TABLE OF CONTENTS

Editor’s Note........................................................................................................... 1
Recent Releases ........................................................................................................ 1
Eye-Dome Lighting: A Non-Photorealistic Shading Technique ................................ 3
Virtually Everywhere ............................................................................................... 5
A Tour of VTK’s Pointer Classes .............................................................................. 8
Hosting Binary Files on MIDAS to Reduce Git Repository Size .............................. 9
Kitware News ............................................................................................................ 11

EDITOR’S NOTE

The Kitware Source contains articles related to the development of Kitware projects in addition to a myriad of software updates, news and other content relevant to the open source community. In this issue, Alejandro Ribes and Christian Boucheny discuss the use of Eye-Dome lighting in ParaView and how it can improve depth perception in the visualization of 3D datasets. Luis Ibanez explores the use of virtual environments in different applications and the opportunities they can provide. Marcus Hanwell gives an overview of the pointer classes in the Visualization Toolkit (VTK) and the need each satisfies in VTK. Zach Mullen talks about MIDAS’s use of Git and explains how to utilize macros to manage the synchronization of data between the MIDAS server and source repositories. In Kitware News, Will Schroeder reflects, in honor of Kitware’s 13th anniversary, on the factors that have enabled Kitware’s growth over the years.

The Source is part of a suite of products and services offered to assist developers in getting the most out of our 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 our website at www.kitware.com.

RECENT RELEASES

ITK MODULARIZATION IN ITK 4.0

One of the major undertakings of the upcoming ITK 4.0 release is the modularization of ITK. Modularization is the process by which the many classes of ITK will be grouped into smaller and cohesive components. We will refer to those components as modules. This grouping will enable users to select a subset of those components to be used for supporting the development of their application.

The rationale for modularizing the toolkit is the following:

- Growth management
- Raising the bar of software quality
- Removing outdated pieces of software
- Facilitating the use of add-ons to ITK

The need for managing the growth of the toolkit is clearly illustrated in the figure below. Active development of the toolkit was funded by the National Library of Medicine from 1999 to 2005. After that period, only maintenance of the software was funded. Despite the fact that after 2005 there was no direct funding for development, the toolkit continued to grow in an almost linear way. This fact in itself demonstrates that the toolkit was adopted and cared for by a large community of contributors. We anticipate that the toolkit will continue to grow at a similar rate, as new algorithms appear in the literature and are ported to ITK. We must therefore prepare a process of organized growth ensuring the manageability of the software despite its size.

Modularization will also facilitate the evaluation of software quality control metrics at a finer granularity, such as code coverage, valgrind errors, doxygen documentation warnings and coding style. This finer granularity of reporting enables developers to work on problems more effectively, since the
effort can be focused in a given module, and the outcomes of the effort is more clearly verified without being confused with the state of the many other files in the system.

The process of modularizing the toolkit is similar to the process of moving from one home to another. It gives us an opportunity to find any forgotten objects and to reconsider whether we want to keep them or not. We have identified a number of classes to be deprecated, and have also remade the entire CMake infrastructure of the toolkit, now taking advantage of the most recent functionalities of CMake 2.8.3.

Finally, we will configure the modularization in such a way that institutions can offer additional ITK modules to third party users, essentially creating an open ITK market for add-ons. The ITK ecosystem will then expand to include specialized modules that may be of interest to a subset of the community and that traditionally may not have been considered general enough to be included in the toolkit itself.

One of the key aspects of the modularization is the formalization of dependencies across modules. The figure below illustrates these dependencies by taking advantage of the Information visualization functionalities of VTK.

Application developers will be able to prepackage the subset of ITK that they actually use, and thus reduce the burden of maintenance of their complete software environment.

We look forward to the feedback of the ITK community regarding their use cases for modularization and suggestions for making the toolkit a more useful and maintainable resource for the decade to come. Please see http://www.itk.org/Wiki/ITK_Release_4/Modularization

CMAKE 2.8.5
The CMaKe version 2.8.5 release is scheduled for April 2011. The bug fix roadmap for version 2.8.5 can be found at http://public.kitware.com/Bug/roadmap_page.php. The change log for bugs that have already been fixed is located at http://public.kitware.com/Bug/changelog_page.php.

Please try the release candidates for CMake 2.8.5 to build your projects as they become available in April. Let us know on the mailing list if you run into anything unexpected.

MIDAS
The team working on MIDAS has just released updates for the MIDAS Server and MIDAS Desktop. The MIDAS Server 2.8.0 release includes improved server-side processing via the BatchMake plugin, enhanced API to support the MIDAS desktop and an updated shopping cart for data download and aggregation. Additionally, we have enhanced its stability on Windows servers, and added support for files greater that 4 GB. Also of note is the addition of hash-addressed file downloads via the web API and user agreements to afford community administrators greater flexibility in data dissemination. Lastly, MIDAS has moved to Git for version control.

MIDAScpp 1.6.0 has added several new features. The MIDASDesktop GUI layout has been redesigned for usability, progress reporting during upload and download has been improved and a one-step upload command to MIDAScli has been added. There are now options in the Preferences menu to copy or move the entire resource tree to a new location and the client tree is now updated dynamically as data is pulled. Many aspects of MIDASDesktop that are slow have been multithreaded and MIDASDesktop now actively monitors the filesystem so that changes to files under local database control will be recognized automatically by the application. Metadata fields have been added, including total size of communities, collections, and items and can be updated with rich text editors.

Additionally, with this new release, users can:

- Import existing data into their database without having to pull it from the server.
- Cancel long running network-based operations in MIDASDesktop, such as pulls and refreshes.
- Create a new empty local database at any time via MIDASDesktop.
- Pull data from the server by dragging and dropping resources between trees.
- Authenticate using username and password instead of with a web API key.

IGSTK
The Image-Guided Surgery Toolkit team released IGSTK 4.4 in February. This minor release has several new features, including support for Ascension’s 3D Guidance trackers (medSAFE, driveBAY and trakSTAR); PET image readers, spatial object and representation classes; and PET/CT fused image and electromagnetic tracker-guided needle biopsy application examples. The build instructions and new release download can be found on the IGSTK public wiki.
EYE-DOME LIGHTING: A NON-PHOTOREALISTIC SHADING TECHNIQUE

Eye-Dome Lighting (EDL) is a non-photorealistic, image-based shading technique designed to improve depth perception in scientific visualization images. It relies on efficient post-processing passes implemented on the GPU with GLSL shaders in order to achieve interactive rendering. Solely projected depth information is required to compute the shading function, which is then applied to the colored scene image. EDL can, therefore, be applied to any kind of data regardless of their geometrical features (isosurfaces, streamlines, point sprites, etc.), except to those requiring transparency-based rendering.

In this article, we first briefly describe EDL and then give some details about how it has been integrated into ParaView. EDL was developed by Christian Boucheny during his Ph.D [1]. Its original aim was to improve depth perception in the visualization of large 3D datasets representing complex industrial facilities or equipments for Electricité de France (EDF). Indeed, EDF is a major European electrical company where engineers visualize, on a daily basis, complex data such as 3D scans of power plants or results from multi-physics numerical simulations.

WHAT IS EYE-DOME LIGHTING?

