We are pleased to share our latest developments in the creation and support of open-source software and state-of-the-art technology, including new releases for CMake, ITK, RGG, and Slicer.

In November, we attended Supercomputing 2014. The event was highly successful, and we made several announcements in conjunction with the conference. We announced that we are working with NVIDIA to drive massive acceleration in HPC visualization, that we received three ASCR awards to fuel scientific discovery, and that we have collaborated with TACC to become an Intel® Parallel Computing Center.

We attended many other events including IEEE Vis, IEEE Nuclear Science Symposium & Medical Imaging Conference, and Qt Developer Days 2014. A list of our upcoming events can be found on www.kitware.com/events. If you would like to set up a time to meet with us at any of the listed events to discuss employment opportunities, potential collaboration, or consulting services, please contact kitware@kitware.com.

**RGG 1.1 RELEASED**

Kitware is pleased to announce the release of Reactor Geometry Generator (RGG) 1.1. RGG allows engineers to design advanced nuclear reactor core models and to create proper analysis meshes based on the models.

Version 1.1 utilizes next-generation rendering improvements made to VTK, allowing the designer to model and interact with nuclear cores consisting of up to 100,000 pins and rods, while the improved GUI provides separate schematic, 3D, and mesh views. In terms of model and mesh generation improvements, MeshKit’s AssyGen and CoreGen tools are now bundled with the application, eliminating the need for users to build them separately. In addition, version 1.1 of RGG supports running AssyGen and CUBIT in parallel, which reduces model and mesh construction time. Another improvement is that non-RGG created meshes in Exodus format can now be displayed in RGG. Finally, users can export subsets of their meshes based on material.

To learn more and to download RGG 1.1, please visit http://www.computationalmodelbuilder.org/rgg/.
ITK 4.7.0 RELEASED
ITK 4.7.0 was released in December. Congratulations and thank you to everyone who contributed to this release.

Documentation improvements for the release include updated Software Guide registration examples to the ITKv4 framework and a clean Software Guide dashboard build. ImageIO improvements include PNG sCAL unit support, enhanced support for system GDCM, and updates to MetalO. In addition, several infrastructure and filtering improvements were made. For example, there is a new method to erase a MetaDataDictionary entry, a new global method to set physical space tolerance, and new experimental thread pool support.

The release also features important bug fixes, wrapping improvements, performance improvements, registration improvements, and new remote modules.

To try the new software, please visit ITK’s download page.

Triangle Mesh Subdivision image created using code from the release. Image courtesy of Zhu W., School of Psychiatry, University Of New South Wales. http://hdl.handle.net/10380/3307

CMAKE 3.1.0 RELEASED
Kitware is pleased to announce the release of CMake 3.1.0. In this release, support for Windows Phone and Windows Store was added to Visual Studio 11 (2012) and above Generators. In addition, NVIDIA Nsight Tegra support was added to Visual Studio 10 (2010) and above Generators.

Also new in CMake 3.1.0: The “if” command no longer automatically dereferences variables named in quoted or bracket arguments; the target property “SOURCES” supports “Generator Expressions”; and the syntax for Variable References and Escape Sequences was simplified to allow for faster parsing.

Furthermore, a new “target_compile_features” command allows for populating target-based compile features; ctest now supports Intel coverage files with the “codecov” tool; and CPack supports lzma compressed archives.

The source and binaries for CMake 3.1.0 are available on the CMake download page, and documentation can be found on http://www.cmake.org/documentation/.

SLICER 4.4 RELEASED
The Slicer community recently announced the release of Slicer 4.4. Developed in collaboration with the National Association for Medical Image Computing (NA-MIC) and the broader Slicer community, Slicer 4.4 is a free, open-source (BSD-style license) software platform for the segmentation, registration, visualization, and analysis of medical images for preclinical and clinical research involving disease detection, diagnosis, treatment planning, response monitoring, population analysis, and image-guided therapy. Available for Linux, MacOSX, and Windows systems, Slicer is extensible, with powerful plug-in capabilities for adding algorithms and creating custom applications.

Features of Slicer include support for multi-modality imaging such as MRI, CT, ultrasound, nuclear medicine, and microscopy; a bidirectional interface for devices; and multi-organ support from head to toe. Spearheaded at Kitware by Jean-Christophe Fillion-Robin, Slicer 4.4 introduces an improved App Store, known as the Extensions Manager, for adding plug-ins to Slicer. Currently, over 50 extensions are available.

