In the past few months, we have made many new developments that we are pleased to share with you. We released new versions of Avogadro and MoleQueue, which are part of the Open Chemistry project. We also released ITK 4.5.

In addition, we attended numerous conferences and events, including TEDxAlbany: Innovation, Design, and Emerging Alliances in Surgery (IDEAS): Virtual Surgery; Neurosurgery Simulation Symposium; RSNA 2013; Supercomputing 2013; and the IEEE International Conference on Computer Vision. More details on these events can be found in Kitware News at www.kitware.com/news.

For our full calendar of events, please see the Kitware events page at www.kitware.com/events. If you would like to set up a time to meet with us at any of our upcoming events to discuss employment opportunities, potential collaboration, or consulting services, please contact us at kitware@kitware.com. We would be happy to arrange a meeting!

**ITK 4.5.0 RELEASED**

The ITK 4.5 release is a major milestone that marks the hard work of many outstanding community members. Among the major contributions in this release are registrations computed with 32-bit floats, a new edition of the Software Guide, the first release of the wiki examples tarball, and the first release of the Sphinx examples tarballs. New Remote Modules include MGHIO, an Image reader for Freesurfer formats, and SmoothingRecursiveYvvGaussianFilter, a fast smoothing Gaussian image filter. There are also many more new features and plenty of bug fixes. More details can be found in the release announcement. Links to the tarballs are on the itk.org Resources Download page.

**VERSION 0.7.2 OF AVOGADRO 2 RELEASED**

In December, Kitware announced the release of version 0.7.2 of the Avogadro 2 application and libraries. The new version’s source and binaries are available on Open Chemistry’s website or on SourceForge. The release features bug fixes for version 0.7.1 of Avogadro 2, which was also released in December. The bug fixes include a Python interpreter path fix and improvements regarding interaction with MoleQueue when submitting jobs.
MOLEQUEUE 0.7.1 RELEASED
Kitware announced the release of MoleQueue 0.7.1 in December. MoleQueue is a part of the Open Chemistry project, which was developed to integrate computational chemistry codes on desktop and high-performance computing (HPC) resources. The release of MoleQueue 0.7.1 features several bug fixes regarding subjects such as the loss of program entries in queues and the state of queued jobs in some circumstances.

MoleQueue has been developed as a standalone, system-tray resident application that runs a graphical application and a local server. It uses local sockets for communication and is powered by Qt. MoleQueue supports the configuration of multiple queues, including local and remote.

PARAVIEW 4.1.0 RELEASED
ParaView 4.1.0 is now available. The new version of ParaView introduces a completely redesigned panel for editing color and opacity transfer functions for scalar coloring. The panel, which is now a dockable widget, makes it easier to edit transfer functions, labels, and annotations on the color legend (or scalar bar).

In addition, Find Data dialog, which was introduced several versions ago, has quickly become one of the main mechanisms for creating new selections. Until now, Find Data dialog could not be used to inspect a selection that was created by a user through other mechanisms. For the new release, Find Data dialog has been updated to always show the active selection, no matter how it was created.

Several new editions of ParaView Catalyst are now available for download and are routinely built and tested. These editions are:
- Base: minimal set of dependencies needed for Catalyst
- Base-Python: Base with Python support
- Essentials: Base and dependencies needed for Calculator, Contour, Clip, Cut, and Glyph filters
- Essentials-Python: Essentials with Python support
- Extras: Essentials and dependencies needed for ExtractHistogram, WarpScalar, WarpVector, Integrate-Attributes, and Extract Surface filters
- Extras-Python: Extras with Python support

Furthermore, ParaViewWeb is now part of the ParaView binaries. ParaViewWeb is a framework for building custom web applications. Before it was included in the binaries, ParaViewWeb underwent a major redesign to ease its deployment and usage. As part of the redesign, the external dependency and strenuous deployment setup were eliminated. In addition, all of ParaViewWeb’s core components were moved to VTK, which now has web capabilities.

Significant community contributions for this release include a completely redesigned, feature-rich, and parallel enabled line integral convolution (LIC) from Lawrence Berkeley National Laboratory (LBNL).

CMAKE, CTEST, AND CDASH AT NETFLIX
By Pedro Navarro

At the Core Technologies team at Netflix, we develop the application framework and streaming engine used by millions of consumer electronics devices, game consoles, tablets, and phones. With such a varied ecosystem is that in-house ports of our libraries (mainly game consoles) have very specific requirements that, more often than not, require heavy patching, usage of custom non-POSIX APIs, and linking against platform specific libraries. All those changes could not come back to us, as our build system was not capable of handling them, so the game console teams had to apply their patches and resolve the conflicts every time we released a new version of the SDK.

We set out to find a solution to this problem and defined our requirements for our perfect build system:

- All platforms should build from the same source tree, so all could benefit from bug fixes and new features just by getting the latest version of our code. At Netflix, we develop for consumer electronic devices (mostly Linux-based), game consoles (Windows-based, using different versions of Visual Studio with a mix of gcc and non-gcc compilers), iOS devices (using OS X and XCode), and Android. Therefore, our build tool had to handle as many of those environments as possible.
- We need to perform system introspection to detect compiler and platform features and library availability and version (in case we have to enforce a specific one).
- Integrated changes from one platform might break another, so the build tool had to be flexible and let us specify different configurations at build time.
- Ideally, the selected tool would be open source so we could customize it if needed. As Netflix is a very active Open Source believer and contributor, we would make any changes available.
- Our Jenkins build farm is as varied as the platforms for which we develop, so we wanted to minimize the number of system dependencies (like Python or bash) our chosen build tool had.
We found that CMake was the tool that better fit our needs: It created project files for all development environments we used, was easily extensible with its own scripting language, provided cross-platform commands to copy and delete files and directories, and was easy to deploy on our Jenkins nodes.

