The Kitware Source contains articles related to the development of Kitware products in addition to software updates, news, and other content relevant to the open-source community. In this issue, Aashish Chaudhary, Nikhil Shetty, and Bill Sherman discuss using ParaView in immersive environments; Luis Ibáñez talks about teaching open-source; Kyle Lutz introduces ParaView's new query selection framework; Carlos Ruiz, Javier Velasco, and Diego Morate explain Ginkgo CADx; Marcus Hanwell describes code review and topic branches in VTK; Eran Swears presents his work in video analysis on business intelligence data; Charles Marion, Pat Marion, Luc Soler, Johan Moreau, and Julien Jomier introduce the Visible Patient; Julien Jomier and Jean-Christophe Fillion-Robin explore new additions to CDash; and Andrea Galeazzi explains how to make CMake play music.

The Source is part of a suite of products and services offered to assist developers in getting the most out of Kitware's open-source tools. Project websites include links to free resources such as: mailing lists, documentation, FAQs, and Wikis. Kitware supports its open-source projects with textbooks, consulting services, support contracts, and training courses. For more information, please visit www.kitware.com.

**PARAVIEW 3.14 RELEASED**

In late February, Kitware and the ParaView team released ParaView 3.14. This release features usability enhancements, improvements to the Plugin framework, new panels, and more than 100 other resolved issues.

ParaView 3.14 features a redesigned Find Data dialog and Color Editor. The updated Find Data dialog makes it possible to use complex queries to select elements, including combining multiple test cases with Boolean operations. The Color Editor, used to edit lookup tables or color tables for scalar mapping, now enables independent editing of the color and opacity functions.

ParaView’s charting capabilities have been extended with a new scatter plot matrix view and the ability to visualize multiple dimensions of data in one compact form. This improved view enables users to more easily discern patterns in small scatterplots and change the focus of the plots.

With this release, netcdf4 has been updated to 4.1.2, and ParaView can now handle large netcdf files that rely on hdf5 data storage.

ParaView showing various vehicle characteristics such as MPG, acceleration, and weight in a scatter plot matrix view.

There are several usability enhancements in ParaView 3.14. To simplify the process of working with many views simultaneously, ParaView now supports creating multiple tabs for placing views, which enables rapid switching of views.
The ubiquity of immersive systems is increasing. The availability of low-cost virtual reality (VR) systems [5] coupled with a growing population of researchers accustomed to newer interface styles makes this an opportune time to help domain science researchers cross the bridge to utilizing immersive interfaces. The next logical step is for scientists, engineers, doctors, etc. to incorporate immersive visualization into their exploration and analysis workflows; however, from past experience, we know that having access to equipment is not sufficient. There are also several software hurdles to overcome. The first hurdle is simply the lack of available usable software. Two other hurdles more specific to immersive visualization are: subdued immersion (e.g. the software might only provide a stereoscopic view, or might provide stereo with simple head tracking but no wand); and lack of integration with existing desktop workflows (e.g. a desktop computational fluid dynamics (CFD) workflow may require data conversions to operate with an immersive tool).

ParaView is a community based, multi-platform, open-source data analysis and visualization tool that scales from laptop to high performance supercomputers. ParaView has gained immense popularity amongst the scientific community as a universal visualization system. Early on we decided to integrate support for immersive environments directly into ParaView, rather than linking to an external library. We felt that there were sufficient components in ParaView's design that could be extended to support immersive rendering and interactions, and that limiting dependencies would make our efforts more impactful and sustainable for the long term.

**ARCHITECTURE**

The Visualization Toolkit (VTK) is a freely available, open-source software system for visualization, image processing, and computer graphics. However, VTK is not an end-user application. Using VTK requires writing code to produce interesting results, which may not be trivial to a general user. ParaView provides a user friendly environment via its graphical user interface, allowing researchers immediate opportunities for performing visualization tasks. At a very high level, the overall ParaView architecture is represented in Figure 1.

The Server Manager is the layer responsible for exchanging data and information between the client and the servers. As the name suggests, the Data Server is responsible for reading...
raw data, filtering, and writing out data which then can be rendered by the Render Server. It should be noted that the Data Server and the Render Server modules can exist either across a network, or together with the GUI client on a single machine. We determined the Render Server to be the apt place to drive Immersive displays. The typical use of the Render Server is for parallel rendering and image compositing; more broadly however, it is simply a framework where multiple views of same data can be rendered synchronously. The Client reads a configuration file and sets the appropriate display parameters on the Render Server. The parameters passed include coordinates of each display with reference to base coordinate of tracking system. A similar technique is used to pass device data to each of the participating nodes. The client reads the device data from a locally- or remotely-connected device and sends it across to the rendering nodes. Either the Virtual Reality Peripheral Network (VRPN)[1] or the Virtual Reality User Interface (VRUI) device Daemon[2] protocols can be used to collect the input data. Figure 2 demonstrates the client receiving head tracking data via VRPN from a remote tracking device and routing it to cameras on every Render Server.

IMPLEMENTATION

Off Axis Rendering
Like many computer graphics systems designed for desktop interaction, the VTK library calculates the perspective viewing matrix with the assumption that the viewer looks along an axis from the center of the image. This assumption is generally acceptable for desktop interfaces, but breaks down when applied to immersive interfaces where the view is affected by the location and movement of the head.

We added a new rendering option that allows a VTK-camera to be set to “OffAxisProjection” mode, opening up new parameters that can be used to precisely set the projection matrix. The key parameters required for this are the relationship between the eye and the screen. This could be done in relative terms, but with the goal of having this work in large immersive displays, it is more natural to specify screen and eye positions relative to a specific origin.

Therefore, the following new methods have been added to the VTK camera class:

- SetUseOffAxisProjection <bool>
- SetScreenBottomLeft <x> <y> <z>
- SetScreenBottomRight <x> <y> <z>
- SetScreenTopRight <x> <y> <z>
- SetEyePosition <x> <y> <z>

These are sufficient for monoscopic rendering, but most immersive displays operate stereoscopically, so we need to provide an eye separation and the direction of the head to properly adjust the location of each eye:

- SetEyeSeparation <distance>
- SetEyeTransformMatrix <homogeneous matrix>

With off-axis projection in place within VTK, adding immersive rendering to ParaView become a matter of configuring the screens and collecting and routing head tracking information to continuously set the proper rendering parameters.

Additionally, we want to provide an immersive user interface, so hand (or wand) tracking data is also typically collected. We can map that data through the ParaView proxy system to affect visualization tools.

Figure 2: Client distributing head tracking data to render server cameras

Figure 3: Stereo rendering with head tracking using VTK in 4-sided VR environment

Figure 4: ParaView running on a portable VR system

Interactor Styles
In addition to providing “OffAxisProjection,” we added support for new interactor styles via ParaView’s newly-added VR Plugin.

Figure 5: ParaView VR Device Plugin (a) VRPN Server or VRUI DD Connections are specified here. (b) Device Events and corresponding Interaction-Styles are specified here.
Paraview’s VR Plugin lets users perform operations such as collect events from VRPN and/or VRUI and dispatch these events to various interactor styles (see Figure 5).

A VR module typically needs to handle numerous types of input devices and perform various interactions on them. We offload the burden of managing devices to popular device servers like VRPN and VRUI and provide mechanisms to simply connect and read device information from them. As seen in Figure 5, we can handle any number of connections to any type of device servers. Each connection is an asynchronous thread that continuously polls the server for device data, converts this data to an internal representation, and pushes this converted data to an asynchronous event queue.

On the other end of the asynchronous queue is an event dispatcher, which dispatches incoming events to corresponding interactor styles. The VR plugin defines a limited set of Interactor styles, which have to be registered with the dispatcher. Interactor styles can specify which events they will handle and objects to be manipulated as a result of that event. The dispatcher has the additional responsibility of making sure older events are rejected and the latest events are passed along, so as to avoid any lag during interactions.

RESULTS
Adding immersion to ParaView uses existing VR tracking protocols (VRPN and VRUI) in conjunction with ParaView’s current capabilities for multi-screen rendering to enable immersive viewing and interaction. As with open-source packages released by Kitware, Immersive ParaView also relies on CMake for compilation of the application. We tested our additions to ParaView using several different immersive environments including a portable IQ-station[5] (a low-cost display based on commercial off-the-shelf parts), and a four-sided CAVE. The four-sided CAVE was tested using a single render server with two NVidia QuadroPlex’s.