Other major Slicer 4.4 features include an improved Transforms module with support for non-linear transforms and visualization of transforms in 2D and 3D, a new Subject hierarchy module to efficiently organize and manipulate data loaded in Slicer, and a new Landmark registration module for interactive registration. In addition, Compare Volumes was added to allow for the creation of an overview of one or more volumes, and the user interface was improved for Colors, Markups, DICOM, and Interactive Editor.

In total, Slicer 4.4 includes close to 400 feature improvements and bug fixes for enhanced performance, utility, and stability. For a complete list of highlights, please see Slicer’s Wiki page, and to download version 4.4 of Slicer, please visit http://download.slicer.org.
RAY-CASTING & RAY-TRACING WITH VTK

Adamos Kyriakou (IT’IS Foundation)

VTK has long evolved beyond just visualization. It offers some amazing functionality that just cannot be found elsewhere. Two examples are the ‘ray-casting’ and, consequentially, ‘ray-tracing’ capabilities provided by the vtkOBBTree class. In this article, I would like to introduce these capabilities and show examples of ray-casting and ray-tracing performed exclusively through Python, a dash of NumPy, and VTK.

**Disclaimer:** The ray-casting and ray-tracing examples I will be presenting here are severely condensed versions of my posts “Ray Casting with Python and VTK: Intersecting lines-rays with surface meshes” and “From Ray Casting to Ray Tracing with Python and VTK” that appear on my blog [1]. If they pique your interest, please visit the aforementioned posts, where you can find all of the material and code (in the form of IPython Notebooks), as well as an excruciating amount of detail pertaining to each aspect of the process, as these posts were written for people with little to no experience in VTK.

**RAY-CASTING VS. RAY-TRACING**

I would like to emphasize a pivotal difference between ‘ray-casting’ and ‘ray-tracing.’ In the case of the former, we only ‘cast’ a single ray, test for its intersection with objects, and retrieve information regarding the intersection. Ray-tracing, on the other hand, is more physically accurate, as it applies laws of physics (e.g., reflection, refraction, attenuation, etc.) to the rays to ‘trace’ (i.e., follow) that ray and its derivative rays. However, ray-casting is the natural precursor to ray-tracing, as it tells us with what part of which object the ray intersects and provides all necessary information to cast subsequent rays.

**THE VTKOBBTREE CLASS**

The star of this post is the vtkOBBTree class, which generates an oriented bounding-box (OBB) ‘tree’ for a given geometry under a vtkPolyData object. Upon generation of this OBB tree, the vtkOBBTree class allows us to perform intersection tests between the mesh and the lines of finite length, as well as intersection tests between different meshes. It can then return the point coordinates where intersections were detected, as well as the polydata cell IDs where the intersections occurred.

**RAY-CASTING WITH VTKOBBTREE**

For this demonstration, we are assuming that we have a surface model of a human skull stored in a .stl file, whose contents we have loaded into a vtkPolyData object, named mesh, through the vtkSTLReader class. A rendering of this model through the vtkPolyDataMapper class can be seen in Figure 1.

![Figure 1. Rendering of the surface model of a human skull, which we will use to demonstrate ray-casting.](image1)

The center of the skull is centered around the cartesian origin, i.e., the (0.0, 0.0, 0.0) point. Now let’s assume we want to cast a ray emanating from (100.0, 100.0, 0.0) and ending at (0.0, 0.0, 0.0) and retrieve the coordinates of the points where this ray intersects with the skull’s surface. A rendering including the ray, prior to actually casting it, can be seen in Figure 2.

![Figure 2. The ray that will be tested for intersection with the skull model. The ‘source’ point of the ray is rendered as red, while the ‘target’ point is rendered as green.](image2)
Prior to intersection, we need to create and initialize a new vtkOBBTree with the vtkPolyData object of our choice. In our case, this is called mesh and is done as follows:

```python
obbTree = vtk.vtkOBBTree()
obbTree.SetDataSet(mesh)
obbTree.BuildLocator()
```

Note the call to the BuildLocator method, which creates the OBB tree.