To get more speed, we are evaluating precompiled headers, incremental linking, and tools like Cotire [2]. CMake makes it very easy to switch tools and play with different compiler and linker settings, encouraging experimentation and moving us toward our goal of reducing as much as possible the time developers have to wait for a build to complete.

**CUSTOMIZATION**

The stock CMake provided almost all the functionality we required, but we still needed to extend it to suit our needs. Fortunately, everything could be written using CMake's powerful scripting language.

**JavaScript build system:** We include many JavaScript source files in our product, so we developed a small build system that handled dependencies, concatenating source files, stripping source files of debug statements, and minimizing the code.

**Resource compiler:** We embed most of our baked-in assets (image files, JavaScript, error page, fonts, etc.) in the executable, which is then signed. CMake handles locating resources using objcopy, ld, or a custom tool to convert them to object files (depending on the platform), generating the C++ code that accesses the symbol table, and creating the static library that's linked to the main program.

**Component system:** We added the ability to specify the paths from which any component can be loaded, supporting binary or source builds, and the ability to add the component as an external library or a subproject with add_subdirectory.
At Netflix, we have a highly automated test infrastructure that starts our application (a custom JavaScript based application framework) and thoroughly tests every component, as well as our layout and rendering engine. We need to make sure that the User Interface has pixel fidelity across many graphics libraries such as OpenGL or DirectFB.

The test framework produces a detailed report of test pass/fails, and a list of image diffs where the current rendering differs from a set of reference images. It also gathers memory usage across the test run and performs six, eight, and 12 hour stress tests.

We wanted to automate the testing runs, consolidate the results on an easy to read web page, and run memory leak detection, so we looked to CTest and CDash, as they offer the functionality we needed.

The first hurdle was that neither CTest nor CDash supported Perforce, the Version Control System we use, so we wrote patches to add support for it and extended CDash to also support P4Web.

An AJAX based log viewer to view and search huge log files without having to download the file to the client machine.

A vector charting package. We measure memory usage for a test run by gathering memory usage every second. In the past, we were creating a PNG file and uploading it as another measurement. Now, we upload a JSON object with the data and use Highcharts to plot it. This allows the user to inspect each data point and zoom in real time.

Image diffs. Differences created by a change in our layout engine can be very difficult to see by the naked eye. We implemented a new ImageDiff viewer that displays the before/after images, letting the user drag a slider to see both. We also use resemble.js [3] to render the differences to a canvas object in bright magenta.

Static analysis. We plan on integrating reports by open source tools like the Clang Static Analyzer or cppcheck.

CONTRIBUTING

We have already contributed back the patches for Perforce support and some general CDash bug fixes, which we expect will make the next CMake and CDash releases. Our CDash custom viewers and static analysis integrations are works in progress, which we fully expect to commit once the code is production quality and has been thoroughly tested.

CMake, CTest, and CDash have proven to be invaluable tools for us to build multiplatform code, track changes, run tests, and improve code quality by performing code coverage and memory leak analysis.

REFERENCES

[1] https://github.com/icecc/icecream

Pedro Navarro is a Senior Software Developer on the Core Technologies team of the Streaming Devices group at Netflix, where he works on the application framework and SDK that drives the Netflix experience for consumer electronics, game consoles, and mobile devices.

UPCOMING CMAKE COURSE REGISTRATION

Kitware is hosting an on-site training course titled “Project Lifecycle Management with the CMake Family of Tools” in Santa Fe. The course will be held March 4, 2014. Through a set of tutorials and exercises, the course will provide people with an in-depth examination of how CMake works, as well as how it can be used to efficiently write scripts for small to larger projects.

To register, visit at http://training.kitware.fr/browse/52
After evolving for several decades, 3D printing has finally reached the consumer market. Today, it is possible to access 3D printers with a wide variety of features and characteristics at prices ranging from $300 to $3,000.

Last month, our first two 3D printers arrived at Kitware. Since then, Kitwareans have been busy exploring the realm of possibilities involving the printers’ interplay with our open source platforms for visualization, image analysis, and computer vision.

**THE PRINTRBOT**

The first printer to arrive was the humble Printrbot, a fully Open Source/Open Hardware printer. The Printrbot comes disassembled in a box. The structural parts are laser-cut wooden plates, which fit together with ingenious connections that give the printer a simultaneously retro and futuristic look.

Following its well-detailed instructions, it took us close to seven hours to assemble the Printrbot. The Printrbot can be run from Windows, Mac, or Linux machines by using Repetier.

Using this printer, we created adapters to optical instruments. For example, we created an adapter to connect the Raspberry Pi Camera to a small low-cost microscope.

The model was created with OpenSCAD, processed with Slic3r to generate G-Code, and printed from Repetier.

This model is now available at [http://www.thingiverse.com/thing:214466](http://www.thingiverse.com/thing:214466) and is distributed under a Creative Commons by Attribution 4.0 license.

**THE MAKERBOT REPLICATOR 2X**

The second printer, the Makerbot Replicator 2X, arrived a week after the Printrbot. It required much less assembly. The Makerbot Replicator 2X is an experimental version of the Replicator model.
Motorcycle model is from a fluid flow simulation that is commonly presented in ParaView. (bottom) Classic teapot model from Utah. More information can be found at http://en.wikipedia.org/wiki/Utah_teapot.

Model of a caffeine molecule. For more information, visit http://www.thingiverse.com/thing:14114.

