The code still needs porting, but any issues over licensing have now been cleared up, and we will work on porting all major features over to the new codebase while ensuring testing, validation, and scalability remain a core focus. All of the new code is licensed under the three-clause BSD license, and is not only intended to power the Avogadro 2 application but also provide reference implementations that can be reviewed, tested, and reused in other applications and codes.

MODULAR DESIGN
One of the major complaints we would get from potential users of Avogadro was that it only had one library, and it contained all of the reusable API. Virtually all classes were derived from, or heavily used, Qt classes; even if you just wanted to reuse a few widgets there was a hard dependency on OpenGL and other components used by the rendering code. When developing the Avogadro 2 libraries, we reexamined the assumptions we made back then and added libraries with minimal dependencies, to allow for core components to be reused without any dependencies. The AvogadroCore library only depends on STL C++11 (with fallbacks to Boost where the compiler doesn’t support C++11). The AvogadroIO library
adds a dependency on HDF5, with the AvogadroRendering library using OpenGL and GLEW, and the Qt dependency is contained in the AvogadroQtOpenGL (QGLWidget integration). This allows many core components to be reused on the command line with minimal dependencies, and opens the door to integration with other GUI toolkits if you would like to reuse the rendering code.

The modular design allows for various components to be reused in a far wider variety of places than was possible before, such as using the file IO classes in command line applications to perform batch operations, or making use of the rendering components and file IO in client-server applications. There is also a lot to be said for a division of concerns within the library, where the rendering code is written to be largely concerned with rendering without directly handling the molecular data, and the scene created has only some degree of integration with the underlying data necessary to facilitate interaction and testing.

SOFTWARE PROCESS
It should come as no surprise that we have employed an adapted Kitware software process in the development of Avogadro 2, and the wider set of Open Chemistry projects. This involves the use of Git for distributed version control, Gerrit for online code review, CMake/CTest/CDash for cross-platform building and testing, and CDash@Home to test proposed changes. We also use a few things that other projects don’t make use of, such as Google’s testing framework for unit tests, the Qt Testing project from ParaView for regression tests, and image-based regression tests adapted from the VTK project.

TESTING
Testing is important, and ideally tests should be written at the same time (or even before) the code. When developing Avogadro 2 we have worked hard to ensure that code can be tested, and results verified. We have reused several components developed in other projects, such as the Google Testing library, image differencing developed for VTK’s testing infrastructure [10], and QtTesting developed to record and replay Qt events in ParaView. This has allowed for a much more thorough set of tests, from unit tests validating functionality in classes through to regression tests verifying drawing operations function as expected in a running instance of the application.

SEPARATE LIBRARIES AND APPLICATION
Avogadro 1 was written as a library and an application, but because they shared the same repository and build system, issues could slip by that would not work for an Avogadro-based application. When writing Avogadro 2, it was decided to make the application a dependent application, just like any other. This allows us to ensure that everything the application uses has been correctly exposed for external users, and to offer the Avogadro 2 application as an example of how one could use the libraries to develop a full-fledged application of their own. The use of CMake’s external project support, and a superbuild that takes care of coordinating the configuration, build, and installation of all related projects makes it easy for developers to get up-and-running while maintaining the level of separation desired.

SCALABLE DATA STRUCTURES
When considering the design of the core data structures in Avogadro we looked at the need for some data structures in the core that had no Qt dependency, and also at what data structures really needed to be QObject derived in the Qt dependent libraries. QObject derived classes carry quite a bit of overhead, and it was clear from our previous experiences that it was only really the Molecule class that needed to support concepts such as signals and slots. The atoms, bonds, basis sets and other components in the molecule could use lighter weight objects in their representation.

One major issue we wanted to get away from was heavy objects. We could have hundreds of thousands of objects containing optional parameters that must always be allocated; and by adopting a proxy flyweight pattern for most
small objects, we could move to a much more efficient core data structure, with simple objects that still feel familiar when accessing components such as atoms and bonds. We are continuing to work on improved data structures, with an emphasis on maintaining flexibility and scalability without placing undue burden on the users of these data structures. It was also important to create classes that could easily and efficiently be serialized/deserialized.