FUTURE DIRECTIONS
In our current implementation, every render node for the VR system receives the geometry at the highest resolution. This is very inefficient as data generated via simulations could be extremely large. For VR systems, interactive user experiences require rendering at a minimum of 15 frames-per-second. Thus, we would like to have a level-of-detail strategy where we can adaptively change the details of the geometry depending on the view. We hope to leverage recent work on multi-resolution geometry by Los Almos National Laboratory and Kitware [4] for this purpose. Also, on certain occasions we have observed poor tracking feedback, especially when using VRUI, which is supported only on Linux and Mac platforms; this is an issue that we are hoping will be resolved in next release of ParaView.

The user interface for manipulating the visualization uses the ParaView proxy system, which presently lacks a good user interface for creating and altering proxies. A high-priority item is to provide an interface that allows users to specify proxy relationships from the main ParaView client interface.

CONCLUSION
Immersive ParaView makes strides towards a fully-featured, open-source visualization system that can leverage immersive technologies to serve broad scientific communities. It offers a low barrier to entry, drawing users to immersive displays through familiar software and migrating visualization sessions. As it attracts a user base and forms a community, we expect Immersive ParaView to become sustainable, gain support, and raise awareness and utilization of immersive displays. We ultimately aim to improve scientific workflows as low-cost VR systems, and have Immersive Paraview and other immersive visualization tools become integral components in scientists’ labs.

We would like to thank Idaho National Lab (INL) for their support of this effort.

REFERENCES

Aashish Chaudhary is an R&D Engineer in 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.

Nikhil Shetty is an R&D Engineer at Kitware and a Ph.D. student at University of Louisiana at Lafayette, where he also received his M.S. in Computer Science. Nikhil originally joined Kitware as an intern in May 2009 working on ParaView, and returned in May 2010 as a member of the Scientific Computing Group.

Bill Sherman is Senior Technical Advisor in the Advanced Visualization Lab at Indiana University. Sherman’s primary area of interest is in applying immersive technologies to scientific visualizations. He has been involved in visualization and virtual reality technologies for over 20 years, and has been involved in establishing several immersive research facilities.

TEACHING OPEN SOURCE
This year, for the first time during the Spring semester, we are teaching the Open Source Software Practices course at the Rensselaer Polytechnic Institute. The course has traditionally only been offered during the Fall. This is the sixth edition of the course and with 30 students registered, we had to move to a larger classroom after the second week.

This year we are emphasizing (a) Community management, (b) Economics, and (c) Applications of Open Source in Healthcare. One of the topics in particular is discussion of the importance of growing the size of a community to the right scale where the networking effects of the open-
source software peer-production model become effective. For example, only with sufficient participants do we get to benefit from the virtues of the Linus Law: “Given enough eyeballs, all bugs are shallow”.

Our estimation is that a project needs one developer per every 1,000 lines of code in order to remain healthy. For example, a project with 800,000 lines of code would require at least 800 developers to take care of it. These 800 developers will not be full-time dedicated developers of course; rather, they follow the typical power-log distribution in which 20% of the developers make 80% of the contributions, and the large majority of the developers make contributions very occasionally. Figure 1 shows this type of power-log distribution for the Insight Toolkit. The vertical axis is the number of commits, while the horizontal axis is the list of developers ranked by their number of commits. In this particular case, 10% of the developers have made 80% of the commits.

![Figure 1: Statistics of number of commits per developer in ITK](image)

It is common to make the mistake of thinking that this 10% is more important than the rest of the community, underestimate the role of the long-tail 90%, and ultimately fail to bring the project to the operating regime in which the network effects of open source pay off. This concept of a required critical mass of contributors leads us to insist in that the most important behavior of well-managed open-source projects should be the continuous recruitment of new developers, regular efforts for retaining existing well-behaved developers, and converting the not so well-behaved ones. To achieve this, the class regularly discusses the motivations that drive developers to make contributions to a project, and how project gardeners must ensure that those motivations are nurtured in the community. In particular, community gardeners must focus on the essential trilogy of: autonomy, mastery, and purpose.

The economic perspective is used again in this edition of the course to illuminate the basic properties of peer-production systems, and to clarify why open-source software projects are highly efficient at the production of knowledge-based goods and services. This perspective also provides context for the cost-benefit analysis in which the creation of monopolies, such as copyrights and patents, has to be balanced against the social and economic costs that they impose to society; particularly how they restrict the pool of resources available to the next generation of creators. With a background on the economics behind open-source and references to Kitware’s experience, the class covers the key elements needed to build successful business models including removal of barriers to the flow of knowledge and information, creation of collaboration platforms, focus on services and support, and agile programming processes that rely on rapid iterative cycles of development and customer interactions.

As a class project, the students are being introduced to VistA, the Veterans Health Information Systems and Technology Architecture, and are making contributions to the Code Convergence effort of the project. To this end, the students are learning the “M” language. This is a great teaching instrument because it has helped to put all students on a level playing field, where they are learning the language together and we can focus on the open source aspects of the project, instead of drifting into a pure programming class.

This semester we are also using “Moodle”, the Open Source Course Management System, for the second time. This has streamlined the execution of quizzes, assignments, and class activities into a single platform. As part of the practical exercises in the class, students are being introduced to the use of the distributed revision control system Git, the issue tracker MANTIS, the code review system Gerrit, and virtual machines. These are all part of the set of tools that create the communications and coordination infrastructures that allow large communities of contributors to work together to further the goals of a project.

The class also benefits from a diverse set of invited speakers including Conor Dowling (Caregraf), Marc Natter (Children’s Hospital, Indivo Project), Christy Collins (M-CM), Richard Fontana (Red Hat General Counsel), and David Wheeler (Institute for Defense Analysis).

The combination of practical software tool exercises, with invited speakers, and essential reading from authors such as Eric Raymond, Yochai Benkler, Lawrence Lessig, James Boyle, Steve Weber, and Larry Rosen, is intended to provide a balanced background with which the students will be well prepared to participate and promote the development of open-source projects and craft profitable business models around them.

**PRACTICE AND CODE CONVERGENCE**

Since the course started in 2007, one constant challenge has been identifying mechanisms for engaging students in real-life, direct experiences with open-source software projects. We initially expected that this aspect of the course was going to be trivial. The initial thought was to simply direct students to identify a project to participate in and we expected that involvement would rapidly follow after they knocked at the doors of any open source community. It turned out however, that in the initial attempts, the experiences of the students were widely diverse. Some of them started their own projects, some attempted to join small projects with communities of two-to-four developers, and others went after large established projects. Each one of those approaches led to the following drawbacks:

**Starting your own project.** This is the exciting option where students can craft the project to their preferences. However, in this case, what we observed is that students get to be too focused and too busy with the initial design and
coding of the project that they don’t get to be exposed to the variety of experiences that were sought in this exercise. In particular, they don’t get to interact with a community of fellow developers, nor a community of users; they don’t get to appreciate the full power of revision control as the mechanism for maintaining the technological and social history of the project; and they miss out on the opportunity to appreciate the importance of documentation, bug tracking, and software quality process. From all these limitations, probably the worst consequence is that they propagate the misconception that “development” is what matters most in software; that the final software can be a finished product and that open source is simply posting your code on-line once “you are done”. In reality, we know that design is only about 2% of the total cost of ownership (TCO) of a software project, writing the code is about 7%, and that the lion’s share is maintenance, and that accounts for 67% of TCO.

Joining a small project. This is the case where students attempted to join existing projects with a current crew of two-to-four developers. In these cases, we found that the attention of the developers was sporadic and there was not enough activity to expose the students to the full range of behaviors and technologies that the course was covering. These projects are typically more prone to a governance model of a “benevolent dictator,” which means that new comers are at the mercy of being aligned with the vision of the main developer, or to end up clashing with that vision.

Joining a large project. In these cases, students joined the developer community of a well-established project and started making contributions to the code. What we found is that many open-source projects have not done enough to facilitate and expedite the engagement of newcomers to their communities. This came as a revelation about our own projects; for example, in ITK, it taught us a lot about the ramps that we were missing to offer to new developers to facilitate their rapid and painless engagement in the community. It turns out again that many projects are so focused on the technology and the engineering of the code that they tend to forget the human and social aspect of their communities. Only a handful of projects have active initiatives for recruiting new developers and training them in the culture and practices of their community. This is a positive criticism to open-source projects, and one from which we learned a valuable lesson that we have applied to our own projects.