That’s it! We now have a world-class intersection tester at our disposal. At this point, we can use the IntersectWithLine method of the vtkOBBTree class to test for intersection with the aforementioned ray. We merely need to create a vtkPoints object and a vtkIdList object to store the results of the intersection test.

```python
points = vtk.vtkPoints()
cellIds = vtk.vtkIdList()

code = obbTree.IntersectWithLine((100.0, 100.0, 0.0),
                                 (0.0, 0.0, 0.0),
                                 points,
                                 cellIds)
```

As mentioned above, the points and cellIds now contain the point coordinates and cell IDs with which the ray intersected as it was emanated from the first point, i.e., (100.0, 100.0, 0.0), onto the second point, i.e., (0.0, 0.0, 0.0), in the order they were ‘encountered.’ The return value code is an integer, which would be equal to 0 if no intersections were found. A rendering showing the intersection points can be seen in Figure 3.

The Python package pycaster [2] wraps the functionality shown above, assuming no VTK experience, and provides additional methods to calculate the distance a given ray has ‘traveled’ within a closed surface. It is currently being served through PyPI. The repository [3] can be found on BitBucket.

As I mentioned at the beginning of this article, the entire process shown above, including all of the material and code needed to reproduce it, is detailed in my post "Ray Casting with Python and VTK: Intersecting lines/rays with surface meshes" [4].

### RAY-TRACING WITH VTKOBBTREE

Now, in order to perform ray-tracing, we can take the lessons learned from ray-casting and apply them to a more convoluted scenario. The rationale behind ray-tracing with the vtkOBBTree class is the following:

- Cast rays from every 'ray source' and test for their intersection with every ‘target’ mesh in the scene.
- If a given ray intersects with a given ‘target,’ then use the intersection points and intersected cells to calculate the normal at that cell, to calculate the vectors of the reflected/refracted rays, and to cast subsequent rays off the target.
- Repeat this process for every ray cast from the ‘ray source.’

Let’s assume a scene is comprised of a half-sphere dubbed sun, which will act as the ray-source, and a larger nicely textured sphere called earth, which will be the target of those rays. This ‘environment’ can be seen in Figure 4.

![Figure 3. Result of the ray-casting operation and intersection test between the skull model and the ray. The blue points depict the detected intersection points.](image)

![Figure 4. Scene defined for the ray-tracing example. The yellow half-sphere, sun, acts as the ray-source, while the textured sphere, earth, will be the target of those rays.](image)
In this example, we will be casting a ray from the center of each triangular cell on the sun's surface along the direction of that cell's normal vector. The cell-centers of the sun were calculated through the \texttt{vtkCellCenters} class and stored under the \texttt{pointsCellCentersSun} (of type \texttt{vtkPolyData}). The cell-normals of the sun were calculated through the \texttt{vtkPolyDataNormals} class and stored under \texttt{normalsSun} (of type \texttt{vtkFloatArray}). A rendering of the cell-centers as points and cell-normals as glyphs through the \texttt{vtkGlyph3D} class can be seen in Figure 5.

![Figure 5. Rendering of the cell-centers of the sun that will act as source-points for the rays and the cell-normals along which the rays will be cast.](image)

Similarly to what was done in the previous example, prior to ray-casting, we first need to create a \texttt{vtkOBBTree} object for the earth:

```python
obbEarth = \texttt{vtk.vtkOBBTree()}
obbEarth.SetDataSet(earth.GetOutput())
obbEarth.BuildLocator()
```

Now, since we will be casting a large number of rays, let's wrap the \texttt{vtkOBBTree} functionality in two convenient functions:

```python
def isHit(obbTree, pSource, pTarget):
    code = obbTree.IntersectWithLine(pSource, pTarget, None, None)
    if code==0:
        return False
    return True
def GetIntersect(obbTree, pSource, pTarget):
    points = \texttt{vtk.vtkPoints()}
    cellIds = \texttt{vtk.vtkIdList()}
    code = obbTree.IntersectWithLine(pSource, pTarget, points, cellIds)
    pointData = points.GetData()
    noPoints = pointData.GetNumberOfTuples()
    noIds = cellIds.GetNumberOfIds()
    pointsInter = []
    cellIdsInter = []
    for idx in range(noPoints):
        pointsInter.append(pointData.GetTuple3(idx))
        cellIdsInter.append(cellIds.GetId(idx))
    return pointsInter, cellIdsInter
```

The \texttt{isHit} function will return True or False, depending on whether a given ray intersects with \texttt{obbTree}, which, in our case, will only be \texttt{obbEarth}.