**UPDATED AND IMPROVED RENDERING**

We received a lot of positive feedback from the Avogadro community that the abstraction of rendering code was very useful, with plugins able to implement rendering code that could simply render a blue sphere at position x, y, z with a radius of r. The major problem with the approach we took was one of scalability; every frame we rendered used this API and drew each element one-by-one, which allowed the same rendering code to use OpenGL for interactive rendering and POV-Ray for ray-traced images. However, this approach also created a large bottleneck when attempting to render larger molecules. In Avogadro 2, we made use of a simple, specialized scene graph, which enabled us to maintain a simple API where developers of representations can simply specify sphere, cylinder, and triangle parameters without worrying about the underlying rendering code. It also allows the rendering code to store the scene, and greatly reduces the need to go back to the code that creates the representation. This easily lends itself to batching the rendering of spheres, and making use of efficient vertex buffer objects to store and render geometry.

Going beyond simple glyphing techniques for spheres and cylinders used previously, we have also implemented billboarded impostors using GLSL-based rendering code to perform simple calculations on the graphics card to determine sphere positions in the fragment shader and lighting calculations. This allows for a sphere to be represented using just two triangles, while still producing results that look more accurate than the typical explicit triangle geometry used in most glyphed representations. We hope to extend this approach in the future to other geometries, such as cylinders and cones, but the largest benefits are seen in spheres when considering typical molecule representations.

**LOCAL SOCKET COMMUNICATION**

The use of a JSON-RPC 2.0 based communication protocol facilitates loosely-coupled coordination of work between the three Open Chemistry projects, and provides simple entry points for other applications and scripted workflows to make use of Avogadro 2. The server was initially developed in MoleQueue, and has now been reused in Avogadro 2 and MongoChem. Named local sockets provide services that can be registered and used from the other applications; some simple functionality already exposed, in addition to the MoleQueue integration, is the opening of a molecule in MoleQueue, and finding similar molecules in the MongoChem database from Avogadro 2. MongoChem also reuses the 3D rendering and file IO from Avogadro in its dialogs, as you can see in the figure below.

**PYTHON-BASED INPUT GENERATORS**

We thought very carefully about how we could make input generators easier to add, modify, and integrate into Avogadro. Recent work has ported our input generators from Qt/C++ to Python, using the Python interpreter in a separate process, and using multiple passes to get the name in the interface, supported options and syntax highlighting rules, and the generated input. This has a number of advantages over the previous approach, including the use of a
The framework is largely language agnostic, although we have not exposed the ability to use other languages. As the framework relies on the QProcess class to manage the execution of a process, any language could be used for the input generators that are able to understand JSON and have text processing capabilities. A simple generator can offer no options and generate a fixed output, even taking advantage of text replacement features in Avogadro to insert the molecular geometry into the files. More complex input generators can of course take a large array of options, make use of Python modules, define rules for syntax highlighting, or even execute external processes themselves before producing output. As everything takes place in a separate process, this can all be done serially without any concern about locking the interface.

**INTEGRATION WITH COMPUTATIONAL CHEMISTRY CODES**

The new input generator framework allows users to easily run simulations using the generated input files. By staging and managing computational jobs through MoleQueue, local and remote resources can be used to perform calculations on molecules in Avogadro. This allows users to, for example, draw a molecule, choose a simulation code and parameters, perform the calculation on either the local machine or a remote HPC cluster, then load and visualize the results when the job finishes.

**CHEMICAL FILE FORMAT FRAMEWORK**

Dealing with chemical file formats can be difficult, and we have spent some time developing a flexible and extensible framework for file formats. We make use of a singleton class to manage registration and querying of available file formats, and a simple API to open and save files to different formats. This has been used for several core file formats such as the Chemical Markup Language, Chemical JSON, XYZ, MDL, and several quantum code output formats. We had several requirements up front to be able to seamlessly deal with files on disk, from over the network and from strings, and the goal of supporting composite file formats where data may not reside in the same file.