One common method for printing models is to first generate an STL file. This is something we can do directly with VTK or ParaView. Once generated, the STL file is loaded into the Makerbot application. In the case of the Printrbot, the file is loaded into Repetier, which then uses Slic3r. From the STL model, a full sequence of movements is computed for the printer head extruders to draw the object layer by layer.

Depending on the geometry, some models may need to be printed with supports (see the motorcycle and the caffeine molecule pictures for examples). In certain cases, adding a supporting base, known as a raft, helps to keep the multiple parts of a model together as it is being printed.

Although 3D printing has now reached the consumer market, 3D printers are still more like “tools” than they are “appliances.” That is, there are many parameters that need to be taken into account and that may need to be adjusted to print different models. Examples include the temperature in the extruders, the type of plastic material (ABS/PLA), the temperature of the platform, and the thickness of the layers. For the Makers community, the printer’s flexibility is considered a nice feature, as it leaves those using the 3D printers with a fair amount of control.

Using the Markerbot Replicator 2X and ABS plastic, we printed bone models that were originally segmented with Slic3r. The models were reoriented with Paraview before they were printed separately. The models correspond to sections of the Mandibular bone and the Temporal bone.

WHAT IS NEXT?
As we grow more familiar with the 3D printing process, we are starting to explore synergies with 3D medical imaging, especially with the segmentation of anatomical structures and with the crafting of devices that can be coupled with sensors and small computing platforms such as the Arduinos and the Raspberry Pis.

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.
A LITTLE SOFA HISTORY

SOFA is a software project created eight years ago to tackle a seemingly obvious but unaddressed issue: Many scientists and Ph.D. Students in the mechanical simulation field had to regularly reinvent the wheel. Collision detection, FEM simulation, and contact solver are some of the many common algorithms that have been rewritten several times by different people.

In the French research institute INRIA, three teams working on medical simulation joined their efforts to write a set of reusable algorithms around a common, open framework. As a result, Simulation Open Framework Architecture (SOFA) was born. The main paradigm was to separate the layers of a simulation into different components (collision models, deformation models, force fields, solvers, etc.) so that, when working on a new algorithm for one of the components, you no longer need to care about the implementation of the other components. This saves a substantial amount of time.

From some of the architecture's core components, the codebase grew rapidly as new users became involved in SOFA's development. There was no real project management; the only principle was to share work with others in order to save development time in the future.

SOFA has since become a standard for research teams around the globe. Its modular architecture makes it suitable for many simulation subjects, regardless of whether or not they fall under the domain of the medical field.

NEED FOR A NEW BUILD SYSTEM

As SOFA has been used for a large number of research papers and dissertations, it has had dozens of contributors who have added a comparable number of sub-projects. This includes around 200 library projects that have many external library dependencies and run in various hardware and software environments. As SOFA is used on three different operating systems (not including game consoles), in addition to GPU support (Cuda & OpenCL), various haptic devices, stereoscopic video output, and spatial tracking systems, it can present quite a challenge for the build system!

One year ago, the entire build system was based on qmake. The setup process to build SOFA from source involved copying and patching some text preference files, setting up the paths to the external SDKs, launching the project generation, making the product, and waiting and hoping for everything to be fine in the following two hours. Having everything turn out fine was rarely the case on the first run.

Despite my 12 years of software engineering experience, between the external dependency problems, understanding the setup of configuration files, and the build time, my first SOFA build took me a whole working day. It took even longer for me to begin to understand what was under the hood of SOFA, but that is another story.

Our build system configuration was primarily missing:

1. **Ease of customization:**
   While it is not uncommon in software development to open and modify a text preferences file, it is definitely not the easiest task for users who are not software developers.

2. **Reliability of the external dependencies:**
   No automatic library detection was provided. It was up to the user to setup the library path and to specify what library was present on the system. The user had to hope that the library on the system corresponded to the version used by SOFA.

3. **Early setup problems detection:**
   It was common to issue a « make » command and come back from lunch one hour later to see that the build went wrong because the « x » or « y » library was not present.

CMake looked very promising in addressing these points and, as it turns out, we would not be disappointed.

CMAKE IMPROVEMENTS

Among the different advantages of our new CMake-based build system, I am personally thankful for several features.

First, I am thankful for CMake's shadow builds feature. For a project like SOFA, which has many optional features and possible option combinations, being able to quickly test a
modification on several typical setups on my machine before pushing a significant change to the repository is a very valuable security. This is especially true for a complex project like ours.

Second, in my opinion, the library system is very elegant. Issuing a « find_package(Python 2.7 REQUIRED) » is much more elegant than directly quoting the name of the library file. Moreover, most of the time, you do not have to bother determining the library location on your system. CMake finds it for you. In rare cases when CMake does not know what library you are referencing, you can add your own « findXXXX.cmake » scripts to the build system, which is great!

And last but not least, CMake is substantially stricter than qmake regarding configuration problems. If there is a missing library or an incorrect path, you are immediately told that there is an error during the makefiles generation. Therefore, you are no longer given a link error after two hours of build! As a result, I have reduced my coffee consumption while I wait for the result of the build.

The only lacking feature we would have liked to use was the multiple choice option, as we met several cases where we had options with three or more possibilities. But to be honest, I recently learned this feature has existed since CMake 2.8! As we already require CMake 2.8 or greater for our scripts, this is no longer a problem. This is yet another confirmation we made the right choice, using a tool which is constantly evolving in the right direction.

Anyway, this was a small inconvenience in comparison to all the benefits we have experienced by using CMake over our former build system.