Shading occupies a special place among the visual mechanisms serving to perceive complex scenes. Global illumination models, including a physically inspired ambient occlusion term, are often used to emphasize the relief of surfaces and disambiguate spatial relationships. However, applying such models remains costly, as it often requires heavy pre-computations, and is thus not suited for an exploratory process in scientific visualization. On the other hand, image-based techniques, such as edge enhancement or halos based on depth differences, provide useful cues for the comprehension of complex scenes. Subtle spatial relationships that might not be visible with realistic illumination models can be strengthened with these non-photorealistic techniques.

The non-photorealistic shading technique we present here, EDL, relies on the following key ideas.

Image-based lighting: Our method is inspired by ambient occlusion or skydome lighting techniques, with the addition of viewpoint dependency. Contrary to the standard application of these techniques, in our approach the computations are performed in image coordinate space, using only the depth buffer information, like in Crytek Screen-Space Ambient Occlusion (SSAO) [2]. These techniques do not require representation in object coordinate space, and thus there is no need for knowledge of the geometry of the visualized data or for any preprocessing step.

Locality: The shading of a given pixel should rely predominantly on its close neighborhood in image space, as the effects of long range interactions will not likely be initially detected by the viewers.

Interactivity: Our primary concern is to avoid costly operations that would slow interactive exploration and thus limit the comprehension of the data. Due to the evolution of graphics hardware, a limited set of operations performed on fragments appears to be the most efficient approach.

Figure 1 presents a diagram of the architecture that integrates EDL. The algorithm requires a projected color image of the 3D scene and its corresponding depth buffer. The depth buffer is the input of the EDL algorithm, which generates a shadow image to be combined with the color rendering of the scene (e.g., by multiplying each pixel's RGB components by its EDL-shading value).

Figure 1. Depiction of the rendering architecture integrating Eye Dome Lighting. A 3D scene (left colored surface) is first projected using an off-screen OpenGL rendering. The resulting color and depth images are stored in two 2D textures. The depth image is then used to calculate the shading by applying the EDL algorithm. The result is used to modulate the color image that is finally drawn on the screen (e.g., by multiplying each pixel's RGB components by its EDL-shading value).

The basic principle of the EDL algorithm is to consider a half-sphere (the dome) centered at each pixel p. This dome is bounded by “a horizontal plane,” which is perpendicular to the observer direction at point p. The shading is a function of the amount of this dome visible at p, or conversely, it is inversely determined by the amount of this dome hidden by the neighbors of p (those being taken on a ring around p in image space). In other words, a neighbor pixel will reduce the lighting at p if its depth is lower (i.e. closer to the viewer) than the one of p. This procedure defines a shading amount that depends solely on the depth values of the close neighbors. To achieve a better shading that takes into account farther neighbor pixels, a multi-scale approach is implemented, with the same shading function being applied at lower resolutions (typically half and quarter image size). Those shaded images are then filtered in order to limit aliasing induced by lower resolution, using a cross bilateral filter [3] (Gaussian blur modulated by depth differences), and then merged together with the full resolution shaded image. This approach is graphically represented in Figure 2.

Figure 2. The shading function is computed at full resolution and a lower resolution. The final rendering (right) is a weighted sum of the two shading images, with a cross bilateral filter (Gaussian blur modulated by depth differences) applied to the lower resolution map to prevent aliasing and achieve a “halo” effect.

Figure 2. Depiction of the rendering architecture integrating Eye Dome Lighting. A 3D scene (left colored surface) is first projected using an off-screen OpenGL rendering. The resulting color and depth images are stored in two 2D textures. The depth image is then used to calculate the shading by applying the EDL algorithm. The result is used to modulate the color image that is finally drawn on the screen (e.g., by multiplying each pixel's RGB components by its EDL-shading value).
where the Imaging Pass is inserted does not currently allow added by Kitware for EDF R&D. However, the position framework for post-processing image passes was recently in the correct position in the visualization pipeline. This SetImageProcessingPass that will insert our algorithm is a ParaView view. The view contains a new method

The plugin itself consists of a vtkPVRenderView, which is a ParaView view. The view contains a new method SetImageProcessingPass that will insert our algorithm in the correct position in the visualization pipeline. This framework for post-processing image passes was recently added by Kitware for EDF R&D. However, the position where the Imaging Pass is inserted does not currently allow transparency to be applied properly. This is due to the more complex pipeline design for transparency rendering, which relies on depth peeling, and further development in VTK should be done if this is needed.

A method SetUseDepthBuffer has been added to vtkPVSyncronizedRendered to switch on the use of depth buffers. In fact, EDL uses a depth buffer, but most algorithms do not. Always switching on the use of the buffer could lead to a slowing down of the system when the depth buffer is not needed. To avoid this problem, SetUseDepthBuffer is provided. The user is then responsible for activating the depth buffer if his/her algorithm requires it. Its default value is off.

The use of the depth buffer by EDL was a main challenge for the integration of EDL in ParaView. Indeed, the plugin should work in standalone, client-server and parallel-server mode. Moreover, tiled displays are also taken into account. Some developments were performed to allow parallel compositing of the depth-buffer using IceT (the parallel library used in ParaView for parallel compositing operations). Exposing this functionality from IceT to the render passes was the main object of a collaboration between EDF R&D and Kitware for the proper implementation of EDL.

### SHADING ALGORITHMS

The vtkEDLShading is based in another class called vtkDepthImageProcessingPass. This class contains some general methods that are not specific to the EDL algorithm and can be used to implement other algorithms. For instance, we implemented an image-based ambient occlusion shading algorithm (based on Crytek SSAO) and a ParaView view based on it (not currently included in the plugin). Thus, any user could derive a class from vtkDepthImageProcessingPass to implement this kind of algorithm.

### ACKNOWLEDGEMENTS

The EDL algorithm is the result of joint work by Electricité de France, CNRS, Collège de France and Université J. Fourier as part of the Ph.D. thesis of Christian BOUCHENY.

### REFERENCES


Leonardo Ribes is currently working in scientific visualization at EDF R&D. He has experience at the University of Oxford, U.K; the French Atomic Energy Commission, Orsay, France; and the National Yang-Ming University, Taipei, Taiwan. He received his Ph.D. from the Ecole Nationale Supérieure des Télécommunications, Paris, France.

Christian Boucheny is a research engineer at EDF R&D, France, specialized in scientific visualization and virtual reality for maintenance and training. He received his Ph.D. in 2009 from the University of Grenoble, where he worked on perceptual issues related the visualization of large datasets.

---

**Figure 2.** An image produced on EDF physical simulation data rendered with point sprites illustrates the effect of EDL (left) compared to basic Phong shading (right).
WHAT IS A COMPUTER?
There was a time when it was easy to answer this question, a time when a computer was mostly a physical, or hardware, device (see Figure 1). It used to be that on top of that hardware device, a thin layer of logic was used to control the tasks performed by the physical layer.