**QUANTUM CALCULATION OUTPUT**

The work that started in Avogadro in the surfaces plugin was later separated into a small library called OpenQube. When we started developing Avogadro 2, it became clear that it would be far easier to fold this functionality into the Avogadro libraries rather than to maintain it in a separate library so that it could effectively reuse many of the data structures developed for Avogadro. While moving it into Avogadro, it was also advantageous to remove the dependencies of the code on Qt and refactor the core classes so that there were data containers and algorithms that act on the data and produce output.

Not only does this make it possible to reuse the quantum data structures in a larger number of places, it opens up the possibility of using different parallelisation strategies for calculating the electronic structure properties. The original code was very tightly coupled using QtConcurrent, which may not always be the ideal choice depending upon application area and hardware/software environment available. This makes it far simpler to explore and implement alternative calculation strategies, including the use of GPGPU computing, OpenMP, and other parallelization technologies without forcing these dependencies.

**OPEN BABEL INTEGRATION**

Open Babel provided a great deal of functionality in Avogadro, and we didn’t want to abandon that in Avogadro 2. We also wanted to move to a more liberal BSD license, and get away from problems associated with the Open Babel design when used in a multithreaded application. We real-
IZED that a large amount of the functionality we were most interested in was translating from one file type to another, along with utility functionality such as 3D coordinate generation, simple geometry optimization, and the perception of bonding for organic molecules. This led to the development of an Open Babel extension that manages interaction with the command line obabel executable, which is able to take care of moving data in and out of the code. Not only did this allow us to make use of Open Babel in a separate process, it also simplified what our library needed to link to - allowing us to search for the executable at runtime rather than compile time. This makes it much easier to upgrade Open Babel independently of the application code when desired.

**CLIENT-SERVER COMMUNICATION**

ProtoCall is an asynchronous remote procedure call (RPC) framework that supports RPC using the Google Protocol Buffers library (protobuf). It is being developed to support client-server communication within Avogadro. However, it is a generic framework that can provide support for RPC in C++ for any application. Within Avogadro, ProtoCall is being used to enable the application to run in client-server mode to support computationally-intensive tasks that can be off-loaded onto a server, with the result being retrieved and visualized locally.

Services in ProtoCall are defined using the protobuf definition language in a .proto file, and a protobuf plugin is used to generate client and server stubs, see the figure above. The client code makes method calls on local classes and the framework takes care of marshalling requests and network communication. On the server-side, the implementer simply provides the service implementation by subclassing the server stub and registering it with the framework. The main idea is that ProtoCall hides most of the complexity of RPC from the user, making it look as simple as a local method call to consumers of the services provided.

The communication layer uses VTK, so the standard VTK data types are supported within ProtoCall messages. External data types can be supported by providing serialization/deserialization classes used to write and read instances of the data types to the wire. This is the approach we are using within Avogadro to transfer the base Molecule data type between the client and server, for example.

**FUTURE DEVELOPMENT**

As we look forward to the next six months of development we will focus on integration of the client-server functionality into Avogadro, providing deeper integration with VTK for advanced visualization and analysis; and making a richer desktop application capable of addressing more complex workflows that need multiple molecules loaded at any one time, and multiple widget types that perform specific editing, visualization and analysis tasks. We would also like to explore exposing some simple distance collaboration using the client-server code, and enriching the experience by exposing animation of molecular dynamics trajectories using accelerated views and methods of creating movies.

We are actively seeking collaborators as we look towards tackling new challenges. Please get in touch if you would like to discuss working with us in this area.

**ACKNOWLEDGEMENTS**

The majority of the work described here was funded by Engineering Research Development Center (W912HZ-12-C-0005), along with community contributions.

**REFERENCES**

[8] http://www1.gly.bris.ac.uk/~walker/FoX/
NEW AVOGADRO COLLABORATION FEATURED IN THE JOURNAL OF CHEMINFOMATICS

The Avogadro molecular editor, part of the Open Chemistry suite of tools being developed at Kitware, is part of a new collaborative project with NWChem and the FoX library which is featured in the Journal of Cheminformatics.

“From data to analysis: linking NWChem and Avogadro with the syntax and semantics of Chemical Markup Language” describes an end-to-end use of semantically-rich data in computational chemistry through the use of the Chemical Markup Language (CML) framework. In this paper, the semantically-rich data is generated by the NWChem computational chemistry software and the FoX library, and is used by Avogadro for analysis and visualization.