A LONG EXODUS TO THE PROMISED LAND

It took almost six months to handle the compatibility issues we met during our migration from the qmake world to the CMake Promised Land. After three weeks, we had the first running build. Problems started to arise, however, when we began to make some volunteer « guinea pigs » using CMake. You could not imagine all of the strange use cases we met.

Nevertheless, this was not a problem related to CMake. On the contrary, the problem was on our side. SOFA is a very complex piece of software, and the former build system had been developed for years to address all potential issues. This development, however, came at the price of various tricky non-documentated hacks that are known by some experts.

Now, after some changes have been made, it is much easier for someone who does not know the framework to generate a default build of SOFA on (almost) the first try. There is still a long way to go to clean and simplify SOFA, but on the particular point mentioned above: mission complete!

USERS’ FEEDBACK

It was such a big surprise to me that almost no one complained about our move to CMake. I was expecting there to be more yelling and screams of horror. I thought that habits would be much harder to change. Perhaps the lack of resistance can be attributed to the notion that we took the time to adequately develop the CMake scripts before making people switch. We also found that some people were naturally contributing to our CMake effort during the creation process of the CMake scripts, which makes me think people wanted to switch to CMake.

We held a « SOFA Day » in November that had almost 50 attendees. Throughout the day, we were given feedback by SOFA users during roundtables and coffee-break discussions. The feedback regarding the CMake transition was unanimously good. It is always pleasant to hear you made the right choice.

FUTURE ENHANCEMENTS

SOFA has always been distributed in a source code format. We really hope CMake will soon help us provide binary packages for various system environments. While not impossible, the SOFA options combinatory problem and its dependencies make providing binary packages for different system environments a non-trivial task. Moreover, we have a lot of work to complete on refactoring some core mechanisms before we can release a decent stand-alone binary package. After all, SOFA is still, and always will be, under development!

ACKNOWLEDGEMENTS

Engineers who worked on this migration:

Olivier Carre (INRIA Grenoble – Rhône-Alpes)
Bruno Carrez (INRIA Lille – Nord Europe)
Brina Goyette (INRIA Sophia Antipolis – Méditerranée)

SOFA framework: http://www.sofa-framework.org
INRIA: http://www.inria.fr/en

Bruno Carrez has been employed in the video game industry for 12 years. During this time, he has worked on several commercial console games on PS2, Wii, PSP, and X360. As a software engineer, he focused on multi-platform engines (sound, IO, and physics) and project management. He joined the SOFA dev team in INRIA Lille – Nord Europe in 2012 as the new head developer.
In recent months, we have been actively working with patient advocacy groups in the Rare Diseases communities. In particular, we have been getting together with civic hackers to better understand how information systems can be of benefit to the rare diseases communities, and using that knowledge to develop prototypes of software systems that could provide support for patients and their families.

**WHY RARE DISEASES?**
Rare diseases are defined as those that afflict populations of less than 200,000 patients, or about 1 in 1,500 people.

There are, however, about 7,000 rare diseases.

The patients affected by them, and their families, struggle due to lack of information and general knowledge on the nature and treatment of these afflictions.

It takes an average of 7.5 years for a patient to get a correct diagnosis for a rare disease, after having seen an average of eight doctors. By then, these patients have been treated for a variety of incorrect diagnoses and have missed the proper treatment for their case.

Most rare diseases are genetic and, thus, are present throughout a person’s entire life, even if symptoms do not immediately appear. Many rare diseases emerge early in life, and about 30 percent of children with rare diseases will die before reaching their fifth birthday.

Although each one of the rare diseases by itself affects a relatively small number of patients, the combined 7,000 communities result in an impact of about 30 million people in the US and three million in the UK.

Due to the nature of rare diseases, it is difficult for healthcare professionals to become familiar with any of them and, much less, to develop expertise and competence with their treatment. Instead, families become the regular caregivers and, in turn, the qualified experts on the symptoms, manifestations, and day-to-day management of the patients.

Families also become very active in following the activities of the research community on topics that might be relevant to the particular disorders affecting their family members. In certain cases, these communities also drive research activities. Some even fund clinical trials for potential drugs. As a consequence, families in these communities are strong supporters of Open Access policies for publications that result from publicly funded research.

One of the common challenges for families in the rare diseases communities is to gather and consolidate the medical records of their patients, given that they will originate from a diverse set of medical institutions with a variety of medical records systems, as well as a large number of doctors from different specialties. It is also challenging for these families to network with others who might be afflicted by the same or similar disorders.

In this context, the rare diseases communities can greatly benefit from electronic health records (EHR) systems, which are designed to be managed by patient communities through a combination of Personal Health Records (PHR) systems and Health Information Exchanges (HIE). Given that this problem is common to most, if not all, of the rare diseases communities, it is also a natural target for pursuing a common effort that will benefit the aggregate of their 7,000 communities.

The economics of software development dictate that such systems can be developed at lower costs, in shorter time frames, and at higher qualities by taking advantage of Open Source methodologies.

**THE HACKATHONS**
A series of four hackathons have taken place so far. The Hackathon events have been coordinated in collaboration with Ed Fennell, who is driving the Forum on Rare Diseases at the Albany Medical Center. Ed Fennell also recently gave a talk, raising awareness about rare diseases, at TEDxAlbany on November 14th.

During the hackathons, students and faculty from the State University of New York at Albany and the Rensselaer Polytechnic Institute joined forces with civic hackers and...
patient advocates to craft the outline of the type of information system that could track symptoms and phenotypes of any rare disorders.

This leaves us with the impression that, in some cases, the licensing terms are longer than the medical dictionaries themselves.

Following agile methodologies, a first prototype has been crafted, which is hosted in Github at [https://github.com/EmilyAndHaley].