![Figure 1. The ENIAC, which became operational in 1946, is considered to be the first general-purpose electronic computer. It was a digital computer that was capable of being programmed to address a range of computing issues (http://en.wikipedia.org/wiki/ENIAC).](image)

There have been significant changes since those early days. Modern computers are an amalgam of physical hardware complemented with layer upon layer of abstraction logic. Users of modern computers interact with the higher abstraction layers and no longer get exposed to the details of the lower layers. This multi-layered organization has made it possible to intercept the middle layers and fully disconnect the operating system from the actual physical device in which it is running. As a consequence, what we used to call “the computer” is now disembodied from the physical layer and can therefore be moved across different receptacles where it can be incarnated.

This technology, in general called virtualization, has recently transitioned from advanced and exclusive to mainstream. We have therefore started taking advantage of it, mostly in transitioned from advanced and exclusive to mainstream.

The beauty of virtualization is that the virtual computer is literally a stream of bytes, and therefore can be stored, copied, modified and redistributed just as any other digital good. Granted, it tends to be quite a large file, but so are movies in digital formats.

The main scenarios in which we have recently been using virtualization are:

- Teaching
- Debugging
- Providing reference systems

Here we elaborate on our experience using virtualization in some of these scenarios.

DEBUGGING
Despite being aware of virtualization for some time, the event that sparked our attention came as a secondary effect of teaching our class, “Open Source Software Practices,” at Rensselaer Polytechnic Institute. In order to expose students to the inner working practices of an open source community, they are required to work in a joint class software project and then they work in small groups projects. In a recent class, for the joint project we chose to work with Sahana [1], a piece of humanitarian free and open source software (HFOSS)[2], designed to coordinate the delivery of humanitarian relief to disaster zones.

One of the easiest ways to get introduced to the system was for the students to sign up with the Sahana bug tracking database and select easy bugs that could be tackled in a couple of days. The Sahana development team has done an excellent job of preparing an easy entry path for new contributors. One of the most remarkable items in that reception area was the presence of a virtual appliance with a fully configured build of Sahana, along with a minimal mockup database intended for testing purposes [3]. Students were able to download the virtual appliance (for VirtualBox), boot it in their own laptops, and be working on fixing a bug in a matter of minutes. This was an eye-opening experience.

We know quite well, from our maintenance experience with our open source toolkits, that bringing new developers into an open source project is not a trivial feat. The fact that Sahana succeeded so well in delivering a “portable” platform (in the form of these virtual machines) that new developers can take and use instantly without installing additional software, dealing with version incompatibilities, or having to get familiar with the installation details of a database and an associated web server such as permissions and policies, make this approach a clear winner.

One of the very appealing properties of this method is that the new developers do not need to compromise the configuration of their current computers just to work on a particular bug of a given project. We have seen on many occasions that the effort to replicate the conditions in which a bug happens may require the installation of specific versions of libraries or software tools and their configuration in a particular fashion. Over time, the computers of developers who must do this on a continuous basis end up having an overwhelming mixture of installed libraries, which can easily generate conflicts and obstruct maintenance. With a virtual machine, on the other hand, the process is perfectly clean. The machine is downloaded and booted; the bug is explored and fixed; a patch is submitted; and the virtual machine is shut down and discarded. The developer then returns to a clean computer, with no secondary traces or inconvenient remnants from his recent bug fixing excursion.

TEACHING
When teaching a course to a group of, let’s say, 30 people, it is highly desirable to ensure that all of them have the software correctly installed, a similar directory structure, and
access to the necessary source code and eventually any other binary tools that may be needed for the course. For example, a typical ITK course will require you to have the source code of ITK installed, the proper version of CMake, a well-configured compiler and a set of data files suitable to be used as input for hands-on exercises. This has to be done despite the fact that attendees will use their personal laptops for the course, and therefore will have a large variety of hardware platforms, operating systems and development environments installed on them.

In this context, virtualization offers an interesting alternative. Should a virtualization application be installed in the course attendee’s computers, it becomes possible to give to each one of them a virtual appliance that has been carefully crafted to contain all the software tools needed for the course. Such an appliance can be delivered to attendees in the form of a USB memory stick or a traditional CD.

The Microscopy Tutorial at MICCAI

At the MICCAI 2010 conference, we delivered a tutorial on “Microscopy Image Analysis.” As usual, following a pragmatic approach to training, we wanted to incorporate hands-on exercises in this tutorial, but we were challenged by the need to install a full application (V3D developed by Hanchuan Peng’s team at the HHMI - Janelia Campus) along with a full build of ITK and the set of input data required to run exercises.

The computers used for the tutorial, however, were the laptops that attendees brought as their personal machines to the conference. This was a double challenge. First, a wide variety of different machines was used (Macs, Linux and Windows), and second, the configurations of these machines was vastly different. They had different versions of operating systems and different types and versions of build systems. Virtual machines were therefore a natural choice to isolate that heterogeneity from the uniform platform that we needed to use for delivering a common experience to the course attendees.

The preparation for the course used two independent stages. The first step was installing the virtualization software (in this case VirtualBox, from Oracle). The second step was installing the image of the actual virtual machine (also known as a “Virtual Appliance”). This preparation of the virtual machine certainly requires considerable time and attention, but it has the advantage that its outcome becomes reusable and redistributable.

The slides used for this tutorial [4] and the virtual machine [5] are available on the MIDAS database.

In this particular case, the file containing the virtual machine is about 2 GB in size, but there are better ways to compact a virtual appliance than what we used in this particular case. This virtual appliance can be run both in a VirtualBox application and in a VMWare server.

VirtualBox runs as a standard application on top of the operating system of the host machine. The VirtualBox application can launch the image of a preconfigured virtual machine in which a guest operating system has been installed.

Our use of virtual appliances in the MICCAI Microscopy tutorial was so rewarding that we will be using them again for the following upcoming tutorials:

- CVPR 2011: Video Bridge between ITK and OpenCV
- MICCAI 2011: Microscopy Image Analysis
- MICCAI 2011: Simple ITK : Image analysis for human beings
- MICCAI 2011: ITKv4 : The next generation

INSIGHT JOURNAL

The Insight Journal is the vehicle for members of the ITK community to share their own developed classes with others. One of the most unique characteristics of the Insight Journal is that it is the only Journal that verifies the reproducibility of the work that is described in a paper. In order to do this, authors must include with their papers the full set of source code, data and parameters that enable others to replicate the content of the paper. The system run by the Insight Journal takes this source code, compiles it, runs it, and finally compares the generated data with reference data provided by the authors. Given that the Journal receives submissions from any registered user, and that registration requirements are minimal, the implementation of the Journal translates into: “Here is an online system in which we are going to take source code submitted by anyone on the Internet, and we will compile it and run it.” This is something that is not necessarily the most prudent thing to do.

In order to restrict the risk of damage, whether by malicious or defective code, a Xen virtualization platform was put in place. In this platform, a Linux virtual machine in which commonly used libraries and software tools have been preinstalled is instantiated from scratch in order to test each individual paper submission. In this way, every paper is tested in a uniform environment that is in pristine condition. Should anything go wrong with the build or the execution process, the damage is contained because the instantiation of the virtual machine is discarded after the paper verification terminates.
The virtualization environment also enables us to create a safe “walled garden” in which the code is tested with limited access to risked services such as networking. The image of the virtual machine is updated regularly to include recently released versions of ITK, VTK and CMake, among other tools.