The article is featured in the same collection on Semantic Physical Science as the Avogadro paper published last summer, “Avogadro: An advanced semantic chemical editor, visualization, and analysis platform,” which has become one of the Journals most-read papers.

ITK COMMUNITY MEMBERS CONTRIBUTE FEDORA PACKAGE

In May, the Insight development team and community recognized the efforts of contributors Mario Ceresa and Peter Lemenkov for the creation of a new Fedora package for ITK. To create this package, the most complicated issue was ensuring that it complied with Fedora guidelines, which includes rules such as “no bundled libraries” and the inclusion of arch-dependent paths, etc.

As this applies to all required packages from the ground up, Mario and Peter looked through all the sources, carefully patched CMake files in order to make and install everything in accordance with the Fedora rules, decoupled libraries that were already available in the repositories, and included repositories that were missing bundled libraries. The “Insight Toolkit” package is available in the recently-released Fedora 19 distribution.

NEW INFOVIS TOOL ENABLES INTERACTIVE EXPLORATION OF CMS MEDICARE COST DATA

When the U.S. Centers for Medicare and Medicaid Services (CMS) released “Medicare Provider Charge Data,” a dataset of pricing information for the top 100 most-billed medical procedures in the country, the informatics team wanted to investigate the data. They developed a web-based, interactive exploration tool to enable further analysis and understanding of the data.

Kitware’s web application enables visualization of the data at the national or state level, and highlights individual hospitals. A publicly available dataset of mortality rates (from data.medicare.gov) for these same procedures has also been included, allowing visitors on the site to explore the relationship between cost, reimbursement, and mortality rate. Future enhancements and additional datasets are planned for the application, which we hope will spur further conversation and lead to new insight into the data.

Explore the data for yourself by visiting: http://infovis.kitware.com/hospital-costs/. This effort is sponsored by the Air Force Research Laboratory and DARPA XDATA program.

DOE PHASE II FUNDING AWARDED FOR AN HPC SIMULATION FRAMEWORK

Kitware has been awarded $1M in Phase II SBIR funding to further develop an open-source, high performance computing simulation and computation framework for the U.S. Department of Energy. The developed model-centric integration framework enables researchers to select only the toolkits and applications needed to perform a set of simula-
tion, computation, or analysis tasks. The resulting workflow will largely benefit small and mid-sized manufacturing firms that may not have access to HPC resources otherwise.

Phase I of this project led to the creation of the Simulation Model Based Architecture (SiMBA) framework for supporting the entire simulation lifecycle, the Simulation Modeling Toolkit (SMTK) which provides an abstract layer for the model geometry, and an attribute data manager for storing the necessary simulation information.

Phase II will focus on enhancing the current system, in addition to enabling the easy creation and providing support for the execution of applications in an HPC environment. The result will be a model-centric simulation framework for building simplified applications that are customized to a specific workflow. This work is supported by the U.S. Department of Energy under Contract No. DOE-SC0007615.

**PHASE II FUNDING AWARDED FOR AN IN SITU COSMOLOGY FRAMEWORK**

The Department of Energy has awarded Kitware with Phase II SBIR funding for the continued development of a framework for performing in situ analysis of large-scale cosmological simulations.

In collaboration with researchers at Argonne National Laboratory, Phase I saw the development of a lightweight in situ analysis framework and the successful coupling of the framework with the Hybrid/Hardware Accelerated Cosmology Cost (HACC), one of the core N-Body codes at the DOE. Completion of Phase I demonstrated the feasibility of the approach and highlighted the need for a data management component to enable direct comparison of simulations and observations.

With this Phase II award, the team will further develop the in situ library and in situ algorithms for identifying, extracting, and tracking features in large-scale cosmological simulations. The library will also be integrated with data management functionalities and web-based technologies, which will be developed to simplify data mining and visualization workflows. This work is supported by the U.S. Department of Energy under Contract No. DE-SC0007647.