Currently, the development of the prototype has included an exploration of the many medical dictionaries that need to be combined in a typical medical information system, such as UMLS, SNOMED, RxNORM, CPT, and ICD. Each of these dictionaries is licensed under a variety of very specific terms.

The prototype has also led the group of civic hackers to discuss the technology that will be used for storing medical information that is gathered from a variety of sources. The sparse and highly interconnected nature of medical information has led the group to consider the use of Semantic Web technologies such as RDF and its associated serializations and querying mechanisms such as the SPARQL query language.

SYNERGIES

The effort pursued in these hackathons have natural synergies with the work that many other organizations have done in the realms of:

- Personal Health Records
- Health Information Exchanges
- Quantified Self
- Data Sharing
- Education for medical students
- Testing for medical records systems
- Aggregation of medical information
- Interoperability of Electronic Health Records systems
- Open Source Electronic Health Records

The open source nature of the hackathon events makes it possible to take advantage of the network effects resulting from the resonance between the abilities and interests of many of these communities.

The hackathon activities will continue on a regular basis in 2014. If you would like to join and contribute to these efforts, please reach out and let us know.

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.
The VES/Kiwi development team has added a new feature to the kiwi library for use in iOS and Android applications: animated transitions.

Transitions are animations of a change in the scene graph over time. They are useful because they provide users with information about how items in the two scene graphs (the initial and final graphs) are related to each other. Heer and Robertson [1], in “Animated Transitions in Statistical Data Graphics,” offer a thorough discussion of transitions and present a taxonomy of transition types in scientific visualization.

In the kiwi library, transitions are transient in nature; once you construct a transition and queue it for the application to process, you need never inspect it again, as the object representing the transition is reference-counted and disowned by the application once the animation is complete.

Transition objects hold the initial and final state of one or more scene graph node properties (e.g., a color) and compute intermediate values based on the amount of time elapsed since the transition was queued. Any transition can be run in series or in parallel with other transitions. When run in parallel – as long as they do not set the same attribute on the same scene-graph node – the changes will appear simultaneously. When run in series, one transition starts as its predecessor completes, making complex sequences of actions possible.

The new transitions also provide easings. Easings are ways to alter the rate at which a transition occurs relative to the actual elapsed time. They are frequently used in traditional (hand) animation to achieve realistic motion and/or add “character” to motion. Instead of linearly interpolating a property between its initial and final values according to Δt (the time since the transition began), the time is composed with a nonlinear easing function. Many easing functions are illustrated on easings.net and most of them are available in the kiwi library.

What follows is a tutorial on the specific way transitions are implemented in kiwi.
removing an object from the scene graph may keep it from being rendered, but it will still exist and be modified by the transition until completion.

Transitions can be told to queue other transitions as they become active or when they complete using one transition’s alsoStart or followedBy, respectively:

kiwiApp->addTransition(
    vesKiwiCameraTransition::create(camera)
    ->setInitialFocus(vesVector3f(0., 0., 0.))
    ->setFinalFocus(vesVector3f(0., 0., 10.))
    ->setDuration(2.0)
    ->alsoStart(
        vesKiwiTransformTransition::create(transform)
        ->setTranslation(vesVector3f(10., 0., 0.))
        ->setDuration(2.0))
    ->followedBy(
        vesKiwiTransformTransition::create(transform)
        ->setRotation(vesVector3f(1., 0., 0.), 90.0)
        ->setDuration(2.0)));

You can see how indentation can be used to show the transitions that are being referenced, but you must be careful to maintain the code properly because it can also be misleading in a way that makes debugging difficult.

In certain cases, you might want to reference one transition’s properties while setting up another transition. However, creating variables to hold the transition references so that you can refer to property values can be awkward. Thus, the create methods usually take an optional final argument that is a shared pointer reference to the transition. The pointer is set as the instance is created. This example sets the duration of one transition to match another one:

// Ensure transitions started at the same time
// complete at the same time:
vesKiwiCameraTransition::Ptr tx1;
kiwiApp->addTransition(
    vesKiwiCameraTransition::create(camera, tx1)
    ->setInitialFocus(vesVector3f(0., 0., 0.))
    ->setFinalFocus(vesVector3f(0., 0., 10.))
    ->setDuration(2.0)
    ->alsoStart(
        vesKiwiTransformTransition::create(transform)
        ->setTranslation(vesVector3f(10., 0., 0.))
        ->setDuration(tx1->duration())));

INITIAL CONDITIONS
Transitions do their best to obtain initial conditions from the current state of the objects they will be animating at the time they are queued by the application, not at their construction time. The queue time and construction time can be different, specifically when transitions are run in series using the followedBy() method. Most transitions also provide a means for you to manually set the initial state. However, this can lead to discontinuities at the beginning of a transition. Therefore, it is important to be careful.

FINAL CONDITIONS
In some instances, you may not want to spend the time precomputing a transform matrix by hand in order to get a good final state for a transition. In these situations, you can force initial/final state settings during construction:

vesKiwiCameraTransition::Ptr tx1;
kiwiApp->addTransition(
    vesKiwiCameraTransition::create(camera, tx1)
    ->setInitialFocus(vesVector3f(0., 0., 0.))
    ->setFinalFocus(vesVector3f(0., 0., 10.))
    ->setDuration(2.0));

You can see how indentation can be used to show the transitions that are being referenced, but you must be careful to maintain the code properly because it can also be misleading in a way that makes debugging difficult.