Given that in some cases, users may want to further customize the configuration in which the paper source code is evaluated, we are considering the option in which authors can create the submitted paper by taking a publicly available copy of a pre-configured virtual appliance, then proceed to customize it by installing additional software, including the one that implements the content of their paper, and finally pack the resulting new virtual appliance and submit it as a paper to the Insight Journal.

This of course will have the overhead of transmitting and storing very large files, but it also provides a whole new horizon of possibilities when it comes to the richness of content that can be made part of a technical or scientific publication. It would be one step closer to achieving an ideal environment for verification of reproducibility in computational sciences.

THE CLOUD
Cloud environments are yet another implementation of virtualization technologies. In this case, the cloud provides three main components:

- A repository of virtual machine images.
- A computation service in which users can request hardware on a pay-per-use basis.
- A storage service in which users can deposit data and pay per data size and transfer.

These platforms enable us to provide preinstalled virtual machine images with a configured and built version of ITK, that users can instantiate for their own testing. Prices of running machines in the cloud are in the order of $1 per hour, and users only pay for the time between the instantiation of the machine and when it is shutdown.

Users in the cloud can also take pre-existing virtual machine images, modify them (for example to install in them additional software), and then put these new images back into the cloud repository for others to use. A permission system makes it possible to make some of these images fully public, or to restrict access to a limited set of users.

Cloud computing is a virtualization paradigm in which a collection of computing resources are made available to pay-per-use customers in the form of virtual computers. The cloud service provider (for example Amazon EC2, Rackspace or Microsoft Azure) actually owns a large collection of hardware equipment distributed in different geographical locations. Those hardware devices have been configured to be able to run virtual computers at the request of customers. A virtual computer is equivalent to a one-to-one copy of the byte data existing in the hard drive of any modern desktop or laptop. Such a copy includes not only the software applications that users interact with, but also the operation system layers that normally interact with real hardware. In the context of cloud computing, those virtual machines are essentially run as emulated computers in an environment where the emulation presents a minimum overhead.

Customers of the service can choose among many pre-configured virtual computers (also known as “images”), and can choose to instantiate them on hardware platforms of different capacity (memory, number of processors, disk space), also known as “instances”. User can also select to instantiate as many of these virtual devices as they need, release them or reduce them according to their usage needs, and all along, only pay for the resources that they are using. Software infrastructure is available for performing this scaling automatically, according to the load that an application may be experiencing.

Open-source software platforms are a natural fit for cloud computing environments because open-source software is not crippled by licensing limitations, and therefore can be copied, instantiated, modified and redistributed without any legal concerns.

By storing scientific data in cloud storage services, it becomes available directly to cloud computing devices without further transfer of data. Modern cloud storage providers offer multiple options for uploading large amounts of data, ranging from a high-speed multi-channel upload network to mail-shipped high-capacity storage media (such as multi-terabyte hard drives), that is still the most cost-effective way of transferring large amounts of data. Once customers have uploaded data to the cloud, they can make it available to the virtual machines they instantiate to process it.

We currently host an “image” in the Amazon EC2 elastic computing service. You could instantiate that image and have a functional computer in which an ITK source tree and it corresponding binary build are already available.

CONCLUSION
The luxury of being able to configure, pack and ship around the digital version of a fully configured computer gives us plenty of opportunities to address in a more effective manner the challenges of large scale software development and the joys of building communities around them.

REFERENCES

RESOURCES
If you are interested in trying some of the available pre-configured images, the following resources are helpful:

VirtualBox
The community provides the following two resources.
http://virtualboxes.org/images/
http://virtualboximages.com/

VMWare

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.
One way in which VTK’s API differs from that of some other libraries is that all vtkObjectBase-derived classes must be allocated from the heap. This means that we tend to deal with raw pointers a lot more often than those developing code using other libraries and frameworks, such as the standard template library, Boost and Qt. This has led to the addition of several templated classes to make pointer management, along with allocation and deallocation, easier.

### Automatic Allocation

The newest addition to the set of pointer classes is vtkNew, which is designed to allocate and hold a VTK object. Due to the nature of VTK constructors (no arguments), it is a very simple class; on construction it allocates a new VTK object of the type specified, and on destruction it will call the Delete() method of the class.

```c++
vtkNew<vtkPoints> points;
points->SetDataTypeToDouble();
```

This class effectively maintains ownership of the object, provides a very compact way of allocating VTK objects, and assures that they will be deleted when the pointer goes out of scope in much the same way as stack allocated objects.

The class is new in VTK 5.8.0. To pass the raw pointer to other classes it is necessary to use the GetPointer() method, as mentioned previously, each of the pointer classes can be assigned to the vtkWeakPointer does not increase the reference count of the instance. When the instance being pointed to is destroyed, the vtkWeakPointer being held on to gets set to null, avoiding issues with dangling pointers. It is especially useful in cases where you need to hold on to a pointer for an instance, but there is no need to keep it around should it be deleted. This often makes it far easier to avoid any reference loops too.

```c++
vtkTable *table = vtkTable::New();
vtkWeakPointer<vtkTable> weakTable = table;
```

The weakTable will remain valid until Delete() is called, so in the following code weakTable will never evaluate to true.

```c++
if (weakTable)
{
  // We'll never get here, as table was deleted.
  vtkIdType num = weakTable->GetNumberOfColumns();
}
```

This makes it very easy to check for null before doing anything with the pointer. It is again useful for member variables of classes, in the case where there is no need to maintain ownership of the instance being pointed to.

### Use of Pointer Classes

As mentioned previously, each of the pointer classes can be used for member variables in classes. They have different attributes, and so the intended API around the data being used should be considered.

- **vtkNew**: Strong ownership, where the class instantiates an object which it owns for the lifetime of the class. The member variable cannot be changed in place (much like stack variables).

- **vtkSmartPointer**: Ownership of the instance is maintained. The class may or may not instantiate the initial object and the instance being pointed to can be changed.

- **vtkWeakPointer**: Weak ownership, where the class does not instantiate an object. If the object instance is deleted, then the weak pointer is null, which avoids dangling pointers without ownership.

A simplified class definition might look like,

```c++
#include “vtkSmartPointer.h”
#include “vtkNew.h”
#include “vtkObject.h”
```
#include "vtkSmartPointer.h"
#include "vtkObjectFactory.h"
#include "vtkExample.h"

class vtkTable;
class vtkExample {
  static vtkExample * New();
  void setInputTable(vtkTable *table);
  vtkTable * getInputTable();
  void setColorTable(vtkTable *table);
  vtkTable * getColorTable();
  vtkTable * getTableCache();
protected:
  vtkExample();
  ~vtkExample();
  vtkNew<vtkTable> TableCache;
  vtkSmartPointer<vtkTable> InputTable;
  vtkWeakPointer<vtkTable> ColorTable;
};

The corresponding implementation file would contain,

#include "vtkExample.h"
#include "vtkObjectFactory.h"
#include "vtkTable.h"
 vtkStandardNewMacro(vtkExample)
vtkExample::vtkExample()
{
  this->InputTable =
  vtkSmartPointer<vtkTable>::New();
}

vtkExample::~vtkExample() {}
void vtkExample::setInputTable(vtkTable *table)
{
  this->InputTable = table;
}

vtkTable * vtkExample::getInputTable()
{
  return this->InputTable.GetPointer();
}

void vtkExample::setColorTable(vtkTable *table)
{
  this->ColorTable = table;
}

vtkTable * vtkExample::getColorTable()
{
  return this->ColorTable.GetPointer();
}

vtkTable * vtkExample::getTableCache()
{
  return this->TableCache.GetPointer();
}

Note that there was no need to call Delete() on any of the member variables. The vtkNew class allocates the vtkTable on construction, and that instance cannot be replaced over the lifetime of the class instance. When the class is destructed, the vtkNew object calls delete on the table. Next, the vtkSmartPointer allocates an initial instance of vtkTable, which can later be replaced using setInputTable. Finally vtkWeakPointer may be set using setColorTable if it is, it will point to that instance until it is destroyed.