The \texttt{GetIntersect} function simply wraps the functionality we saw in the first example. In a nutshell, it will return two list objects: \texttt{pointsInter} and \texttt{cellIdsInter}. The former will contain a series of tuple objects with the coordinates of the intersection points. The latter will contain the 'id' of the mesh cells that were 'hit' by that ray. This information is vital, as we will be able to get the correct normal vector for that earth cell and calculate the appropriate reflected vector through these ids, which we will see below.

At this point, we are ready to perform the ray-tracing. Let's take a look at a condensed version of the code:

```python
noPoints = pointsCellCentersSun.GetNumberOfPoints()
for idx in range(noPoints):
    pointSun = pointsCellCentersSun.GetPoint(idx)
    normalSun = normalsSun.GetTuple(idx)
    # Calculate the 'target' of the 'sun' ray based on 'RayCastLength'
    pointRayTarget = list(numpy.array(pointSun) +
                          RayCastLength*numpy.array(normalSun))

    if isHit(obbEarth, pointSun, pointRayTarget):
        pointsInter, cellIdsInter = GetIntersect(obbEarth, pointSun, pointRayTarget)
        # Get the normal vector at the earth cell that intersected with the ray
        normalEarth = normalsEarth.GetTuple(cellIdsInter[0])

        # Calculate the incident ray vector
```

```
vecInc = (numpy.array(pointRayTarget) -
          numpy.array(pointSun))
# Calculate the reflected ray vector
vecRef = (vecInc - 2*numpy.dot(vecInc,
          numpy.array(normalEarth)) *
          numpy.array(normalEarth))
# Calculate the 'target' of the reflected
# ray based on 'RayCastLength'
pointRayReflectedTarget =
    (numpy.array(pointsInter[0]) +
     RayCastLength*l2n(vecRef))

Please note that all rendering code was removed from the
above snippet. What is done above is the following:

We looped through every cell-center on the sun mesh
(pointSun), stored under pointsCellCentersSun.

We casted a ray along the direction of that sun cell’s normal
vector, stored under normalsSun. The ray emanates from
pointSun to pointRayTarget.

As the vtkOBBTree class only allows for intersection tests
with lines of finite length, not semi-infinite rays, the rays
cast in the code above are given a large (relative to the
scene) length to ensure that failure to intersect with earth
would only be due to the ray’s direction and not an insuf-
ficient length.

Every ray was tested for intersection with the earth through
the isHit function and the obbEarth object defined above.

If a ray intersected with the earth, the intersection test was
repeated through the GetIntersect function in order to
retrieve the intersection point coordinates and the inter-
sected cell IDs on the earth mesh. The intersection point
coordinates and the intersected earth cell normal vector
(normalEarth) were used to calculate the normal vector of
the reflected ray and cast that ray off the earth’s surface.

The cell normals on the earth’s surface were calculated
through the vtkPolyDataNormals class and stored under
normalsEarth (of type vtkFloatArray). This is the same way
that normalSun was calculated.

A render of the ray-tracing result can be seen in Figure 6.

As I mentioned at the beginning of this article, the entire
process shown above, including all of the material and code
needed to reproduce it, is detailed in my post “From Ray
Casting to Ray Tracing with Python and VTK” [6].

Figure 6. Result of the ray-tracing example. Rays cast from
the sun that missed the earth are rendered as white. Rays that
intersected with the earth are rendered as yellow. The intersec-
tion points, normal vectors at the intersected earth cells, and the
reflected rays can also be seen.

CONCLUSION
As you can see, VTK provides some little-known pearls that
offer fantastic functionality.

While the above examples fall short of real-world ray-tracing
applications, as one would need to account for effects like
refraction and energy attenuation, the sky is the limit!

REFERENCES
python-and-vtk-intersecting-linesrays-with-surface-meshes/
docs/2006--degreve--reflection_refraction.pdf
[6]  http://pyscience.wordpress.com/2014/10/05/from-ray-
casting-to-ray-tracing-with-python-and-vtk/

Adamos Kyriakou is an Electrical & Computer
Engineer with an MSc. in Telecommunications