In certain cases, you might want to reference one transition’s properties while setting up another transition. However, creating variables to hold the transition references so that you can refer to property values can be awkward. Thus, the create methods usually take an optional final argument that is a shared pointer reference to the transition. The pointer is set as the instance is created. This example sets the duration of one transition to match another one:

// Ensure transitions started at the same time
// complete at the same time:
vesKiwiCameraTransition::Ptr tx1;
kiwiApp->addTransition(
    vesKiwiCameraTransition::create(camera, tx1)
    ->setInitialFocus(vesVector3f(0., 0., 0.))
    ->setFinalFocus(vesVector3f(0., 0., 10.))
    ->setDuration(2.0)
    ->alsoStart(
        vesKiwiTransformTransition::create(transform)
        ->setTranslation(vesVector3f(10., 0., 0.))
        ->setDuration(tx1->duration())));

INITIAL CONDITIONS
Transitions do their best to obtain initial conditions from the current state of the objects they will be animating at the time they are queued by the application, not at their construction time. The queue time and construction time can be different, specifically when transitions are run in series using the followedBy() method. Most transitions also provide a means for you to manually set the initial state. However, this can lead to discontinuities at the beginning of a transition. Therefore, it is important to be careful.

FINAL CONDITIONS
In some instances, you may not want to spend the time precomputing a transform matrix by hand in order to get a good final state for a transition. In these situations, you can force initial/final state settings during construction:

vesKiwiCameraTransition::Ptr tx1;
kiwiApp->addTransition(
    vesKiwiCameraTransition::create(camera, tx1)
    ->setInitialFocus(vesVector3f(0., 0., 0.))
    ->setFinalFocus(vesVector3f(0., 0., 10.))
    ->setDuration(2.0));

// Force the transition to grab initial values now:
// =>takeInitialOrientation();
// It is safe to modify the camera here because the // transition will reset the state at the beginning // of the next render.
camera->setViewPlaneNormal(vesVector3f(1., 0., 0., 0.));
camera->reset();
// Tell the transition to use the camera's current // state as the final endpoint:
// =>takeFinalOrientation();

Methods for forcing a transition to sample the current state of an object to obtain an initial or final value generally have names beginning with take and do not accept arguments.

TRANSITION TIME VERSUS PROGRESS
Most transitions have a clear beginning and ending state and a fixed duration. If this is the case, you can change the rate at which the transition occurs to make it visually smoother (or more overt). The way to do this is with easing functions à la Penner [2] or Sitnik [3]. Kiwi provides several functions, all of which are derived from vesKiwiEasing. Each one takes a progress indicator number between 0 and 1 and returns a modified number between 0 and 1. The modified number is passed to the transition, which uses it as a “fraction completed.”

NON-TRANSITIONS
Sometimes you need a transition for which no completion time is known a priori. One example would be showing FTP download progress when the file size is unavailable; there is no way to know how much of the download is completed, so you cannot estimate where to position an absolute-valued progress indicator. This is not something vesKiwiTransition supports directly yet, but you can write subclasses that provide this kind of behavior. In such cases, your transition should ignore the “fraction completed” passed through the easing and to the transition’s prepareState() method. Instead, prepareState() can query the transition for the current time and its start time, using the difference to define some state. In the future, a method may be available to obtain the time since the last rendering.
**SPECIFIC TRANSITIONS**

**Camera Transition**

Camera transitions allow you to independently change the following:

- The initial and final focal points (where the camera is aimed)
- The initial and final orientations of the camera (the direction of the “lens” from the focal point) as a quaternion
- The focal distance of the camera lens from the focal point
- The parallel scaling factor used when parallel projection is enabled

Camera transitions also allow you to set the entire initial or final “frame” at once by specifying an eye, aim, and up vector.

See src/kiwi/Testing/TestAnimation.cpp for an example of how to use this class.

**Actor Transitions**

There are three classes you can use to transform actors:

- vesActorTranslationTransition provides a way to transition the translation property of a vesActor instance
- vesActorCenterTransition provides a way to transition the center property of a vesActor instance controlling its center of rotation
- vesActorRotationTransition provides a way to transition the rotation property of a vesActor instance

These are all simply template-specializations of the vesKiwiVarTransition class, which uses get and set methods to obtain initial values and prepare intermediate states.

See src/kiwi/Testing/TestGradientBackground.cpp for an example.

**REPRESENTATION COLOR TRANSITIONS**

The vesKiwiPolyDataRepresentation class provides methods to set the color and opacity of VTK polydata. As with actor transitions, we provide typedefs to template specializations of the vesKiwiVarTransition class to control color and opacity:

- vesKiwiPolyDataColorTransition
- vesKiwiPolyDataOpacityTransition

Note that currently, the kiwi library ignores opacity. Therefore, transitioning it is not likely to be useful.

**SCALAR VALUE**

Finally, you can transition any variable that uses a primitive storage type given just its address. The vesKiwiScalarTransition is a template class that takes a pointer to a primitive storage type (e.g., double or int) and transitions the value stored. Note that you must guarantee that the address is valid for the duration of the transition.

See src/kiwi/Testing/TestTransitions.cpp for an example of how to use this class.

**AN EXAMPLE**

Finally, although it is difficult to capture the action of a transition, Figure 1 shows several frames of an animation generated by translating two actors (using a “bounce” easing) as the camera is rotated. The actors are polydata created using ParaView’s “3D Text” source and were placed in the same directory as the other Kiwi sample data.