**POINTER CLASS INTERACTION**
The pointer classes all work with each other. The vtkNew class is ideal for allocation of new objects in a very concise form. Since it just decrements the reference count when it goes out of scope, it works as expected when used with the other two pointer classes.

vtkNew<vtkTable> nTable; // Reference count of 1
vtkSmartPointer<vtkTable> spTable = nTable.GetPointer(); // Reference count of 2
vtkWeakPointer<vtkTable> wpTable = nTable.GetPointer(); // Reference count of 2

Once the nTable and spTable objects go out of scope, the reference count would drop to zero, and the wpTable would be set to null. There is no assignment of objects to vtkNew, and so other pointers cannot be assigned to it. Both the vtkSmartPointer and vtkWeakPointer have implicit casting to the pointer type of the class and define the equality operator between them, and so the GetPointer() call is not strictly necessary when converting between these two pointer types and raw pointers.

**CONCLUSIONS**
Each of the classes has online documentation, and each satisfies a specific need in VTK. They should make memory management in your classes and applications simpler if used correctly, reducing line counts and decreasing code complexity. They are also suitable for use in classes, and only their header needs to be included in the declaration file. The new vtkNew class is especially useful in things like tests and small sample applications where several classes must be allocated. The vtkSmartPointer and vtkWeakPointer complement one another where pointers must be held on to for later use.

**HOSTING BINARY FILES ON MIDAS TO REDUCE GIT REPOSITORY SIZE**

Many of the projects to which Kitware contributes have recently switched to using the distributed version control system Git as a replacement for older, centralized version control systems such as SVN or CVS. Among these projects are CMake, ITK, MIDAS, and VTK. One of the key advantages of these version control systems is the ability to develop projects using a "branchy" workflow that allows developers to work on related changes on a separate "topic branch" and commit new features onto a separate branch that can be tested and made stable before being integrated into the "master" branch, which is considered stable. There are, however, challenges to using Git, most notably the saving of each version of a binary file and the resulting large repository size, which is addressed in this article.

**DISTRIBUTED VCS**
The major difference in using a distributed VCS is that the entire repository history is stored locally on each user’s machine, instead of being stored in only one central server. When changes are committed to any text file, such as a source file, the changes are stored as the line-by-line difference between the two files. For any binary file, however, it’s much more difficult to store the difference, so instead Git simply stores each version of binary files in the repository in their entirety. Naturally this causes the size of the history to become unacceptably large if the repository contains sizable binary files that change frequently. We needed an alternative place to store binary files outside the repository that could be referenced from within the source code.

Marcus Hanwell is an R&D engineer in the scientific visualization team at Kitware. He joined the company in October 2009, and has a background in open source, Physics and Chemistry. He spends most of his time working with Sandia on VTK, Titan and ParaView.
The solution we created uses MIDAS as the place to host the binary files. MIDAS provides a hierarchical organization of data on the server that can emulate a filesystem structure, and also provides access and administrative controls to the data in each directory. MIDAS also provides, via its web API, a mechanism for downloading a file stored on the server by passing the MD5 checksum of the file’s contents.

HOSTING AND REFERENCING BINARY FILES

In order to move files out of the source repository and onto MIDAS, the first step is to upload the files to the MIDAS server. Once the files (called “bitstreams” in MIDAS) have been uploaded to the server, we will remove them from the source code repository and replace each removed binary file with a “key file.” This key file acts as a placeholder for the real file; it simply contains the MD5 checksum of the actual file’s contents. A key file has the same name as the actual file, with a “.md5” extension added to the end.

To get the key file corresponding to a file stored on the MIDAS server, navigate in your browser to the item containing the desired bitstream. Click the checkbox that says “Advanced View,” and a link titled Download MD5 Key File will appear next to each bitstream in the list. These links can be used to download individual key files.

Alternatively, you can download all of the key files for an item at once using the Item menu at the top.

Choose “key files (.tgz)” or “key files (.zip),” depending which compression format you prefer, and the keys will be downloaded in a zipped directory to your machine. You can then unzip them and copy them into the source repository in place of the actual files. Each of these key files is text and is only 32 bytes, so the overhead of storing them in the repository is minimal.

The most common form of binary data in our source repositories is data used for automated testing, such as baseline and input images. To allow the MIDAS key files to be used as placeholders for real files, we created a CMake macro that’s a thin wrapper around the usual "add_test" command. The main difference is that instead of referring to actual binary files in the source tree, you can call this macro with a reference to a placeholder file. Then, at test time, all of the files referenced as test arguments will be downloaded from MIDAS just prior to running the test, and the test will be run on the files that have been downloaded (by convention into the build tree).

To use this macro, you’ll need to add the following line in your CMakeLists code:

```
include(MIDAS)
```

Additionally, you need to make sure that MIDAS.cmake is in your CMake module path and set a few CMake variables prior to running the macro.

```
set(MIDAS_REST_URL "http://midas.kitware.com/api/rest")
set(MIDAS_KEY_DIR "${PROJECT_SOURCE_DIR}/Testing/Data")
set(MIDAS_DATA_DIR "${PROJECT_BINARY_DIR}/Testing/Data")
```

This is an optional variable. This directory is the location where the actual files will be downloaded at test time. By convention, this should be placed outside of your source tree so as not to pollute it.

Once you have set these variables, you may call the new macro, midas_add_test(). This macro should be called with the same parameters as you’d call add_test, but substitute any references to moved files with a new type of reference to the placeholder file. An example is shown here, taken from the BRAINSTools module of Slicer4. The original call to add_test was:

```
add_test(NAME ${BRAINSFitTestName} COMMAND ${LAUNCH_EXE} $<TARGET_FILE:${BRAINSFitTestName}> --compare $(${BRAINSFitTestName}.result.nii.gz $(${BRAINSFit_BINARY_DIR})/Testing/${BRAINSFitTestName}.test.nii.gz --compareIntensityTolerance 7 --compareRadiusTolerance 0 --compareNumberOfPixelsTolerance 777 BRAINSFitTest --costMetric MMI --failureExitCode -1 --writeTransformOnFailure
```

The macro communicates with MIDAS via its rest API, so you must specify the URL of the server from which you will download the data.
The references to binary files in the source directory have been changed to refer to the key file instead, and wrapped with the MIDAS(...) keyword to let the macro know that the files need to be downloaded. When you configure the project, calling the midas_add_test macro actually creates two tests. The first of these is the fetchData test, which performs the download of all the data required by the actual test, which is then added by the macro. The actual test is made to explicitly depend on the fetchData test, which makes this macro safe for use in parallel-CTest environments.