As a result of these experiences, recent editions of the course have used a more controlled approach in which the students are now required to participate in a large-scale, open-source project, and to engage in directed activities mentored by existing developers. In the particular case of the last two editions of the course, students have been working with VistA and making contributions to the community of the Open Source EHR (http://www.osehra.org).

There have been multiple advantages to the choice of this project: (a) the higher purpose of contributing to a project of great social and economic importance, (b) the participation in the engineering challenges of a project of large dimensions, and (c) a level playing field where all the students have to learn the “M” language and familiarize themselves with the particularities of the project.

On the front of learning the “M” language, students were exposed to two introductory lessons on M programming, and then worked together on writing an M tutorial using a distributed model of peer-production. The content of the tutorial is hosted in a publically-available Git repository (https://github.com/OSEHR/M-Tutorial), and the generated material is posted online for all users looking to learn M (http://www.opensourcesoftwarepractice.org/M-Tutorial).

Students are also making a transformative contribution for solving one of the critical challenges of the VistA community. The fact that the code base of VistA has been forked on multiple occasions by multiple organizations, and as a result the attention and focus of the larger community has been fractured and divided into maintaining each one of the individual forks. Given the large size of the VistA code base, about 2.5 million lines of M code, it is counterproductive to have a fractured community. Reunifying these communities and regrouping them around a single code base is one of the critical transformations that will lead to realizing the benefits of the open-source, peer-production economic model.

To this end, multiple organizations are working together on guiding the students through the exercise of analyzing the differences between the multiple VistA distributions, including: WorldVista, OpenVista, vxVista, and Medsphere. To facilitate this goal, Conor Downling of Caregraf has developed a semantic analysis of the multiple instances by taking advantage of VistA’s self-description capabilities (http://www.caregraf.org/semanticvista/analytics).

This system uses a query language to retrieve information describing the schema of the underlying database and the specific versions of the code patches that have been applied to the systems deployed in production. Students have compared source code routines between the various distributions, and become familiarized with the use of Git, the MANTIS issue tracker and the code review system Gerrit.

The participation in a large-scale project, under active mentorship of experienced developers provides a better guarantee of consistently exposing students to the social, cultural, and technological aspects of participating in an open-source software project. This indeed also prepares them better for one of the most likely activities that they will engage in once they join the job market.

Luis Ibáñez is a Technical Leader at Kitware, Inc. He is one of the main developers of the Insight Toolkit (ITK). Luis is a strong supporter of Open Access publishing and the verification of reproducibility in scientific publications.
With ParaView version 3.14 (released February 19th) comes a revamped data selection framework. The most prominent user-facing feature is the redesigned query dialog, which is available from the “Find Data” tool in ParaView’s Edit menu.

Data queries are entered by the user as Python expressions, which then act on the data and yield a selected subset of that data. The selected data can then be extracted and manipulated further by use of the Extract Selection filter just as before.

Selection for composite (e.g. multi-block) datasets is also supported by specifying which blocks or levels are to be examined. The query expression is then tested against each piece of the composite dataset to determine the selected points or cells. An example of selecting a single block in a multi-block dataset is shown in Figure 2.

The example in Figure 3 shows how to perform a threshold selection using the new query selection framework. Multiple queries can be joined together with boolean logic operators (e.g. and, or, not) allowing for specification of arbitrarily complex selections.

The new selection framework in ParaView 3.14 allows for data to be queried and examined in much more detail than in previous versions. Future developments will aim to support selection expressions for individual blocks of composite datasets.

Kyle Lutz joined Kitware in June 2011 as a member of the Scientific Visualization team. He holds a bachelor’s degree in Biological Sciences from the University of California at Santa Barbara. While in college he developed the chemkit library, an open-source software library for chemistry.
Ginkgo CADx [1] is a multi-platform, open core medical visualization and image-based diagnosis framework designed for specialists, practitioners, and physicians. Developed by MetaEmotion Healthcare [2], this framework provides an end-user friendly interface and extensible application for visualizing DICOM [3] studies, diagnoses, and creation of new DICOM studies.

The main input source for the application is DICOM series/studies, especially on non-radiology images based on specific workflows under IHE specifications.

Integration workflows provide formal and effective guidelines for the clinical pathway, which is the major gap we intend to fill. Without going into any technical details, a generic workflow could be summarized in five simple steps:

1. An appointment is scheduled or implicitly requested.
2. Ginkgo CADx generates or consumes the data.
3. Ginkgo CADx produces the output: Aided derived images, enhance, registration, automatic diagnose candidate or metadata.
4. The trial is reported.
5. The trial is stored and linked in the medical history.

Image integration is a mature deployment in radiology environments, where almost any image produced and consumed is integrated using HL7 and DICOM standards. Ginkgo CADx is willing to provide these workflows in other environments such as ophthalmology, dermatology, cardiology, pathological anatomy, etc.

We have developed an innovative workflow in ophthalmology diagnosis covering teleophthalmology workflows and specialist diagnosis. Advanced image processing algorithms are applied in image acquisition, including image stitching or the creation of synthetic images. The Ginkgo CADx visualization module has been extended to show all generated information in a simple way that focuses on ophthalmology.

Our image processing results are applied to dermatology samples to provide automatic skin segmentation and body surface area (BSA) calculations.

FEATURES
The project covers these main features:

- **DICOM visualization**: Multiple modalities support for neurological, radiological, dermatological, ophthalmological, ultrasound, endoscopy, electrocardiogram applications, amongst others.
- **DICOM communication with PACS servers**.
- **3D Reconstruction**: volume and surface reconstruction.
- A pluggable architecture.

ARCHITECTURE
Ginkgo CADx’s backbone is a wrapped ITK and VTK streaming I/O, and the processing pipeline is adapted to the medical environment workflow. Thanks to the streaming features supported in these libraries, the application can handle large DICOM datasets with low memory requirements. A graphical UI layer has been developed using the wxWidgets library, and the DICOM backend is a wrapped DCMTK subset.

The main focus in the framework design is to wrap complexity at many levels:

- Visualization
- Interaction
- Image processing
- DICOM format and protocol
- Interoperability
- GUI
- Tasking

As these tasks are deeply specialized in heterogeneous technologies, we have been focused on providing a soft wrap over them, as well as orchestrating components interaction.

CONCLUSIONS AND FUTURE WORK
Robust and standard workflows provide several improvements to diagnosis and reviewing outcomes availability; reduce the patient wait list, which results on better quality of diagnosis outcomes and economical refunds due unmissed trials; and avoid wasted time spent on tedious procedures.

The Ginkgo CADx project is constantly growing. We expect to cover more medical specialties and enhance the framework with architecture and technical improvements.

BUILDING
Ginkgo CADx takes advantage of the CMake build system. CMake helps us to scale our build with no effort on different platforms. CMake is a great tool that lets programmers save time spent on repeating common tasks; for example, library modularization and singular settings setup. Its dependency calculator and powerful control sentences are just brilliant.

The Ginkgo CADx project is hosted at Sourceforge and uses Launchpad to support the translations platform. It is being maintained by debian-med community, bringing the binary package into debian (currently in unstable) and ubuntu (latest stable) linux-based distributions.

ACKNOWLEDGEMENTS
We wish to thank the Nuestra Señora de Sonsoles Hospital at Avila (Spain) for their funding contribution and support.

We would also like to thank Kitware’s team for developing and bringing such quality technologies and knowledge to the community.
CODE REVIEW, TOPIC BRANCHES AND VTK

We began using Gerrit for code review in 2010 and a blog post from October detailed much of the early success in using it for the ITK project [1]. David Cole wrote about the new CDash@Home functionality being developed in CDash at the time [2], and we were able to integrate this with the event stream in Gerrit to trigger automated builds when a new proposed patch was pushed to ITK (provided the user was in a whitelisted group on the server).

We were reasonably happy with this support, but really wanted to be able to review topic branches rather than single commits. This workflow is described in the Source article “Distributed Version Control: The Future of History” [3]. Since then it has proven to be a successful method of developing and merging features when they are ready. It follows a similar pattern to that employed in many other projects, but was poorly supported in Gerrit. The general advice was to squash a topic into a single commit, review and test that, and then possibly merge the topic (assuming everything matches up).

MOTIVATION

One of the biggest issues in software development is balancing the need to develop and merge new functionality with the desire to keep the integration branch stable and functional. A commit from a developer on Linux can cause compilation failures for Windows developers, possibly halting their development until the branch is fixed. In the days of CVS and Subversion, this problem was especially pronounced; but with the advent of distributed version control, the problem is less of an issue when “branchy workflows” are employed. The developer can now simply branch off from before a breakage and continue working independent of activity in the development branch until they reach a point where they wish to merge.