![Figure 1. Several frames of an animation](image)

```cpp
demoApp::Ptr app; // A subclass of vesKiwiBaseApp
std::string filename1 = this->sourceDirectory() +
  std::string("/Apps/iOS/Kiwi/Kiwi/Data/txt-ves.vtp");
std::string filename2 = this->sourceDirectory() +
  std::string("/Apps/iOS/Kiwi/Kiwi/Data/txt-kiwi.vtp");
vesKiwiPolyDataRepresentation::Ptr rep1 = app->
  loadData(filename1);
vesKiwiPolyDataRepresentation::Ptr rep2 = app->
  loadData(filename2);
app->resetView(false);
app->addTransition(
  vesActorTranslationTransition::create(rep1->actor())
  ->setInitialValue(vesVector3f(-4., 3.0, 0.))
  ->setFinalValue(vesVector3f(0., 0.0, 0.))
  ->setEasing(vesKiwiBounceOutEasing::create())
  ->setDuration(3.00)
  ->alsoStart( vesActorTranslationTransition::create
    rep2->actor())
  ->setInitialValue(vesVector3f(4., -3.0, 0.))
  ->setFinalValue(vesVector3f(0., 0.0, 0.))
  ->setEasing(vesKiwiBounceOutEasing::create())
  ->setDuration(3.00))
  ->alsoStart( vesKiwiCameraTransition::create(app
    ->camera())
  ->setInitialFrame(
    vesVector3f(3.548, 0.5, 20.), // eye pt
    vesVector3f(3.548, 0.5, 0.), // aim pt
    vesVector3f(1., -1., 0.) // up
  )
```

Kitware exhibited recent work in HPC, data analysis and visualization, and modeling & simulation at Supercomputing 2013 (SC13) in Denver, CO. At the conference, Kitware provided interactive demonstrations to showcase state-of-the-art technology for immersive, interactive visualization. Such technology includes immersive VR with ParaView. In collaboration with Intel, Kitware also presented its vision for a parallel universe. The presentation focused on scientific computing at extreme scale.

For SC13’s Exhibitor Forum Series, Utkarsh Ayachit presented “Tools for Data Analysis and Visualization at Exascale.” The presentation highlighted the Visualization Toolkit (VTK) and ParaView, as well as collaborative efforts on the Data Analysis at Extreme (DAX) toolkit, developed by Sandia and Kitware, and the Portable Data-Parallel Visualization and Analysis Library (PISTON), developed by LANL. In addition, Kitware’s David E. DeMarle, along with W. Alan Scott, Li-Ta Lo, and Kenneth Moreland, taught a “Large Scale Visualization with ParaView” tutorial. The tutorial presented the architecture of ParaView, as well as the fundamentals of parallel visualization. It included hands-on lessons and detailed guidance in visualizing the massive simulations that run on today’s supercomputers.

David Thompson is an R&D Engineer in the scientific computing group at Kitware, Inc., focused on interactive, visual tools for modeling, simulation, and analysis. He was a mechanical engineer once.

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.

Casey Goodlett focuses on developing innovative software solutions to research problems in the fields of medical image analysis and computer vision. In particular, Casey has experience in registration of data (images, point clouds, surfaces), optimization methods, software application development, and software project management.

Kitware and the Lung Cancer Alliance (LCA) announced the launch of a new and improved website for the Give a Scan program in December. The program is the first CT donation program that allows people to donate CT scans and metadata directly to an open data archive for lung cancer research. The new website is a portal that provides researchers access to these data for research purposes.

Kevin Kim

Kitware’s Technology Forms Foundation of Give a Scan Website

Kitware’s Technology Forms Foundation of Give a Scan Website

Kitware exhibited recent work in HPC, data analysis and visualization, and modeling & simulation at Supercomputing 2013 (SC13) in Denver, CO. At the conference, Kitware provided interactive demonstrations to showcase state-of-the-art technology for immersive, interactive visualization. Such technology includes immersive VR with ParaView. In collaboration with Intel, Kitware also presented its vision for a parallel universe. The presentation focused on scientific computing at extreme scale.

For SC13's Exhibitor Forum Series, Utkarsh Ayachit presented "Tools for Data Analysis and Visualization at Exascale." The presentation highlighted the Visualization Toolkit (VTK) and ParaView, as well as collaborative efforts on the Data Analysis at Extreme (DAX) toolkit, developed by Sandia and Kitware, and the Portable Data-Parallel Visualization and Analysis Library (PISTON), developed by LANL. In addition, Kitware's David E. DeMarle, along with W. Alan Scott, Li-Ta Lo, and Kenneth Moreland, taught a "Large Scale Visualization with ParaView" tutorial. The tutorial presented the architecture of ParaView, as well as the fundamentals of parallel visualization. It included hands-on lessons and detailed guidance in visualizing the massive simulations that run on today's supercomputers.

David Thompson is an R&D Engineer in the scientific computing group at Kitware, Inc., focused on interactive, visual tools for modeling, simulation, and analysis. He was a mechanical engineer once.

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

Casey Goodlett focuses on developing innovative software solutions to research problems in the fields of medical image analysis and computer vision. In particular, Casey has experience in registration of data (images, point clouds, surfaces), optimization methods, software application development, and software project management.

Kitware and the Lung Cancer Alliance (LCA) announced the launch of a new and improved website for the Give a Scan program in December. The program is the first CT donation program that allows people to donate CT scans and metadata directly to an open data archive for lung cancer research. The new website is a portal that provides researchers access to these data for research purposes.
to patient-provided scans and supplementary information for use in lung cancer investigations, and will accelerate the pace of research. By having access to larger collections of openly available images and clinical histories, researchers can better collaborate on all aspects of lung cancer including precancerous conditions, risk assessment, early detection and diagnosis, and treatment. Both patients and researchers can contribute to Give a Scan’s database.

The launch of Give a Scan’s new website is supported by LCA, Siemens, and Kitware. The site is powered by Midas, Kitware’s open-source data management system.