Another use case is for tests that pass a directory as an argument instead of a single file. This is the case in Slicer4’s DicomToNrrdConverter module, which tests against many DICOM directories containing a large number of binary files. There is an additional signature for this use case: MIDAS_DIRECTORY(...) Pass in the name of a directory that contains multiple key files. All of the key files will be replaced by the corresponding actual files at test time and the directory where they were downloaded will be passed to the test as an argument.

**NETWORK CONNECTIVITY**

If you want to download all of the required testing data in anticipation of losing your network connectivity, run CMake on your project to configure the test set, and then in the build directory, run the following command:

```bash
cctest -R _fetchData
```

This will fetch all of the data needed for the tests. The data only needs to be downloaded once; subsequent calls to run the tests will reference the data that was previously downloaded to your machine, so no further network connectivity is required.

**CONCLUSION**

The midas_add_test macro is designed so that test developers will have an easy time converting their existing tests and managing the synchronization of data between the MIDAS server and their source repositories. Those running the tests will not have to do anything different except to ensure the data is downloaded once they have network connectivity.

Full documentation for this macro can be found at [http://www.kitware.com/midaswiki/index.php/MIDAS%2BCTest](http://www.kitware.com/midaswiki/index.php/MIDAS%2BCTest)

---

**KITWARE NEWS**

**SPINE MEDICAL IMAGING**

The 2011 SPIE Medical Imaging conference was held in Orlando from February 12-18. Kitware, once again, had a strong showing.

Dr. Andinet Enquobahrie was co-host of the IGSTK Users Group Meeting that was held in conjunction with the conference. Andinet presented on PET-CT support in IGSTK and users presented a diverse set of ongoing applications, clinical trials, and research algorithms.

Dr. Michel Audette was lead author on a paper involving ParaView, Medical Imaging, and Surgical Simulation in partnership with JHU and UNC – “Approach-specific multi-grid anatomical modeling for neurosurgery simulation with ParaView.” The other authors are Denis Rivière, NeuroSpin (France); Charles Law, Luis Ibanez, Stephen R. Aylward, Julien Finet, Kitware, Inc. (USA); Xunlei Wu, Duke University (USA); and Matthew Ewend, The University of North Carolina at Chapel Hill (USA).

Lastly, Dr. Stephen Aylward co-organized the third Live Demonstration Workshop that is held in conjunction with...
the Computer-Aided Diagnosis track. The year's workshop (http://www.kitware.com/workshops/SPIE_CAD_2011.html) featured 15 outstanding demonstrations that spanned a variety of organs (heart, lung, colon, breast), modalities (mammography, ultrasound, acoustic, MR), and diseases (cancer, COPD, mental disorders). Over 120 people attended these demonstrations, which will be repeated next year.

IGSTK USER GROUP MEETING
The fifth IGSTK Users Group meeting was held in Orlando, Florida in conjunction with the SPIE Medical Imaging conference. Each year the meeting is an opportunity for collaborators and users to gather and discuss their projects and use of IGSTK.

In addition to Georgetown University and Kitware, there were six participating institutions
1. Princess Margaret Hospital Toronto, Ontario
2. THE PERK LAB, Queen's University, Kingston, Ontario
3. Humanoids and Intelligence Systems Lab, Karlsruhe Institut of Technology (KIT) Karlsruhe, Germany
4. Innovation Center Computer Assisted Surgery (ICCAS) , University of Leipzig Leipzig, Germany
5. 4D Visualization Laboratory, Innsbruck Medical University Innsbruck, Austria
6. Hospital Italiano de Buenos Aires, Buenos Aires, Argentina

The morning of the meeting was dedicated to a series of presentations by core developers on new developments that were released as part of IGSTK 4.4. In the afternoon, IGSTK users presented their applications. This year the presentations covered image-guided head and neck surgery systems, a facet joint injection training simulator, and neurosurgery.

One of the highlights of the meeting was a presentation by Sebastian Ordas, who owns a small company in Buenos Aires, Argentina. He has a proprietary neurological image-guided navigation system developed using IGSTK. The system is currently under clinical trial and has been tested in 16 clinical cases thus far, including big and superficial tumor resection, tumor biopsy and epilepsy. All the presentations were well done and the slides can be found on the IGSTK wiki at http://public.kitware.com/IGSTKWIKI/.

CTK HACKFEST
Kitware co-hosted the third annual CTK Hackfest from February 7-11 at the Franklin Hotel in Chapel Hill, NC. In preceding years, the event was held in Washington, DC and Barcelona, Spain, with a pre-Hackfest in Heidelberg, Germany. CTK, the Common Toolkit, provides a unified set of basic programming constructs that are useful for medical imaging applications development and facilitates the exchange and combination of code and data. The goal of the hackfest is to spend several days focusing on the most pressing challenges and collaborate to improve the toolkit.

This year there were returning participants from leading research groups around the world, including the teams of MITK (Heidelberg, Germany), MAF (Bolona, Italy), DreamTk (INRIA, France); Steve Pieper (Isomics, Boston) and Kitware attendees Dave Partyka, Jean-Christophe Fillion-Robin, Julien Finet and Stephen Aylward. In addition, this hackfest welcomed new participants including Lawrence Tarbox of the University of Washington St. Louis, an active participant in the DICOM Standards Committee and Nicholas Herlambang from AZE, Japan.

During this event, significant work was done on CTK, with a large emphasis on a DICOM PACS query/retrieve application (with reusable Qt widgets/database); a VTK-free Qt simple image viewer (optionally supporting DICOM image dataset with Dcmtk dependency); data transfer and application hosting (supporting DICOM Part 19 protocols) and xmlrpc event bus. A new development is that CTK, and inherently Slicer, will soon be using the Kitware Qt GUI testing framework for automatic GUI testing.

The CTK hackfest was funded, in part, by:
• The Neuroimage Analysis Center (NAC, PI: Kikinis) P41 RR 013218
• The National Alliance for Medical Image Computing (NA-MIC, PI: Kikinis) 1U54EB005149-01
• Image Registration for Ultrasound-Based Neurosurgical Navigation Project (PI: Aylward, Wells) 1R01CA138419-01

VIZBI 2011 WORKSHOP
The VIZBI 2011 workshop was held March 16-18, 2011 at the Broad Institute in Cambridge, MA. The workshop was a review of the state-of-the-art in biological visualization and
highlighted current and future challenges in visualization across the broad range of biological research areas. VIZBI takes a unique approach to the science and dissemination of biological information. It brings together biologists, geneticists, computer scientists and artists in an exploration of different ways to analyze and present unique information. The Broad Institute is uniquely suited to hosting this type of intimate conference.