**VTK ENHANCEMENTS FUNDED TO ACCELERATE MEDICAL INNOVATION**

Kitware has been awarded $1,484,180 from the National Institutes of Health for the enhancement and extension of the Visualization Toolkit (VTK).

In 1993, the textbook “The Visualization Toolkit: an Object-Oriented Approach to 3D Graphics” introduced Kitware’s flagship software, VTK. Since then, this open-source toolkit for 3D interactive visualization, image processing, and analysis has been adopted and deployed in a number of medical image analysis, bioinformatics, and clinical applications.

This new award comes after a number of requests from the VTK community to reinvigorate the toolkit as a catalyst for interactive visualization research in medicine. As such, this project will update the graphics infrastructure to support the representation and rendering of large data over the web, on mobile platforms, and with interactive 3D widgets. A VTK Extension Manager will also be deployed for easy discovery and download of extensions to VTK.

To fully understand the impact of the proposed refactoring and enhancements of VTK on the medical community, the team will work in close collaboration with the Virtual Lung Project at the University of North Carolina at Chapel Hill, the Center for Imaging Science at John Hopkins University, Information Sciences in Imaging Lab at Stanford, the National Association for Medical Image Computing lead by Brigham and Women’s Hospital, and the FARSIGHT project at the University of Houston. This work is supported by the National Institutes of Health under Contract No. 1R01EB014955-01A1.

**KITWARE IMPROVING TRAUMA ASSESSMENT PROCEDURES WITH SONOGRAMS**

The National Institutes of Health have awarded Phase I SBIR funding to Kitware to the development of hardware and image analysis algorithms that will allow novice ultrasound operators to perform life-saving Focused Assessment with Sonography for Trauma (FAST) procedures.

FAST procedures are rapid ultrasound examinations for identifying abdominal hemorrhaging, which is often associated with blunt abdominal trauma (BAT). Detection of this free fluid is done via ultrasound images of four specific abdominal areas. Patients with positive FAST results are in urgent need of treatment, whereas negative results indicate more stable patients.

In collaboration with InnerOptic, a leader in FDA-approved visualization and tracking technologies, Kitware will develop a low-cost system that visually-guides novice users through the acquisition of ultrasound images from the abdominal regions necessary for FAST procedures. The resulting system will be used to address equipment and training shortages in level 3 and 4 trauma centers, enabling them to reduce the morbidity and mortality associated with trauma, particularly BAT. This work is supported by the National Institutes of Health under Contract No. 1R43EB016621-01.
U.S. ARMY FUNDS A UNIVERSAL HEALTHCARE EXCHANGE LANGUAGE

Kitware is underway on a Phase I SBIR research project to develop an adaptive healthcare exchange language for interaction between patients and their healthcare organizations. The resulting system will improve the flow of information and knowledge between patients, healthcare providers, and researchers, and ultimately lead to better care, evidence-based medicine and diagnoses, and longitudinal research yielding medical breakthroughs.

The project team includes Tom Munnecke (Munnecke Enterprises), one of the original designers of both VistA and the Composite Healthcare System (CHCS), the healthcare information exchange systems for the U. S. Department of Veteran Affairs and the Department of Defense, respectively; Conor Dowling (Caregraf) a company that has developed a Semantic Web interface to both of these systems; and Kitware, a leader in advancing leading-edge technologies through open-source communities.

Together the team will create an open-source, patient-centric “sandbox” that uses existing VistA and CHCS data dictionaries to create a hands-on demonstration of a Semantic Web approach for providing a universal healthcare exchange language. The open-source nature of the system will allow for easy interaction with other open-source health initiatives, including the VA-launched Open Source Electronic Health Record Agent (OSEHRA). This work is supported by the US Army Medical Research and Materiel Command under Contract No. W81XWH-13-C-0042.

KITWARE RECOGNIZES BEST STUDENT PAPERS AT ISBI 2013

Kitware has been a regular sponsor of the International Symposium on Biomedical Imaging (ISBI), including for the Best Student Paper awards. The Best Student Paper awards are determined by a selection committee review of the top seven paper submissions to the conference. Out of this year’s 79 submissions, the committee determined the finalists based on paper originality, quality, and student contribution. Finalists then presented either a poster or oral presentation at the conference and then the three winners were selected.