and a Ph.D. in Biomedical Engineering. He is
currently working as a Research Associate in
Computational Multiphysics at the ITIS
Foundation (ETH Zurich), where his work and
research are primarily focused on computational algo-
rithm development, high-performance computing,
multi-physics simulations, big-data analysis, and medical
imaging/therapy modalities.
Challenging conditions, such as maritime environments, can make it difficult to safely and efficiently operate a crane mounted on a floating vessel when the suspended load begins to swing. Craft Engineering Associates, Inc, is developing a system for crane stabilization that is suitable for installation on a wide range of shipboard cranes. This involves sensing the dynamic motion of the vessel and the orientation of the crane to actuate the controls to reduce the adverse effects of the ship's motion. A major problem has been the effective measurement of any residual swinging of the load so that it may be damped. Kitware was approached to develop a LiDAR tracking solution to assist in this aspect of the crane stabilization project.

To track the position of the crane’s load, we paired a Velodyne LiDAR sensor with a customized version of the VeloView software. The crane, highlighted load, and surrounding environment are visible in the LiDAR sensor, as is shown in Figure 1.

The customized version of VeloView implements algorithms and a user interface to track the crane’s load. The software enables the user to initialize the location of the tracked object in the recording or in the live playback. Then, the position of the object is automatically updated at every frame, or lost tracking is reported. The tracked position can be exported to the control system using a serial port connection in addition to the live 3D visualization. This output will enable the crane control system to compensate for any unexpected motion of the crane’s load. To test the system, we attached the scanner to the side of the crane and recorded data during operation of the crane.

A video demonstration of the results is available on Vimeo [1]. The success of this project highlights the adaptability of the open-source VeloView software for the development of custom analysis solutions and workflows.

REFERENCES

Casey Goodlett is a Technical Leader at Kitware with expertise in business development, medical computing, and computer vision. Casey is currently focused on developing technology in point cloud processing in areas including visualization, registration, object detection, and software development.

J. Dexter Bird, III, is a Senior Design Engineer for Craft Engineering Associates, Inc. His specialty is the practical implementation of theory and advanced technology for electro-mechanical systems.

Figure 1: LiDAR scene showing crane, load, and surrounding environment. The block above the load is highlighted, and its position in meters relative to the sensor is reported.
INTRODUCING TANGELO

Roni Choudhury (Kitware), Jeff Baumes (Kitware)

Over the past two years, we have been developing Tangelo [1] in the pursuit of rich web applications for solving problems, both old and new. We recently passed 2,000 commits in the repository [2], and Tangelo will soon be ready for prime time. This article aims to inform you more about the technology with a high-level introduction to the Tangelo Web Framework. We then answer the question: Why Tangelo?

WHAT IS TANGELO?

At its heart, Tangelo is a web application development platform, geared primarily toward data analytics and visualization applications. This platform is made up of Tangelo itself (a web server with a twist), as well as JavaScript and Python APIs to help drive it. The platform also includes an ecosystem of plugins that provide support for data management, user interfaces, visualization, and more.

What is the twist? Simply that, in addition to serving the static components of a standard web application, Tangelo manufactures server-side components from Python scripts included among your HTML, CSS, and JavaScript files. These services can leverage the power and scope of Python to do Pythonic things in your web application.

Here is a silly but illustrative example. Suppose you create this short Python script in the directory being served by Tangelo, now.py:

```python
import datetime

def run():
    return str(datetime.datetime.now())
```

Because of the presence of that `run()` function, Tangelo will respond with a string representing the current time.

We can try it out right now. Step one is to install Tangelo. The easiest way to do so is using the Python package manager Pip:

```
$ pip install tangelo
```

Step two is to create the file `now.py` from the code listing above. Step three is to start Tangelo, letting it know you want to serve files from the current directory:

```
$ tangelo --root .
```

(If you get an error in this step, it may be that your computer is already using port 8080, the default, for another application. You can add `--port 9090` to the command to choose a different port.)

Step four is to open `http://localhost:8080/now` in your favorite web browser. You should see the current time displayed. Refresh the browser, and the displayed time should update. Your first Tangelo application is complete!

Of course, the example is superfluous because JavaScript has the same ability to tell you the time with a simple `console.log(new Date())`. But, there is plenty Python can do that JavaScript cannot. What if you want to:

- Retrieve data from a database using complex query and application-specific post-filtering
- Kick off a long-running compute job on a cluster and periodically check on its progress
- Use specialized visualization tools, such as Bokeh
- Construct a PNG image to display in your webpage using Python imaging libraries
- Use Whizbang, BeesKnees, CatsMeow, or some other new Python module that does something useful, sexy, and fun

In each of these cases, and many others limited only by our collective imagination, Tangelo puts more power in your hands to do what you need to create data sources, analyses, and visualizations.

We have designed Tangelo around the simple idea that Python should be a first-class citizen alongside HTML and JavaScript. The core of Tangelo dedicated to this idea is lean and mean. Thus, it is well testable. To enable advanced functionality on top of this core, Tangelo includes a powerful and flexible plugin system. Several plugins come bundled with Tangelo to do the following:

**Interact with other Kitware software**

- Girder [3], for bringing high performance data storage to Tangelo applications
- GeoJS [4], for creating map-based visualizations and animations
- VTKWeb [5], to create, control, interact with, and terminate VTK-based visualizations in-browser
Interact with Python packages and other software

- Bokeh [6], for applying ready-made visualizations to custom data sources
- Vega [7], to create interactive charts and visualizations
- PyPNG, to construct PNG images of data for use as a thumbnail or a clickable map, etc.

Implement powerful functionality unique to Tangelo

- Convert a basic data service into a streaming service to enable the handling of “big data” sources

Provide help in constructing HTML user interfaces

- A control panel for runtime application options that slides out of the way when not needed
- An SVG legend for use with maps or other SVG diagrams

We mentioned before that Tangelo is aimed toward visualization and analytics applications, and many of our clients, both inside and outside of Kitware, have been using it to good effect that way. But, you can always step outside the box a little bit too. For instance, Tangelo could be used to create system tools, such as a Wifi manager or a print monitor, since Python also affords a way to peek into such system-level services.

WHY TANGELO?

It is a good time to be developing applications for the web. There are many web application frameworks out there, which means lots of choices. It also makes it more likely that what you need exists. So, among the sea of choices, why choose Tangelo?

First, we have a quick disclaimer: If you are planning to build a traditional web application with a database, models to represent application data, and views to display it, then existing systems such as Ruby on Rails will match up to your project’s goals better than Tangelo. In our work, we have been looking beyond, toward scientific simulation and visualization, as well as toward other unusual applications that must pull together data from many unconventional sources, using experimental technology to process and visualize it.

As we mentioned earlier in the article, we believe that Python scripts should be considered first-class citizens alongside HTML, JavaScript, and CSS files, adding their voices and the richness of the Python ecosystem to your web applications. We have designed Tangelo to deliver this idea as simply as possible in order to open up web application programming to everyone, including those unfamiliar with terms such as “routing,” “content-type header,” and “RESTful.”

Django, Flask, Bottle, and CherryPy are all Python-based web frameworks. They tend, however, to relegate Python itself to second-class status, as it is wrapped up in a bit of red tape. Here is an example in a file called bottle-hello.py:

```python
import bottle

app = bottle.Bottle()

@app.route("/hello/<name>")
def greet(name="Earthling"):
    return "Greetings %s!" % (name)

bottle.run(app, host="localhost", port=8080)
```

When run with python bottle-hello.py, this application will serve a bit of dynamic content at /hello/Roni. Notice, however, how the simple code to create the content is suspended within a spider web of scaffolding. In Tangelo, things are much simpler. In your application directory, creating the following file, named hello.py, and then issuing the command tangelo --root. will serve this minimal application, which behaves the same way as the Bottle application but with fewer lines of code and without exposure of any framework API.