This was the second year of the workshop, which featured four keynote addresses, six scientific sessions and four poster sessions spanning a range of topics, from the genome, to proteins, cells, anatomy and on to populations and evolution. Along the way the workshop addressed outreach, teaching and general visualization principles. Following the workshop, the organizers held a set of tutorial sessions that combined instruction in various visualization tools with hands-on experience using those tools on the students' own data. Kitware was invited to present a tutorial, and Wesley Turner presented on the use of VTK and ParaView for medical and biological visualization.

IS&T/SPIE ELECTRONIC IMAGING CONFERENCE
The IS&T/SPIE Electronic Imaging conference was held in San Francisco from January 23–27 this year. This conference is the must-attend event for all aspects of electronic imaging, including imaging systems, image processing, image quality, and algorithms. It featured multiple, parallel tracks on 3D Imaging, Interaction and Measurement; Imaging, Visualization, and Perception; Image Processing; Digital Imaging Sensors and Applications; Multimedia Processing and Applications; and Visual Information Processing and Communication. This year Kitware joined forces with Portola Pharmaceuticals to present a poster and live demonstration on “Tracking flow of leukocytes in blood for drug analysis,” (authors A. Basharat, W. D. Turner, G. Stephens, B. Badillo, R. Lumpkin, P. Andre, and A. Perera). The work was presented by Arslan Basharat and Wesley Turner of Kitware.

KITWARE AWARDED NASA GRANT
It was announced in March that NASA awarded SBIR funding to Kitware to further develop ParaView to meet the needs of ultrascale visualization. The software will address critical issues in order to enable real-time investigation of extremely large datasets using massive distributed memory architectures with up to 100,000 cores.

We will be collaborating with California-based SciberQuest, leaders in kinetic modeling of space plasmas, to complete the project. This partnership enables us to work with real world data from petascale simulations directly relevant to NASA’s missions and scientific goals, and allows the development to be guided by the ultimate users of the software.

In this investigative phase we will identify scaling bottlenecks in ParaView, which is currently used by NASA to explore the results of trillion element particle simulations on the Pleiades supercomputer. As the number of processors scales up past ten thousand, we anticipate that the most critical issues will be data IO, architectural overhead, and the compositing of the partial results. While the Phase I effort of this project is limited to developing prototypes and only select improvements will be incorporated into the software, if the Phase II effort is funded the complete range of improvements will be merged into ParaView and the underlying Visualization Toolkit (VTK), which will benefit tens of thousands of researchers world-wide.

NA-MIC PROJECT WEEK
The NA-MIC community met for their 7th annual All Hands Meeting and External Advisory Board meeting January 7-10 in Salt Lake City, Utah. With the recent renewal of the NIH National Center of Biomedical Computing NA-MIC grant, this AHM/EAB meeting marked the beginning of the next four years of what will ultimately be a 10-year project. Well over 100 people were in attendance, including science officers and other NIH officials. One common theme to this meeting was the impact Slicer was having throughout the world. That impact is well illustrated by the figure below - it shows the locations to which Slicer has been downloaded.

Kitware was present in force and made significant contributions to the meeting and Slicer. Stephen Aylward, Jean-Christophe Fillion-Robin, Julien Finet, Danielle Pace, Dave Partyka, Zach Mullen and Will Schroeder were in attendance. Some of the areas where Kitware was focusing its efforts included the preview release of Slicer 4 (a Qt-based rewrite of Slicer spearheaded by Julien Finet and Jean-Christophe Fillion-Robin); the beta release of TubeTK (an adjunct toolkit that provides sliding-organ registration and vascular analysis capabilities, by Danielle and Stephen); a 64-bit Slicer Windows build by Dave Partyka and Steve Pieper; DICOM/MIDAS data integration by Zach Mullen; and two new reformat widgets being developed by Will Schroeder. We are looking forward to four more full and rewarding years with NA-MIC, and many more years with Slicer beyond those.

VTK SELECTED FOR GOOGLE SUMMER OF CODE
The Visualization Toolkit (VTK) has been accepted for the 2011 Google Summer of Code, with Kitware acting as the mentoring organization. This program encourages student participation in open source communities through three-month paid development projects. Students interested in the program apply to work on a specific project and work with a mentor at the organization over the course of the summer. This global program gives students the opportunity to work on real-world software projects and provides mentoring organizations with potential new developers. Additionally, since all development is open source, the projects grow and code is contributed back to the community.

Of 417 applications this year, Google selected 175 open source projects for participation, 50 of which are new to the program. Google has posted a list of all accepted projects. Kitware has several project ideas for students, such as the development of new 2D charts, chemistry visualization, volume rendering in WebGL, AMR volume rendering and Apple iOS support for ParaViewWeb.
KITWARE WINS IARPA GRANT

Kitware was awarded a $2 million, one-year grant by the Intelligence Advanced Research Projects Activity (IARPA) to develop a system prototype called General Engine for Indexing Events (GENIE) to address the ALADDIN challenge.

GENIE’s primary purpose is to enhance the capabilities of Automated Low-Level Analysis and Description of Diverse Intelligence Video (ALADDIN), an IARPA program that focuses on finding activities in un constrained or “open-source” video collections, like YouTube, by leveraging the most promising, relevant technologies to continuously search through millions of new and archived videos on the web to detect the few that contain meaningful, operational, and salient information. It will also serve as a powerful platform for conducting novel research in event recognition and search by providing software tools that simplify quantified evaluation of research algorithms.

The idea behind this program is to recognize specific activities based on evidential descriptors contained within the video such as location, objects and activities being performed (i.e., making a cake, hitting a baseball or constructing a shelter).

The GENIE solution requires a strong capability in multimedia content description and event modeling, as well as the ability to architect a scalable, end-to-end solution. These characteristics, beyond ALADDIN’s current state-of-the-art video recognition technology system, are necessary to overcome the key challenge in web video recognition: the depiction of virtually any event and object in a limitless number of styles, qualities and scenes.

“By enhancing ALADDIN’s capabilities, GENIE will have a revolutionary impact on the automated analysis of web-based videos and, undoubtedly, contribute to the creation of additional military and domestic applications,” said Lynn Bardsley, Kitware’s program manager for computer vision.

“This award is a testament to Kitware’s progress in developing unique computer vision and intelligence software solutions.”

Dr. Amitha Perera, a Technical Leader at Kitware and the project leader on GENIE, has assembled a world-class team of researchers and partners for GENIE Phase I including, Honeywell Labs, a leading defense technology company and market leader in commercial video surveillance systems and large-scale video indexing. Kitware has previously worked with Honeywell Labs on the DARPA VIRAT and PerSEAS programs. Additional academic collaborators include Stanford University, Georgia Tech, the University at Buffalo and Simon Fraser University.

KITWARE PUBLICATIONS UPDATE

The printing of the VTK Textbook is complete and it is again available for purchase through the Kitware online store.

Additionally, we have added new ways to provide feedback on our books through our website via a bug tracker or email. We encourage our readers to visit the website and send us any comments or suggestions so that we can improve the next editions of each of our books.