This year the top three awards went to:

**Emily Dennis**, a graduate student researcher at the Laboratory of Neuro Imaging at UCLA, won for the paper “Development of the “Rich Club” in Brain Connectivity Networks from 438 Adolescents & Adults aged 12-30,” which was co-authored with Neda Jahanshad, Arthur Toga, Katie McMahon, Greig de Zubicaray, Ian Hickie, Margaret Wright, and Paul Thompson.

**Junhong Min**, a Ph.D. student studying optical microscopy in the Bio Imaging Signal Processing Lab at KAIST, was chosen for the paper “Continuous-Space Localization Using Sparsity Constraints for High-Density Super-Resolution Microscopy.”

**Nikhil Singh**, a Ph.D. student studying statistical shape analysis at SCI, was selected for the paper “A Vector Momenta Formulation of Diffeomorphisms for Improved Geodesic Regression and Atlas Construction,” which was co-authored by Jacob Hinkle, Sarang Joshi, and P. Thomas Fletcher.

BENDER MOVES THE VISIBLE MAN

Bender, Kitware’s new open-source toolkit for repositioning voxelized anatomical models, was introduced in late April. Bender is used to reposition 3D labelmap models for use in computing voxel-level Specific Absorption Rates (SAR), the rates at which energy is absorbed by the body when it is exposed to radiofrequency electromagnetic fields and ultrasound.

Most anatomical models are derived from MRI or CT images which are acquired with the subject laying on a narrow platform. With these imaging modalities it is not possible to acquire images where the subject is sitting or extending their arms. Bender solves this problem by allowing the pose of the subject’s anatomy in the acquired image to be changed.

Bender provides algorithms and a user-friendly application for rigging, skinning, posing, and volume resampling, enabling researchers to interactively reposition an anatomic labelmap into a desired pose. Upcoming developments to Bender will improve the anatomical repositioning by modeling tissue-specific deformation characteristics more precisely. This work is supported by the Air Force Research Laboratory under Contract No. D22009D10.
ITK WORKSHOP TOUR 2013

ITK team members Matt McCormick, Luis Ibáñez, and Xiaoxiao Liu took part in a traveling trilogy of events and hackathons.

MNI at McGill University

Their first stop was at the Montreal Neurology Institute (MNI) at McGill University, where they were hosted by Dr. Louis Collins and his lab at MNI.

The group at MNI consisted of 70 participants, and waiting list of 30 others due to limited seating! Attendees were given VirtualBox machines with ITK and tutorial materials pre-installed. They were then led through a hands-on image analysis examples using SimpleITK with IPython notebooks, and step-by-step basic image filtering and ITKv4 image registration C++ examples. Dr. Vladimir S. Fonov from the MNI lab also presented an example of IO with the MINC2 file format inside ITK. Support for the MINC2 file format was added in ITK 4.4.0 after a previous hackathon in Montreal and an epic number of patch revisions in Gerrit.

UNC Chapel Hill

The second stop on the tour was at the University of North Carolina at Chapel Hill, where they were hosted by Cory Quammen from the Computer Science department. More than 50 attendees took part in the workshop, most of them from UNC’s Biomedical Research Imaging Center, the Biostatistics department, and the Neuro Image Analysis and Research Lab; along with a small group from the Biomedical Engineering Department at Duke University.

As with the MNI event, the group was led through hands-on exercises and examples. After the workshop, the team was shown some cool new visualization demos, and got to experience the infamous virtual reality pit.

A productive hackathon took place the next day, during which Cory fixed bugs in the SCIFIO module used to read microscopy images. Matt worked on preparations for the release of ITK 4.4.0. Luis helped Pei Zhang submit his first patch, which allowed Pew-Tian Yap to follow soon after. Xiaoxiao helped Jian Cheng organize his code into an ITKv4 module.

Iowa University

Lastly the team traveled to Iowa University, where they were hosted by Dr. Hans Johnson. The forty-person crowd consisted of researchers and graduate students from the Biomedical Engineering, Psychiatry, Radiology, Computer Science, and Electrical Engineering departments. Many attendees were ITK beginners without C++ experience, so the workshop focused on learning to use SimpleITK. Luis came up with a gamification strategy by writing five challenging questions for each exercise and put them up in a simple scoring form on Google Doc to let people compete. This was a great success. Most people were able to progress well to the very end at their own pace.