Still, when branches are merged, there is a strong desire to ensure these fixes will not cause issues or introduce regressions. The Kitware software process has been developed over many years to mitigate these risks and detect regressions early by using a process of continuous dashboard submissions throughout the day, along with more thorough testing of the code each night. This ensures that breakages and regressions will be caught within hours, or by the next day if they are missed in the continuous submissions, while the changes are still fresh in the developer’s mind.

Git (and distributed version control in general) now make it very easy to push commits to multiple repositories, merging them only when they are ready; thus, the idea of pre-testing, where you could push a change to a repository and then tell CDash to build that branch, was born. The prototype worked quite well, but it was still a very manual process. By combining Git with Gerrit, we are able to build and test proposed changes and combine that with code review from expert developers. The event stream offered by Gerrit allows us to automatically submit CDash@Home build requests with minimal human interaction.

THE GERRIT EVENT STREAM

We initially developed a set of Python scripts for the ITK project for monitoring the Gerrit event stream and triggering various actions in response to events of interest, such as new proposed topics or patches. Python was chosen as a simple glue language that would allow the scripts to be updated easily. Python has good support for JSON (the data format used by Gerrit in its event stream), as well as HTTP PUT and POST methods used to interact with the CDash@Home web API.

Several Kitware developers have contributed to this code, and it has grown from a simple monolithic Python script to a pluggable Python process that is configured using a JSON configuration file. The Python script starts up and enters into a long-lived loop after connecting to Gerrit’s event stream using SSH. Events in the stream send new JSON objects into the event stream, which are then acted on according to the event handler functions.

CDASH@HOME

The CDash@Home web API is called from the Gerrit CDash broker, adding build requests for the proposed topic or change if the user who pushed the change is in the appropriate group. After some iterations, we settled on creating build requests with unique names and constructing URLs using the CDash query to filter out results referring to the submitted topic only. The broker script uses the command line SSH API supplied by Gerrit to add a comment to the patch with a link and details of the build.

CDash@Home volunteer machines run a simple CTest script in a loop, where they poll the CDash server for new jobs at specified intervals. The CDash server maintains a list of volunteer machines and dispatches the next available job to the volunteer that matches the specification supplied by the web API calls. The standard pattern requests a build of the proposed patch on Windows, Mac, and Linux. We have taken...
this a little further in VTK, and build with Python wrapping on for the Linux volunteers.

One of the bigger challenges in using CDash@Home is writing a generic script to execute the dashboard submission. This involves cloning the source tree and any data repositories, if they do not exist, and then setting up the build using data from the CDash machine XML file and the variable values supplied by the CDash server. Until recently, VTK has been using an experimental dashboard installation with fixes necessary for the topic branch testing; however, since the main dashboard was upgraded recently, they will be moving to the main VTK dashboard soon.

**GERRIT HOOKS**
The final piece of the puzzle was replacing the hooks on the stage that prevent bad changes from being merged. These are largely patch content checks looking at the sanity of each commit, such as author name and email address, preventing certain paths (such as kwsys) from being modified, and warning about whitespace issues. These checks are now done by a Kitware robot after a patch is proposed, and the verified field is used to approve or block commits.

**NEW VTK WORKFLOW**
Until recently, the topic stage was used to stage and merge topics into VTK’s master branch. This has now been replaced by Gerrit, and augmented with additional features provided by the Gerrit code review application. Anyone is free to sign up for a Gerrit account, and we use OpenID for authentication. Google is one of many OpenID providers that can be used with Gerrit. There are instructions on the VTK wiki to set up a new account and develop code [5].

**Developing New Features**
To get up-and-running, you will first need to install Git and clone the VTK repository,

```
git clone: git://vtk.org/VTK.git
```

Once you have a working clone of the repository, run the set up scripts. They will add the additional remote for Gerrit, along with some useful aliases to make development easier.

```
cd VTK
./Utilities/SetupForDevelopment.sh
```

From there, you can create a topic branch and develop as normal. Following the standard “branchy workflow” documented in several places, make commits as development proceeds. Once you think the topic is ready to be merged into master, upload it to Gerrit for code review and testing. This is done using a git alias set-up for you by earlier scripts.

```
git prepush    # View the commits to be pushed
git gerrit-push # Push the commits for review
```

The command will output a URL where the changes can be viewed. Follow that link to inspect the commits and ensure everything is as expected. Once you are happy with the changes, you can suggest reviews by typing in a person’s names and clicking on “add reviewer”. If you are in the VTK-Core group, a build should have been triggered and you will see a link posted by the Kitware robot to the filtered CDash results. Follow the link to check the build results.

**Reviewing Code**
Gerrit will email you if you have been added as a reviewer of a change. Anyone can give a review in the range of -1 to +1, with VTK developers being granted additional rights to approve a commit (+2 code review), or block a commit from merge (-2 code review). The Kitware robot uses the Verified field to block commits that do not pass basic content checks, or marks them as verified once the checks were successfully run. It will also warn about whitespace issues, but it is up to the reviewer to decide if that should block the submission of a topic.

As a reviewer, you should review the dashboard results and determine if there were any regressions or problems with the proposed topic; this includes compiler warnings, failing tests, and any configuration issues introduced. The CDash@Home builds are roughly equivalent to the Continuous submissions, covering the three major platforms (Linux, Mac, and Windows) in basic configurations. You must still check the nightly dashboards after a topic has been merged to check for regressions introduced by a commit on a much broader range of configurations and hardware.

Once you are happy with the dashboard results, review the code by looking at the individual commits in a topic and going through the diff. Gerrit allows for side-by-side or unified diffs, and both views offer the ability to make comments on any line of code. You should examine the code for any style issues, indentation, whitespace, etc., as well as logical issues with changes introduced and API decisions. This is the best opportunity to ensure any new API is a good fit, and to inform the developer of any problems you might spot. Any comments you make will remain drafts until you submit a review.

To submit a review, please use the scoring mechanism, which describes their meaning on the webpage. A score of -2 indicates that the commit is not suitable for merging into the code base, and must be changed before it can be integrated; this score blocks and does not allow submission. A score of -1 indicates that you see issues in the code, but other reviewers may choose to override your score and submit the change.
If you think the code is fine but feel that someone else should look before it is submitted, then +1 is appropriate. A score of +2 score approves the topic and enables the submit button.

Submitting Code: Merging

Once all commits in a topic have been reviewed and approved, clicking the topic name in any of the commits will take you to the topic view page. This page contains a summary of all commits in the topic and their scores; it is also the only place that can be used to merge a topic. After reviewing the topic summary, the same scoring mechanism is used to assess the entire topic. If you feel the topic is ready to be merged, giving it a +2 score makes it eligible and enable to be submitted. Once the submit change button is pressed Gerrit will attempt to create a merge commit to bring all the changes into the master branch. If there are conflicts, you will be asked to rebase the topic and submit again. Topics do not require that all commits have been approved, but it is generally a good idea to ensure they are.

If issues are seen with topics, the topic can be extended by adding more commits to the end and using the gerrit-push alias. Each time a topic is modified, a new build request will be made. We will also be rolling out an additional field called “Build Status,” which confirms a build has been requested. This will also allow members of the core developer group to request builds for other developers once they have reviewed the code submission.

CONCLUSIONS

The software process for VTK has been significantly augmented using several open source tools developed at Kitware and by others in the wider open source community. The online code review facilitates improved code quality and when coupled with CDash@Home integration, it represents a significant advance in our software process. We rely on skilled developers to review build results and code submissions, but aim to automate as much of the process as possible. We have also attempted to significantly reduce the barrier-to-entry for new contributors by providing a few levels where mentoring is encouraged.

We would appreciate your feedback as we continue to refine the process and roll it out for other projects. If you are interested in learning more about any of the components in the system, please get in touch; we have integrated components from several open source projects in order to achieve these results, and continue to assess its efficacy and impact on the development process.

REFERENCES


Marcus Hanwell is an R&D Engineer in the scientific computing team at Kitware. He is leading work on Open Chemistry, and has a background in open source, open science, Physics and Chemistry. He is a core developer of Avogadro, and has also made significant contributions to VTK, ParaView, Titan and software quality processes.