**ANDINET ENQUOBAHRIE TALKS ABOUT OPEN SOURCE SOFTWARE AT IDEAS SYMPOSIUM**

Andinet Enquobahrie, Kitware’s Assistant Director of Medical Computing, presented a talk on the role of open source-software in surgical simulation at the Innovation, Design, and Emerging Alliances in Surgery (IDEAS): Virtual Surgery Symposium. The symposium took place at the Beth Israel Medical Center in Boston, MA, this past November. The goals of the symposium were to provide a comprehensive update regarding state-of-the-art technology in virtual surgery and to bring together surgeons, engineers, and educators to collaboratively discuss challenges and opportunities in the field.

Andinet presented during the session titled Disruptive Technologies in Virtual Surgery. During his presentation, he provided an overview of open-source tools and detailed current virtual surgery projects at Kitware. He also discussed how technological advancements in medical imaging, image registration algorithms, visualization technologies, and haptics systems are helping to increase the adoption of virtual surgery systems.

**CEO WILL SCHROEDER GIVES TALK AT NEUROSURGICAL SIMULATION SYMPOSIUM**

Will Schroeder, Andinet Enquobahrie, and Ricardo Ortiz attended the 2nd Annual Neurosurgery Simulation Symposium in early November. The symposium took place at The Mount Sinai Medical Center in New York, NY, and was hosted by the center’s Department of Neurosurgery. The symposium’s attendees included those from fields such as mechanical engineering, applied mathematics, brain surgery, and computer sciences.

At the symposium, Will presented on “Open Source Approaches to Simulation.” The presentation detailed an introduction to Kitware’s software processes and explained the significance of open science in the creation of simulation technologies. Other presentations included “Simulation in Aviation” by Christopher Broom, “Simulation and Imaging in Medicine” by Suvaranu De, and “Business Models for Surgical Rehearsal” by Joshua B. Bederson.

**LUIS IBÁÑEZ RECEIVES AWARDS AT VISTA EXPO & SYMPOSIUM**

Luis Ibáñez, a Technical Leader at Kitware, received two xJourney VistA Webinar Awards during the recent 2013 VISTA Expo & Symposium in Seattle, WA. These awards, which recognized his webinar presented earlier in the year entitled “GT.M interface to NodeJS,” were in the Record Hours Watched and the Best Webinar - Community Choice categories. The Record Hours Watched award is granted to the webinar that is watched the most times out of those submitted to the symposium, while the Best Webinar - Community Choice award is based on a vote among members of the VistA community.

In the webinar, Luis describes work by Rob Tweed and David Wicksell to integrate Node.js with GT.M and, thus, provide a server-side Javascript interface into the M database and language. This is important given that there are hundreds of healthcare applications representing millions of lines of code written in M, yet there are only a small (and decreasing) number of developers proficient in M. By providing a more modern Node.js interface into M, the community can attract some of the hundreds of thousands of Javascript developers into this area. The new interface is being used in Database and Web Development classes at the State University of New York in Albany to attract the next generation of developers into healthcare application development.
ITK CELEBRATES ITS 14TH BIRTHDAY
The Insight Segmentation and Registration Toolkit, known as ITK, celebrated its 14th birthday in November with a code base. The dedicated ITK development team, working both locally in the Kitware Clifton Park offices and web conferencing in from around the world, submitted 34 patches over the course of the day, and shared stories of the early days of the ITK project.

According to Kitware’s Luis Ibáñez and Matt McCormick, the future of ITK includes enhancing support for ARM architectures and advancing infrastructure in order to keep pace with upstream third party libraries. The members of Kitware's team also project that, in the future, ITK will not only develop thread pools that will improve the application of many-core architectures, but the software will also offer reliable interpreted language wrapping for all platforms, as well as methods of recognizing contributions from the community and upgraded documentation in design, architecture, and algorithms.

WINTER PROMOTIONS AT KITWARE
Dr. Andinet Enquobahrie has been promoted to Assistant Director of Medical Computing at Kitware. Dr. Enquobahrie joined Kitware in 2005. Prior to his promotion, he was a Technical Leader with a focus on image-guided intervention and surgical simulations projects. In his new role, Dr. Enquobahrie will continue to lead projects in image-guided therapy, while also providing cross-team coordination for hiring, staffing, mentoring, and project management activities in Medical Computing.

NEW EMPLOYEES
Cory Quammen
Cory joined the Kitware team at the Carrboro, NC, office as an R&D Engineer. He earned his B.A. in Computer Science from Gustavus Adolphus College, where he graduated summa cum laude. He later received an M.S. in Computer Science from the University of North Carolina at Chapel Hill. Prior to joining Kitware, Cory was a research assistant at UNC in the Department of Computer Science.

Sandy McKenzie
Sandy joined the Kitware team in Clifton Park, NY, as a Communications Specialist. She received her bachelor’s degree in communication and media studies with a minor in sociology from Fordham University. Sandy also completed a master’s degree in public relations from the S.I. Newhouse School of Public Communications at Syracuse University.

EMPLOYMENT OPPORTUNITIES
Kitware is seeking talented, motivated, and creative individuals to fill open positions. As one of the fastest growing software companies in the country, we have an immediate need for software developers and researchers, especially those with experience in scientific and medical computing.

Interested applicants are encouraged to visit our employment site at jobs.kitware.com and submit a resume and cover letter through our online portal.

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.

Contributors: Lisa Avila, Bruno Carrez, Aashish Chaudhary, Casey Goodlett, Luis Ibáñez, Pedro Navarro, and David Thompson.

Graphic Design: Steve Jordan
Editor: Sandy McKenzie

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

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