The next day, the team met with Hans, his colleagues, and his students and had a productive time closing open issues, including a triage of JIRA issues. As you can see from the charts below, many issues were closed on that day (the 31th). We found many issues that had been completed, but not closed, were no longer relevant, or had no actions that could be taken on them in their current state. Many issues were closed by requesting the original submitter to provide more details and encouraging them to provide at least partial solutions to the problems they identified.

Kitware Hosts 2nd Annual Blood Drive

On June 28th, Kitware headquarters, located in Clifton Park, NY, hosted their 2nd annual Red Cross Blood Drive. The day started at 10 am, with eager employees and several community members ready and waiting to donate.
We were lucky enough to be hosting the drive on a sunny
day, which resulted in a steady and consistent turnout.
Overall the Kitware drive collected 18 pints of whole blood
and six pints of double-red blood.

Keep an eye out on our events calendar next Summer to
take part in our 3rd annual drive!

SPRING PROMOTIONS AT KITWARE
Several individuals were recently promoted on the Business
Development, Computer Vision, Scientific Computing,
Finance, and Communications teams at Kitware.

Dr. Berk Geveci has been promoted to Senior Director of
Scientific Computing. Dr. Geveci joined Kitware in 2000 and
now leads the scientific computing and informatics teams.

Dr. Brad Davis was promoted to Assistant Director of Business
Development, a new position at Kitware which focuses on
building customer relationships and seeking opportunities
for the company to grow.

Drs. Amitha Perera and Matt Turek were promoted to
Assistant Directors of Computer Vision. Both joined Kitware
as R&D Engineers in 2007, and have contributed significant
experience and technical leadership during their time with the
company.

Dr. Arslan Basharat was promoted to Technical Leader on
the Vision team. Dr. Basharat joined Kitware in 2009 as an
R&D Engineer. His promotion is in recognition of his leadership
and growth of the company’s video tracking expertise.

Dr. Sébastien Barré has been promoted to Financial Systems
Lead. Sébastien joined Kitware in 2001 as an R&D Engineer.
His promotion is in recognition of his being a driver of the
creation of Kitware’s internal timekeeping and billing
system, which has significantly improved the company’s
financial workflow.

Katie Osterdahl was promoted to Communications Lead
in recognition of her proposal management and leadership
skills. Katie joined Kitware in 2010 as Communications
Specialist.

UPCOMING EVENTS & CONFERENCES

Thermal & Fluids Analysis Workshop (TFAWS)
July 29-August 2nd; Central FL

Dave DeMarle will present an introductory, hands-on
ParaView tutorial.

MultiMat 2013
September 2-6th; San Francisco, CA

Kitware is sponsoring this conference, and Patrick O’Leary
will be attending.

2nd Annual OSEHRA Summit & Workshop
September 4-6th; Bethesda, MD

Wesley Turner is a Co-chair on the Tutorial committee, and
Rick Avila is a member of the Program Committee; Luis
Ibáñez will be attending.

ASME/FDA 1st Annual Frontiers in Medical Devices:
Applications of Computer Modeling and Simulation
September 11-13th; Washington, DC

Brad Davis is the Chair for the Population Modeling Track.

MICCAI 2013
September 22-26th; Nagoya, Japan

Kitwareans will award the Young Scientist Publication
Impact Award, present papers, assist with the conference as
part of the MICCAI Society board, and attend the 6th annual
workshop on Systems and Architectures for Computer-
Assisted Interventions.

NEW EMPLOYEES

Meredith Lapti
Meredith Lapti joined the Kitware team in Clifton Park, NY
as a Compliance Specialist. Before coming to Kitware, she
worked as a staff accountant where she was responsible for
conducting audits and reviews of non-profits, corporations,
and employee benefit plans; assisted with the filing of finan-
cial reports; and provided third-party administrative services.