VIDEO ANALYSIS ON BUSINESS INTELLIGENCE, STUDIES IN COMPUTATIONAL INTELLIGENCE

Over a year ago, Dr. Fatih Porikli, the Associate Editor for the Springer Journal on Machine Vision Applications and the Springer Journal on Real-time Image and Video Processing, approached Kitware Vision about submitting a book chapter proposal on "Video Analysis on Business Intelligence". Dr. Fatih Porikli was looking for content relevant to the business intelligence aspects of video camera systems and was aware of Kitware’s expertise in the area of video analysis.

Although Kitware does not do much in the area of Business Intelligence, we came up with an idea related to the research we were doing on the Building Labeling in Urban Environments (BLUE) STTR DARPA sponsored program. BLUE’s goal is to combine movers and scene elements from video data to classify buildings by their surrounding activity as shown below.

Figure 1: Ocean City (OC) NJ video, where shops are mostly on the left side of the street.

Figure 2: Heat map showing all tracks for one day of OC webcam video; dark red indicates larger moving object bounding boxes while dark blue indicates smaller ones.
Our approach uses surveillance video to analyze normal pedestrian and vehicle behaviors around a set of store fronts in order to automatically analyze patterns that potentially impact a business. To accomplish this, we compared two functional recognition approaches and then integrated them into a more robust and accurate algorithm for improved analysis of business activities.

The first approach [1] uses supervised track-type classification for a person, driving-vehicle, parking-vehicle, and other classifications that are used as features in weak functional activity detectors to accumulate evidence for function element recognition; these include walkways, roadways, parking-spots, and doorways. The second approach [2] is a fully unsupervised method that identifies functional regions with similar behaviors by clustering histograms based on a trajectory level behavioral codebook. The likelihood maps from [1] are then incorporated into a clustering approach derived from [2] to produce more descriptive and consistent functional regions. The functional regions represent mixtures of functional elements, where one region can have a high concentration of a doorway element as well as a walkway element. Activity profiles can then be extracted from regions with a high concentration of the element of interest (doorway) in order to calculate business related statistics, i.e. number of pedestrians that pass by a store vs. enter/exit in a given scene.

The editors at Springer loved the proposed concept, and Kitware wrote the chapter entitled: “Automatic Activity Profile Generation from Detected Functional Regions for Video Scene Analysis,” with credits going to Eran Swears, Matthew Turek, Roderic Collins, Amitha Perera, and Anthony Hoogs. The book, titled “Video Analysis on Business Intelligence, Studies in Computational Intelligence,” will be published sometime this year by Springer, the leading global scientific publisher.

REFERENCES

Eran Swears is an R&D Engineer on the Computer Vision team at Kitware. He leads the Phase 2 BLUE SBIR and has been the principle researcher on several other DARPA efforts. He specializes in event/activity modeling and recognition using graphical models and probabilistic logic with applications to computer vision.

THE VISIBLE PATIENT

Kitware is pleased to take part in the Visible Patient project, which provides an open-access virtual reality experience for users worldwide. The project is led by the Research Institute against Digestive Cancer (IRCAD) in Strasbourg, France and Kitware assisted through the development and release of three 3D mobile viewing applications for web, Android, and iPhone users.

Inspired by the success of the Patient Specific Simulation and Preoperative Realistic Training (PASSPORT) project (http://www.passport-liver.eu), a surgical planning simulator for dynamic liver modeling, the Visible Patient provides surgeons with virtual 3D anatomies modeled on patient-specific data.

Users can easily access the Visible Patient through three different applications, depending on their device preference. The first solution is based on the latest generation of web browsers (Chrome, Firefox, Opera, and Safari), which are compatible with the Web-based Graphics Language (WebGL) standard. This standard is used on the www.visiblepatient.eu webpage to provide direct access to free, patient-specific 3D virtual anatomies.

The second solution is dedicated to Android 2.3 phones and tablet users. The Android application is based on Kitware’s VES library, and is freely available on the Visible Patient’s website and the Android Market. Similarly, the third solution is dedicated to iPhone and iPad users. This application is also based on Kitware’s VES library, can be run on iOS 5, and will be freely available on the Apple store.
THE VES LIBRARY

The Visual Patient applications for mobile devices are built upon Kitware’s powerful VES library (http://vtk.org/Wiki/VES) and are dedicated to phones and tablet users.

VES is the VTK OpenGL ES Rendering Toolkit. It is a C++ rendering library for mobile devices using OpenGL ES 2.0. VES integrates with the Visualization Toolkit (VTK) to deliver scientific and medical visualization capabilities to mobile application developers, and is licensed with the Apache License Version 2.0.

The online application is built upon the Midas Platform (www.midasplatform.org), Kitware’s open-source digital archiving system. Using the WebGL standard, this online application is supported by the latest web browsers. Three-dimensional meshes generated from the patient datasets are automatically converted by Midas to a highly compressed binary format in order to improve the data transfer to the client. By combining Midas with the open-source library, Threejs, the Visible Patient website presents a 3D, interactive discovery of the human body.

For more information on the Visible Patient, please visit www.visiblepatient.eu.

Charles Marion is an R&D Engineer at Kitware SAS in Lyon, France. He contributes to medical image, computer vision, and data publications, and is one of the lead developers of Midas, Kitware’s multimedia digital archiving system.

Pat Marion is an R&D Engineer currently working on the Scientific Visualization team at Kitware’s North Carolina office. He is leading the development of the VES library.

Luc Soler obtained his PhD in Computer Sciences in 1998. Since 1999, he is a research project manager in computer sciences and robotics at the Research Institute against Digestive Cancer (IRCAD, Strasbourg). His main areas of interest are medical image processing, 3D modelling, virtual and augmented reality, surgical robotics and abdominal anatomy.

Johan Moreau is an R&D Engineer in computer sciences at the Research Institute Against Digestive Cancer (IRCAD, Strasbourg). His main areas of interest are network and software engineering for computer-aided diagnosis and surgery. He is a contributor on the FW4SPL project (http://code.google.com/p/fw4spl/).

Julien Jomier is CEO of Kitware SAS, Kitware’s European subsidiary in Lyon, France. He is also the lead architect for CDash.

CDash aggregates, analyzes and displays the results of software testing processes submitted from clients around the world, conveying the state of a software system to continually improve its quality. This new release fixes more than 60 bugs and adds several new features, many of which are detailed in this article.

SIMPLIFIED LAYOUT

We could not have released a major version of CDash without improving the user interface. We strongly believe that a simple interface is essential for improving the efficiency of any web application. First, we have significantly reduced the size of the header and left more space for the actual content. We have also hidden columns by default, such as the timing report and sections of the dashboard that do not contain any builds. The goal is to provide developers with a quick and easy view of the current status of the dashboard. If issues are reported by the dashboard, users can switch to the “advanced view” and display more information.
We have also listened to feedback from the community and decided to implement an easier way to report timing. The build time is now displayed as the elapsed time and a mouse over tooltip shows the actual timing for configuration, compilation, and testing, expressed in a simple format.

We have also added an “Auto-refresh” button at the top of the main dashboard page. This function automatically refreshes the dashboard every minute and allows users to keep the dashboard in a tab on their web browser without having to refresh continuously. You will notice some other interface changes, such as the operating system icon associated with a build and other design features, which we hope you will enjoy.

On the main dashboard page listing all the projects, we have cleaned up the interface to allow for a better visibility, and added an activity icon which indicates how many builds on average are submitted per day. Low activity (one bar) will be reported for projects with less than 3 builds a day; a medium activity (two bars) for 3-to-10 builds per day; and a high activity for more than 10 builds a day.

**IMPROVED USER ADMINISTRATION**

CDash 2.0 integrates a new user administration page which allows for a better control of the users’ credentials and notifications. More specifically, project administrators can now set the notifications and repository credentials for every user. Repository credentials are used by CDash to make a direct link between the author of a source code modification in a repository (Git, SVN, CVS) and a user in CDash.

The uploaded files are stored on the CDash server within the directory specified using the `$CDASH_UPLOAD_DIRECTORY` variable. In addition to the global settings `$CDASH_MAX_UPLOAD_QUOTA`, it is also possible to set a project quota (in GB) limiting the space that is used to store the uploaded files. If the quota is exceeded, older files will be deleted to make room when new ones are uploaded.

```
$CDASH_UPLOAD_DIRECTORY = 'upload';
$CDASH_MAX_UPLOAD_QUOTA = '10';
```
Alternatively, URLs can also be uploaded to CDash. Indeed, if the uploaded file has the extension '.url', by convention CDash will assume its content is a valid URL and will display a link within the file view. When the file associated with a given build (usually packages) is uploaded to a different server (for example Midas), it is useful to upload the corresponding URL to CDash. This enables users to download the files associated with a given build directly from CDash.

```cpp
file(WRITE "${CMAKE_CURRENT_BINARY_DIR}/foo.url" "http://slicer.kitware.com/...")
ctest_upload
(files "${CMAKE_CURRENT_BINARY_DIR}/foo.url")
```

Figure 8: File view listing a URL allowing a user to download a package from Midas

The integration of CDash and Midas to support packages distribution will be discussed in a future Kitware blog post.