NEW ONLINE HOME FOR THE SOURCE

Based on reader feedback, we have created a new digital version of the Source. This new digital adaptation is in the style of a blog, and readers will now be able to post comments on the articles and subscribe to specific topics and

TIBBETTS AWARD

Kitware received a Tibbets Award for its Software Toolkit for Image-Guided Surgery (IGSTK) Phase I and II STTRS. The award, which recognizes companies who represent excellence in achieving the mission and goals of the SBIR and STTR programs, is named for Roland Tibbets. Tibbets is considered the father of the Small Business Innovation Research (SBIR) program, which he began as an experimental project at the National Science Foundation in the early 1980’s.

IGSTK, our tool for research involving minimally invasive image-guided medical procedures, is being used in investigations into new surgical techniques that may improve surgical accuracy and precision, increase a surgeon’s ability to confidently treat challenging and complex pathologies, decrease surgical trauma, and reduce recovery time for patients.

The IGSTK STTR is a collaborative research project between Kitware’s New York and North Carolina offices and the Computer Aided Interventional and Medical Robotics (CAIMR) group, which is led by Dr. Kevin Cleary at Georgetown University.

Dr. Andinet Enquobahrie, with Roland Tibbetts, the father of the SBIR program, accepted the award at a ceremony in Washington, DC on behalf of Kitware and the Image Guided Surgery Toolkit team.

NA-MIC REGISTRATION RETREAT

From February 19-23, Danielle Pace, Will Schroeder, and Stephen Aylward attended a “Registration Retreat” in San Juan, Puerto Rico. This event brings together registration algorithm researchers from the Neuroimage Analysis Center, NA-MIC, the National Center for Image Guided Therapy, and other invited researchers.

The meeting consisted of a daily mix of 2-4 hours of group discussions followed by break-out meetings. There were 13 participants, including Tina Kapur, Brian Avants (an ITK collaborator), Guido Gerig (University of Utah), Kilian Pohl (University of Pennsylvania), Torsten Rohlffing (SRI International), William Wells (MIT), Matthew Toews (McGill University), Gregory Sharp (MGH), and C-F Westin (Harvard).

The intended product of the meeting is a journal article that presents the unmet challenges and opportunities in the clinical application of registration algorithms. The group made good progress on the paper, which will be led by Tina Kapur and Stephen Aylward.

14
The success of this first course has prompted the Kitware team to teach an advanced, two-day VTK course to be held April 7-8 in Lyon. This course will cover an overview of the VTK architecture, VTK visualization pipelines, information visualization techniques, writing custom filters and parallel processing and rendering techniques.

If you are interested in one of our courses, please visit our website to see our full range of offerings.

KITWARE CELEBRATES ITS ANNIVERSARY
This March Kitware celebrated its 13th anniversary. The company has grown significantly since 1998, from five co-founders to nearly 100 employees, with offices in three cities on two continents. There are several factors that I believe have enabled our success, which I would like to share.

First, at Kitware we have always been fortunate in that we have an attitude of service, doing what each of us does best and helping each other. We have been able to take the best in each of us to create something better than any one of us could do on our own. We also had great early customers to fuel our growth, including Terry Yoo at the National Library of Medicine who established the ITK project in 1999, Jim Ahrens at Los Alamos who pushed for parallelizing VTK architecture, VTK visualization pipelines, information visualization techniques, writing custom filters and parallel processing and rendering techniques.

In the coming years we will continue the practices that have enabled our success: service to our co-workers and collaborators; finding supportive customers; growing our leading-edge technology portfolio; extending our open source outreach; and refining our business processes. Moreover we will continue to hire some of the best talent we can find, and we will do all we can to unleash the diverse potential of the people at Kitware.

-Will Schroeder, President and CEO

UPCOMING CONFERENCES AND EVENTS

2011 Northeast Bioengineering Conference
April 1-3 at Rensselaer Polytechnic Institute in Rensselaer, NY. Rick Avila will be giving a keynote speech and Will Schroeder will be participating in the CEO Forum discussion.

OpenFOAM Workshop
June 13-16 at Penn State University in State College, PA. Dave DeMarle will be teaching an advanced ParaView tutorial on June 13.

NA-MIC Summer Project Week
June 20-24 at MIT in Cambridge, MA. This project week focuses on hands-on R&D for applications in image-guided therapy and other areas of biomedical research. Will Schroeder will be attending.

IEEE Computer Vision Pattern Recognition (CVPR)
June 21-25 in Colorado Springs, CO. Anthony Hoogs is co-organizing two workshops, the Workshop on the Activity Recognition Competition and the Workshop on Camera Networks and Wide Area Scene Analysis. Luis Ibáñez, Matt Leotta, Amitha Perera and Patrick Reynolds will be teaching the tutorial “ITK meets OpenCV: A New Open Source Software Resource for CV” on June 20th.

SIGGRAPH 2011
August 7-11 in Vancouver, BC, Canada. SIGGRAPH is the premiere international event focused on computer graphics and interactive techniques. Aashish Chaudhary will be attending.

If you would like to set up a time to meet with us at any of these events, please contact our office by phone at (518) 371-3971 or email at kitware@kitware.com.
NEW EMPLOYEES

Zak Ford
Zak joined Kitware in February as a new systems administrator. Prior to joining, he worked as a systems administrator and web application developer for Gui Productions, Inc. Zak studied Computer Science at the University at Albany.

Tami Grasso
Tami joined Kitware in April as the new office assistant. Prior to joining Kitware, Tami worked in a customer service role for Time Warner Cable and was a co-owner of Capitaland Flooring Company, where she was responsible for maintaining company records, providing customer support and developing marketing materials.

Michelle Kimmel
Michelle joined Kitware in February as an accountant on the finance team. She received her B.S. in Accounting from the University of Maryland. Prior to joining Kitware, Michelle held accounting positions with Seton Health Systems, Inc. and the Walden Golf Club, where she was responsible for preparing monthly journal entries, managing account reconciliations, processing accounts payable and assisting in preparation of grant paperwork.

John Tourtellot
John joined Kitware in February as an R&D engineer on the computer vision team. He received his B.S. cum laude and M. Eng. degrees in electrical engineering from Rensselaer Polytechnic Institute. Prior to coming to Kitware, John worked as a senior software engineer at Simmetrix, where he developed component software for simulation based design.

George Zagaris
George joined Kitware in January as an R&D engineer for the scientific computing group. He earned his B.S. (Honors) and M.S. in computer science from the College of William and Mary, where his M.S. research focused on parallel unstructured mesh generation for CFD applications and was partially funded by the NASA Graduate Student Research Program Fellowship at NASA Langley Research Center (LaRC).

KITWARE INTERNSHIPS

Kitware Internships provide current college students with the opportunity to gain hands-on experience working with leaders in their fields on cutting-edge problems. Our business model is based on open source software—an exciting, rewarding work environment.

At Kitware, you will assist in the development of foundational research and leading-edge technology across our five business areas. We are actively recruiting interns for the summer. If you are interested in applying, please send your resume to internships@kitware.com.

EMPLOYMENT OPPORTUNITIES

Kitware is seeking talented, motivated and creative individuals to become part of our team. As one of the fastest growing companies in the country, we have an immediate need for software developers, 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 fields of visualization, medical imaging, computer vision, 3D data publishing and technical software development. 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, visa processing; a generous compensation plan; profit sharing; and free drinks and snacks.

Interested applicants should send a cover letter and resume to jobs@kitware.com for their immediate consideration.