Ilseo Kim
Ilseo Kim joined the Kitware team in Clifton Park, NY as
an R&D Engineer on the Computer Vision team. He holds
a Ph.D. from the Georgia Institute of Technology, where
he studied electrical and computer engineering. His techni-
cal interests include image pattern recognition, video and
image processing, image retrieval, machine learning, video
summarization, audio processing, and multimedia informa-
tion retrieval.

Sonia Ayme
Sonia Ayme joined the Kitware team in Lyon, France as a
Communications Project Manager for the growing European
office. Sonia brings a diverse set of skills to KEU, including
strategic communications such as media planning, public
relations, and social media. She also has experience man-
aging and organizing events and seminars, and analyzing
organizational costs for optimization.
Meriadeg Perrinel
Meriadeg Perrinel joined Kitware’s European office in Lyon, France as an R&D Engineer. He holds an engineering degree in digital imaging from ESIR-Rennes, which he received in 2010. Always on the watch for technological developments, his interests and knowledge are largely focused on developing 3D technologies.

INTERNS
Chun-Ming “Jimmy” Chen
Jimmy Chen joins Kitware as an intern on the scientific computing team in Clifton Park, NY. Jimmy is a Ph.D. candidate at Ohio State University, studying computer science and engineering.

Scott Spurlock
Scott Spurlock joined Kitware as an intern on the Computer Vision team in Clifton Park, NY. Scott is a current Ph.D. student and research assistant at the University of North Carolina at Charlotte, where he focuses on computer vision and machine learning topics.

Catherine Dumas
Catherine Dumas joins Kitware in Clifton Park, NY as an intern on the Medical Computing team. Catherine is a Ph.D. candidate studying informatics at the State University of New York at Albany, where she also teaches the informatics senior seminar and topics in information science classes.

Joshua Cope
Joshua Cope joins Kitware in Clifton Park, NY as an intern on the Medical Computing team. He currently studies information systems and technology at the State University of New York at Albany, and holds an A.S. in computer science from Fulton-Montgomery Community College.

Kristine Jessamy
Kristine Jessamy joined Kitware in Clifton Park, NY as an intern on the Medical Computing team. Kristine is a current student of information systems and technology at the State University of New York at Albany.

Christina Viccaro
Christina Viccaro joined Kitware in Clifton Park, NY as an intern on the Medical Computing team. Christin is a current student of computer science and mathematics at the State University of New York at Albany.

INTERNSHIP OPPORTUNITIES
Kitware internships provide college students with real-world, hands-on experiences while working with leaders in their fields.

Our interns assist with developing leading-edge, open-source technologies across our six business areas: supercomputing visualization, computer vision, medical imaging, data management, informatics, and quality software process. Please visit jobs.kitware.com to learn more and apply.

EMPLOYMENT OPPORTUNITIES
Kitware has an immediate need for talented software developers and researchers, especially those with experience in computer vision, scientific computing, informatics, and biomedical imaging. We offer comprehensive benefits including flex hours, six weeks paid time-off, a computer hardware budget, and more.

Qualified applicants will have the opportunity to work with leaders in their field. Please visit jobs.kitware.com to learn more about what it’s like working at Kitware, to view open positions, and to apply.

In addition to providing readers with updates on Kitware product development and news pertinent to the open source community, the Kitware Source delivers basic information on recent releases, upcoming changes and detailed technical articles related to Kitware’s open-source projects.

For an up-to-date list of Kitware’s projects and to learn about areas the company is expanding into, please visit the open source pages on the website at http://www.kitware.com/opensource/provensolutions.html.

A digital version of the Source is available in a blog format at http://www.kitware.com/source.

Kitware would like to encourage our active developer community to contribute to the Source. Contributions may include a technical article describing an enhancement you’ve made to a Kitware open-source project or successes/lessons learned via developing a product built upon one or more of Kitware’s open-source projects. The Kitware Source is published by Kitware, Inc., Clifton Park, New York.


Graphic Design: Steve Jordan
Editor: Katie Sharkey

This work is licensed under a Creative Commons Attribution 3.0 Unported License.

Kitware, ParaView, CMake, KiwiViewer and VolView are registered trademarks of Kitware, Inc. All other trademarks are property of their respective owners.