**NEW CONFIGURATION OPTIONS**

Several configuration options have been added to CDash. For system administrators managing a CDash server, it is always important to know how to configure CDash and obtain the best performance.

**Asynchronous Setup**

Asynchronous submission is not a new feature in CDash 2.0, but we have made it easier to set up CDash for asynchronous submissions. This feature is not necessarily needed if you have a server running only a few projects and your server is not overloaded. The main advantage of running CDash in asynchronous mode is that the clients will not have to wait for CDash to process their submission. CDash will copy the XML files generated by the clients and process them asynchronously when the server is not overloaded.

First you need to make sure that your web server has the php curl module enabled; you can usually verify via phpinfo(). Then you need to turn the `$CDASH_ASYNCHRONOUS_SUBMISSION` variable to true. Most of the web servers do not allow external browsing for security purposes, so you should keep `$CDASH_CURL_REQUEST_LOCALHOST` set to `true`. This means that CDash will trigger the asynchronous submission by calling a webpage at `http://localhost/$CDASH_CURL_LOCALHOST_PREFIX`.

```cpp
$CDASH_ASYNCHRONOUS_SUBMISSION = true;
$CDASH_CURL_REQUEST_LOCALHOST='1';
$CDASH_CURL_LOCALHOST_PREFIX='CDash';
```

**Base URL**

In CDash 2.0 we have added a new configuration variable called `CDASH_BASE_URL`, which allows users to set the base URL for CDash. This variable is not set by default and CDash tries to identify the current installation’s base URL automatically. This variable should be set when using URL rewriting.

```cpp
$CDASH_BASE_URL='http://my.cdashserver.org/CDash';
```

**Allow Users to Create New Projects**

We have added a new option for allowing CDash users to create new projects on a given CDash server. This option is set to ‘false’ by default to keep the previous behavior, but you can allow users other than administrators to create projects by switching it to true.

```cpp
$CDASH_USER_CREATE_PROJECTS = true;
```

**Log Rotation**

Before CDash 2.0, the log file in the backup directory was never deleted, potentially taking a lot of space. We have added a log rotation mechanism which allows users to cap the current log file to a certain size. CDash actually creates 10 rotating log files and even compresses them if possible. The size of the log file (in MB) is controlled by the following configuration variable:

```cpp
$CDASH_LOG_FILE_MAXSIZE_MB = 50;
```

We have also turned off by default the warning of the unregistered committers in order to avoid flooding the log files. This option can be turned back on as necessary.

```cpp
$CDASH_WARN_ABOUT_UNREGISTERED_COMMITTERS = '0';
```

**Registration Email**

CDash 2.0 now sends an email to newly registered users by default. This functionality allows users to make sure that the email address used for registration is valid:

```cpp
$CDASH_REGISTRATION_EMAIL_VERIFY = true;
```

**CONCLUSION**

CDash 2.0 brings a new design and several enhancements and bug fixes. We recommend system administrators to upgrade to the new release.

Moreover, if you would like to use CDash for your software project, Kitware hosts a CDash server that allows anyone to quickly setup a new project. This exciting, free service allows projects to be administered entirely from the CDash web interface. To start submitting to your own personal CDash, register at http://my.cdash.org, and create your project.

Julien Jomier is currently directing Kitware's European subsidiary in Lyon, France. He is also the creator and lead architect of CDash.

Jean-Christophe Fillion-Robin is a R&D Engineer, currently working at Kitware's North Carolina office. He is also one of the lead developers for Slicer.
After an “investigative phase,” I dove into implementation.

THE SYSTEM MIGRATION PROCESS

Korg Italy, the company where I work, is the leader in the production of electronic musical instruments. It was founded in 1996 as a subsidiary company of Korg Inc. (Tokyo). Its mission is very clear: use the latest technologies to create the finest electronic musical instruments and allow musicians to sound their very best. Such an approach to electronic and software technologies led us to CMake and so we started thinking whether it would be a good replacement for our old build system and work-flow. First, let me briefly explain the way we work and the tools we use.

PROPRIETARY HARDWARE PLATFORM & OS

We develop the whole instrument, from the embedded hardware design to the production of sounds. In particular, I’m a member of the software development team where we develop a proprietary, real-time OS. We have always put a lot of effort into writing easily portable code. This effort allows us to run our OS in an emulated environment on a host Windows machine, so we can develop and debug almost all of our code without needing to use a real hardware target. Furthermore, debugging the code on Windows is much easier than doing it on a custom hardware.

Before using CMake, a colleague of mine devised a sort of “hook” that was able to catch the commands between Visual Studio 6 and its compiler, and translate them in order to redirect them to the specific tool-chain of the target. This way we could use Visual Studio to build both the emulated version and the target one.

WHY WE CHOSE CMAKE

As you can imagine, the “hook” program is very complex and requires modification every time the team needs to change the IDE or the tool-chain. This additional work always discouraged us from changing or upgrading the IDE, so we started investigating a different ways of managing our whole software product line. The first thing we did was set some features that would make a build system suitable for our needs. Below I summarize the key-points:

- CMake is an easy way to manage multiple projects containing a lot of dependencies from other modules and libraries.
- It provides flexibility for changing the IDE and build system in a short time without having any problems with setting all the parameters every time.
- The system manages the cross-compilation as any other build system would, without needing particular customization except for the tool-chain specification.
- Developers can keep working with the same tools that they generally use, but they must be able to share the modifications made to the project using our version control system in a way that permits replication of any modifications to all others automatically.

Its cross-compilation support and ability to generate native projects for several build systems, makes CMake the ideal build candidate.

THE SYSTEM MIGRATION PROCESS

At that time we were developing the Pa3X: the arranger flagship among the Pa Series (Pa is an acronym for Professional Arranger). This product is ideal for testing the capability of CMake; being the top level of its series, the build system of Pa3X is the most complex project within our product line. If CMake worked well for Pa3X, it would work for the rest of our products; that being the case, I started to convert all our old projects to CMakeLists.txt.

Due to the huge number of projects and their configuration parameters, I decided to develop a program that can explore the root of our sources and create a CMakeLists.txt for each VS6 project found. Such a program is made up of two components: one is a parser of a Visual Studio 6 project, and the other is a “CMake persiter” that can serialize the project into a CMakeLists.txt according to CMake syntax.

Obviously this cannot be accomplished entirely with an automatic system, so I had to check each CMakeList.txt generated and modify them when needed. Fortunately, this allowed me to avoid the longest and most tedious part of the task.

As soon as I finished this work, I tried to generate my first Visual Studio 2008 (a very long step from VS6). After upgrading some old compiler options to the new ones, I finally completed a build of our OS! The next step was to integrate our ARM toolchain and generate a Makefile. This phase took just a bit longer than the previous one, mainly because of some odd linker configurations. After struggling with some problems of toolchain parameters, I was able to build the executable for our platform target too! It was a great satisfaction to listen to our arranger Pa3X playing.

The final step was to integrate the post build process, which deals with the creation of the packages available on the Korg website to upgrade the old operative system. I have to say, I was very impressed with how CMake allows for decoupling of the build of an executable and the post-build steps by adding several custom targets.

CMAKE MAKES THE CHANGES AS GRADUAL AS NECESSARY

Once I completed the migration on my workstation, one of our biggest concerns was the learning curve needed to get all the other members of the software development team comfortable working with it. That’s one of the most interesting things about CMake: when a project is set up, a developer needs only a little knowledge about CMake because their main activity is working with the native build
system or IDE. I implemented a batch script to generate all native projects, which the team needed in order to let every developer work with their own system. Now the only thing that a developer needs to remember is to add the new source files into CMakeList.txt.

Immediately after we started using CMake as our official build system, we received a request from some of our distributors asking if it’d be possible to make some software customizations focused on their specific market macro zone.

That was actually the first time we needed to change our product line relying on CMake, in order to fit the new requirements we faced. On this occasion I really appreciated the flexibility of CMake; all I had to do was to write a function with signature parameters matching the variation points that every customization needs.

In conclusion, the use of CMake allowed us to make our build system, and our product line in general, much more flexible and it improved our time-to-market. For all these reasons I became a dyed-in-the-wool CMake user!

Andrea Galeazzi graduated in Electronic Engineering at Università Politecnica delle Marche. Andrea works as software architect and build system manager in the R&D Software department at Korg Italy S.p.A.

KITWARE NEWS

ENHANCEMENTS FOR 3D SLICER FUNDED BY NIH

Kitware received grant funding from the U.S. National Center of Research Resources, part of the National Institutes of Health (NIH), to further develop 3D Slicer. Slicer is currently funded by grants associated with the National Alliance for Medical Image Computing (NA-MIC) and the Neuroimaging Analysis Center (NAC), and supported by extensive user and developer communities.

Dr. Stephen Aylward will be the Principal Investigator and Julien Finet will act as the lead architect and developer on this project. Together they will lead the collaboration between Kitware and the NIH-NCRR, Slicer, NA-MIC, and NAC communities to extend Slicer in three critical ways: with a simplified user interface, the implementation of fully-automated user interface testing, and the addition of SimpleITK access through Python.

The development of a simplified user interface will reduce the amount of training required for the most common tasks in Slicer. This effort will minimize the learning curve for end-users; simplify the development of extensions to the user interface for new algorithms; facilitate more frequent use of Slicer; and address time-consuming components to allow for more training time on advanced topics.

An automated user interface testing system will be achieved by adopting the GUI testing environment developed for ParaView. The testing will be run nightly using CTest and CDash regression testing and reporting systems. The nightly feedback will keep developers more informed about the overall quality of their software and the impact of their changes to the code. Additionally, the extensive collection of training material for Slicer will be automated as GUI tests. Those automated interactions with Slicer will be used for online courses and demonstrations.

Slicer will also be integrated with SimpleITK, a simplified protocol language interface built on top of ITK to facilitate rapid prototyping, education, and interpreted languages. This will make 3D Slicer’s Python interface a fully-functional, script-based image analysis and display toolkit that provides access to VTK’s visualization API and the Qt GUI API.

KITWARE 10 YEARS OF DEDICATION AWARDS

In December, Sébastien Barre and Luis Ibáñez were honored with awards for their 10 years of dedication to Kitware, which speaks to their commitment not only to the company but also to the greater open-source community.

Sébastien Barré joined Kitware on September 1, 2001 after completing his Ph.D. in Computer Science from the University of Poitiers (France). Sébastien’s introduction to Kitware came through VTK as one of the first European contributors to the open-source project. Since joining Kitware, Sébastien has been an invaluable member of the medical imaging and software development teams, and recently has focused on user interface development. He is the primary architect of the company’s internal back-office software, KWiK, a critical component of the daily workflow at Kitware.

Luis Ibáñez joined Kitware on January 29, 2002 and is now a Technical Leader on the medical imaging team. Luis earned his D.E.A. and Ph.D. degrees from the Université de Rennes I in France, where he worked on the segmentation of bone structures. He is a staunch supporter of Open Science and Open Culture, and his enthusiasm and dedication have helped build Kitware’s reputation as an open-source company. As one of the principal developers of the Insight Toolkit, Luis focuses on community outreach and coordination, which are critical to the success of large-scale open-source projects.

KITWARE COLLABORATES WITH NREL

Kitware recently began a new collaborative effort with the National Renewable Energy Laboratory (NREL) on implementing important back-end software processes for the Radiance Project.

Originally developed at the Lawrence Berkeley National Laboratory (LBNL) under DOE funding, Radiance provides lighting engineers and researchers worldwide with
physically-based simulation tools for daylighting, electric lighting and energy analysis of buildings. The project’s principal author was Greg Ward of Anyhere Software, who continues to provide oversight to the project’s evolution.

As part NREL’s efforts to incorporate Radiance-based simulation capabilities within OpenStudio, Kitware helped the team integrate CMake into their build process. The use of CMake for Radiance will allow for nightly building and testing of the Radiance HEAD release and will provide automatic updates of any detected build or testing errors.

Additionally, Kitware contributed a new Qt-based GUI for the Radiance Scene Viewer that runs on Windows, Mac and Linux operating systems. Both of these contributions enable a wider scope of researchers to access and use Radiance, either on their own or through OpenStudio. With these new implementations, NREL is encouraging more developers and users to take part in the Radiance project and contribute their own systems to the dashboard in order to create a diverse testing environment.

KIWIViewer and VES Source Code Released
In January, Kitware announced the source code release for VES and KiwiViewer. VES, the VTK OpenGL ES rendering toolkit, delivers scientific and medical visualization capabilities to mobile application developers. KiwiViewer is the interactive application built on VES for exploring geometric datasets on multi-touch mobile devices.

With the release of the source code, we are encouraging developers to get involved with the projects by downloading the source code, further extending or contributing to the software, submitting feedback or requesting new features through the VES UserVoice, and incorporating VES or KiwiViewer into their own projects. If you are a developer using VES or KiwiViewer for a project, we would love to hear about it and feature it on our website. We also encourage developers to send questions, bug reports, or feature requests to ves@public.kitware.com.

Open Chemistry Phase II Funding Awarded
In January, Kitware officially announced $730,000 in funding by the U.S. Army Engineering Research and Development Center (ERDC) for further development of a comprehensive, open-source computational chemistry workbench. Marcus Hanwell is the Principal Investigator, and will lead Kitware’s development of a suite of applications and libraries under the banner of Open Chemistry. Together the components will form an open, extensible application framework that puts computational tools, data, and domain-specific knowledge at the fingertips of chemists. The framework will leverage existing computational chemistry tools to perform calculations, while providing pre- and post-processing of data, information storage, indexing, and visualization.

These tools will empower chemists to work collaboratively and provide improved visualization, exploration of chemical structures and analysis of chemical data. Designed to be user-friendly, scalable, and network-aware, the applications will act as the central component in computational chemistry workflows. These features will also make Open Chemistry applications ideal for training future generations of chemists, as it will allow them to view, explore, and interact with chemical data in ways that are not yet possible.

For more information on Open Chemistry, please visit www.openchemistry.org, or contact us to discuss possible collaborative endeavors.

IGSTK User Group Meeting
In February the sixth annual IGSTK user group meeting was held in conjunction with SPIE Medical Imaging in San Diego, CA. This meeting is an opportunity for the IGSTK user and developer communities to come together and discuss updates on the toolkit and its use in the field. One of the highlights of the meeting this year was the new breakout session to gather community input and feedback on the toolkit. Participants were broken into smaller working groups to answer several questions pressing on the IGSTK team. Responses were then consolidated into a summary for three areas: new features, biggest hurdles and project promotion ideas. The full list can be seen on the Kitware blog, and to continue the discussion, please send feedback to the IGSTK mailing list.

American Football Plays Paper Presented
In January, Eran Swears presented a paper at the WACV conference in Breckenridge Colorado entitled: Learning and Recognizing Complex Multi-Agent Activities with Applications to American Football Plays. The paper, authored by Eran Swears and Anthony Hoogs, will be published in the Workshop IEEE Publication.

The basic research, which stems from the DARPA CARVE program, seeks to determine if pre-planned or scripted coordinated activities can be automatically detected using automatically computed and highly fragmented tracks from video. American football was chosen as a surrogate problem domain as it shares many of the difficulties of reconnaissance video, including difficult tracking, interacting agents (players), high variability within the same play type, and active deception.

The approach pushes the model complexity onto the observations by using a multi-variate kernel density while maintaining a simple HMM model. Temporal interactions of objects are captured by coupling the kernel observation distributions with a time-varying state-transition matrix, producing a Non-Stationary Kernel HMM (NSK-HMM). This modeling philosophy specifically addresses several issues that plague the more complex stationary models with simple observations, i.e. Dynamic Multi-Linked HMM (DML-
Orleans, Louisiana.

92nd American Meteorological Society Meeting in New Orleans, recently presented as part of the keynote address at the event. This research was run at Argonne National Laboratory. This research was conducted by a collaborative group from the National Center for Atmospheric Research, Sandia National Laboratories, Argonne National Laboratory, and Kitware.

In order to model such a complex system, the simulation code required 25 million core hours across 64,000 cores of Blue Gene/P, Intrepid, producing 0.25 simulated-years-per-day. The video depicts two months of the 27 month-long simulation. Using both the CAM5 physics and the highly-scalable spectral element dynamical core, this simulation of total precipitable water clearly shows developing hurricanes in both the Atlantic and Pacific oceans.

This research is important to the CCES, which aims to predict future climates by using scenarios of anthropogenic emissions and other changes resulting from U.S. energy policy decisions. It will be used to improve the scientific basis, accuracy and fidelity of climate models to address their impact on national policy.

### NA-MIC ALL-HANDS MEETING

Several members of the Kitware team attended the National Alliance for Medical Imaging Computing’s winter project week in Salt Lake City, UT. More than 100 attendees discussed the use of Slicer4 for traumatic brain injury assessment, image-guided surgery, head and neck radiation therapy planning, and atrial fibrillation analysis.

Julien Finet and Jean-Christophe Fillion-Robin presented the work done on Slicer4 and worked with users to get feedback to guide the development of new models.

Benjamin Long presented work on automated GUI testing using Qt and Python, which is derived from methods used by the ParaView team.

Stephen Aylward presented “NA-MIC Software Processes and Future Plans” to the External Advisory Board and representatives from the National Institutes of Health.

### UPCOMING CONFERENCES AND EVENTS

#### Big Data Conference

May 9 in Arlington, VA. Dr. Anthony Hoogs will present a talk entitled "Big Video Data: Getting Salient Content out of Many Pixels."

#### NA-MIC Summer Project Week

June 18-22 in Cambridge, MA. Kitware will participate in the hands-on research and development focused on neuroscience and image-guided therapy solutions.

#### IEEE Computer Vision Pattern Recognition (CVPR)

June 18-21 in Providence, RI. We will collaboratively present a day-long course titled "Python for MATLAB Users: Promoting Open Source Computer Vision Research."

#### Ter@Tec

June 27-28 in Paris, France. Will Schroeder will present a keynote address at the event.

#### Computer Assisted Radiology and Surgery (CARS)

June 27-30 in Pisa, Italy. Danielle Pace will present a paper entitled “TubeTK: An open-source toolkit of algorithms operating on images of tubes” as part of the Image Processing and Display session.

---

**KITWARE DEBUTS NEW EMPLOYMENT SITE**

Kitware is an exciting, interesting place to work. Not only is the company doing cutting-edge work with real-world impact, but the people and culture make it a vibrant workplace. To bolster our hiring efforts and showcase the cultural side of Kitware, we have developed a new employment website. The site features all of the unique reasons people love working for Kitware, from flex hours to Birthday Bagel Mondays to our open-door policy. There are also entertaining profiles of Kitwareans, to help give potential employees get an idea of who they would be working with, including several company founders. To see the site and our open positions, visit www.jobs.kitware.com.

**KITWARE PARTICIPATES IN INSCIGHT PODCASTS**

Kitware now sponsors and participates in the inSCIght podcast series. inSCIght is a weekly podcast that focuses on various forms of scientific computing, bringing expert panelists together in head-to-head discussions. Matt McCormick and Luis Luis Ibáñez volunteer as part of the inSCIght team, and other Kitware team members have participated as special guests in podcasts on topics including version control, and other Kitware team members have participated in head-to-head discussions. Matt McCormick and Luis Luis Ibáñez volunteer as part of the inSCIght team, and other Kitware team members have participated as special guests in podcasts on topics including version control, open access, clean code and cross-platform build tools. To listen to previous podcasts and follow inSCIght, visit their website at www.inscight.org.

**PARAVIEW & ATMOSPHERIC CLIMATE MODELING**

Kitware participated in the development of the animation “NSF/DOE Community Atmospheric Model: Total Precipitable Water” for the Climate Science Computational End Station (CCES). The video shows results from a CAM simulation that was run at Argonne National Laboratory. This research was recently presented as part of the keynote address at the 92nd American Meteorological Society Meeting in New Orleans, Louisiana.

The animation of the simulation results was created by Joseph Insley of Argonne, with help from Kitware’s Andy Bauer, using ParaView. This research project is being conducted by a collaborative group from the National Center for Atmospheric Research, Sandia National Laboratories, Argonne National Laboratory, and Kitware.

---

**HMM and the Time-Delayed Probabilistic Graphical Model (TDPGM). These include: smaller training datasets, sensitivity to intra class variability and/or dense uninformative clutter tracks. Experiments are performed in the American football video domain, where the offensive plays are the activities. Comparisons are made to the DML-HMM and an extension of the TDPGM to DBNs (TDDBN). The NSK-HMM achieves a 57.7% classification accuracy across seven activities, while the DML-HMM is 26.7% and the TDDBN is 21.3%. When tested on four activities the NSK-HMM achieves a 76.8% accuracy.**

---

**Open Source Computer Vision Research.”**
IEEE International Symposium on Biomedical Imaging
May 2-5 in Barcelona, Spain. Luis Ibáñez will present a tutorial entitled “Using ITKv4 and VTK in Biological Imaging.”

KDE Akademy
June 30-July 6 in Tallinn, Estonia. Will Schroeder will present a keynote speech at the annual KDE summit.

OSCON 2012
July 16-20 in Portland, OR. Luis Ibáñez will attend OSCON. We anticipate that Kitware will present a talk and a tutorial.

SciPy 2012
July 16-21 in Austin, TX. Matthew McCormick is co-chairing the program committee for the 11th annual Scientific Computing with Python conference.

NEW EMPLOYEES

Johan Andruejol
Johan Andruejol joined the medical team at Kitware’s North Carolina office in February. Johan completed a year-long internship with Kitware before joining the team full time. He is currently in his fifth year at CPE Lyon, where he is focused on image synthesis and processing.

Christie Kil
Christie Kil joined the computer vision team as an annotation specialist at the Clifton Park office in February. Christie earned her M.S. in political science and American studies from Seoul National University, and her M.S. in international affairs from the Georgia Institute of Technology.

Leo Liu
Leo Liu joined the Kitware team at the North Carolina office as an R&D Engineer on the scientific computing team. Leo earned his B.A. in math and computer science from Wittenberg University and his Ph.D. in computer science from the University of North Carolina at Chapel Hill.

INTERNSHIP OPPORTUNITIES

Kitware Internships provide current college students with the opportunity to gain hands-on experience working with leaders in their fields on cutting edge problems in a unique open source environment.

Our interns assist in developing foundational research and leading-edge technology across six business areas: scientific computing, computer vision, medical imaging, data management, informatics and quality software process. To apply, send your resume to internships@kitware.com.

EMPLOYMENT OPPORTUNITIES

Kitware is seeking talented, motivated and creative individuals to fill open positions in our Clifton Park, NY; Carrboro, NC; and Lyon, France locations. As one of the fastest growing companies in the country, we have an immediate need for software developers and researchers, especially those with experience in computer vision, scientific computing and medical imaging.

At Kitware, you will work on cutting-edge research problems alongside experts in the field. Our open source business model means that your impact goes far beyond Kitware as you become part of the worldwide communities surrounding our projects.

Kitware employees are passionate and dedicated to innovative open-source solutions. They enjoy a collaborative work environment that empowers them to pursue new opportunities and challenge the status quo with new ideas. In addition to providing an excellent workplace, we offer comprehensive benefits including: flexible hours; six weeks paid time off; a computer hardware budget; 401(k); health, vision, dental and life insurance; short- and long-term disability; privacy processing; a generous compensation plan; profit sharing; and free drinks and snacks.

Interested applicants are encouraged to send their cover letter and resume to jobs@kitware.com to ensure their immediate consideration.

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, including:

- The Visualization Toolkit (www.vtk.org)
- The Insight Segmentation and Registration Toolkit (www.itk.org)
- ParaView (www.paraview.org)
- The Image Guided Surgery Toolkit (www.igstk.org)
- CMake (www.cmake.org)
- CDash (www.cdash.org)
- Midas (www.midasplatform.org)
- BatchMake (www.batchmake.org)
- KiwiViewer (www.kiwiviewer.org)
- Visomics (www.visomics.org)
- OpenChemistry (www.openchemistry.org)
- Slicer (www.slicer.org)

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 success stories learned via developing a product built upon one or more of Kitware’s open-source projects. Kitware’s Software Developer’s Quarterly is published by Kitware, Inc., Clifton Park, New York.


Graphic Design: Steve Jordan
Editors: Katie Sharkey, Katie Osterdahl

To contribute to Kitware’s open-source dialogue in future editions, or for more information on contributing to specific projects, please contact us at editor@kitware.com.