```python
def run(name="Earthling"):
    return "Greetings %s!" % (name)
```

Tangelo’s internal server logic handles the scaffolding you see in the Bottle example, and the names of files and directories on disk implicitly determine the routing. All that remains is the heart of the web service, expressed in general, Pythonic terms. When we say “first-class citizen,” we mean the “zero-API” creation of Python web services in individual files that are composed almost entirely of application logic.

In the excitement over Python, we should not forget about the humble static content that serves as the foundation of any web application. Serving static content complicates things a bit further for the Bottle application. It needs to declare new routes, listing files or directories to serve at each one. In Tangelo, all you need to do is include the files on disk in paths that reflect the desired routing — zero-API static content for free.

The other frameworks mentioned above look largely similar to the Bottle example. We showed the Bottle example because the other frameworks are all at least as complicated (some significantly more so).
We designed Tangelo this way for simplicity, as well as for the flexibility that results. For example, thinking of a web application as a bundle of HTML, JavaScript, CSS, and Python files enables reusability. There are ways to reuse the Python callbacks that make up a Bottle application. None of these ways, however, are so simple or flexible as simply handing someone a Python file to drop into their web application. The question moves from “How do I get route X to do action Y?” to “How do I do action Y?” This is the same question with which any Python programmer in any context wants to be engaged. In other words, it frees the developer from the mechanics of web servers and, thereby, abstracts function from form, leaving our focus just on the substance of the problem at hand.

Hopefully this gives you a glimpse of the interesting and useful things that will arise from Tangelo’s wider debut. Please take a look at the Tangelo website (http://www.tangelohub.org/tangelo/) for general information or the documentation (http://tangelo.readthedocs.org/en/latest/) to install, set up, and try Tangelo yourself. Drop us a note at tangelo-users@public.kitware.com if you run into trouble, if you have questions, or if you need help.

ACKNOWLEDGEMENTS
Tangelo development is sponsored by the Air Force Research Laboratory and DARPA XDATA program.

REFERENCES
[9] tangelo-users@public.kitware.com

Dr. Roni Choudhury is a Research and Development Engineer at Kitware. He has directed the design and development of Tangelo from the ground up to bring advanced and experimental information visualization techniques to the web.

Dr. Jeff Baumes is a Technical Leader and Data Scientist at Kitware. His primary responsibility is to create tools that effectively visualize large and complex data, spanning relational, geospatial, temporal, bioinformatics, financial, and textual data.

KITWARE NEWS

DR. HOOGS PRESENTS COMPUTER VISION PAPERS AT IEEE WORKSHOP
Dr. Anthony Hoogs, Kitware’s Senior Director of Computer Vision, presented three papers at the 2014 IEEE Applied Imagery and Pattern Recognition (AIPR) Workshop. The workshop was held in Washington, DC, from October 14 to October 16, 2014.

For “Multi-Target Tracking in Video with Adaptive Integration of Appearance and Motion Models,” Dr. Hoogs discussed a hybrid tracking architecture that utilizes motion detections to robustly initialize multiple tracks. The tracking architecture also uses a blended approach to integrate appearance-based trackers, provides a generalized API for interfacing such trackers, and adaptively uses motion detection or appearance match to update a track. The paper’s authors are Arslan Basharat, Ilker Ersoy, Kannappan Palaniappan, Anthony Hoogs, and Gunasekaran Seetharaman.

For “KWiver: A Open-Source Cross-Platform Video Exploitation Framework,” Dr. Hoogs introduced KWiver, a cross-platform video exploitation framework that Kitware has begun releasing as open source. The paper’s authors include Keith Fieldhouse, Matthew J. Leotta, Arslan Basharat, Russell Blue, David Stoup, Charles Atkins, Linus Sherrill, Benjamin Bockel, Paul Tunison, Jacob Becker, Matthew Dawkins, Matthew Woehlke, Roderic Collins, Matt Turek, and Anthony Hoogs.

Dr. Hoogs also presented “Towards visual analysis of unconstrained images in social forums: Studies on concept detection and personalized economy of images in social networks.” The paper’s authors are Sangmin Oh, Eric Smith, Yiliang Xu, and Anthony Hoogs.

KITWARE WORKS WITH NVIDIA TO DRIVE HPC VISUALIZATION
Kitware announced that it is working with NVIDIA to develop several powerful enhancements for the field of high-performance computing (HPC) visualization.

Among these is a new Visualization Toolkit (VTK) rendering backend, which targets graphics processing units (GPUs) and takes advantage of the flexible programmable pipelines available in modern systems. This development has already resulted in significant improvements in rendering performance, especially with large geometries (20 million+ triangles) that can now be rendered over 100 times faster.

“Through our collaboration with NVIDIA, we will enhance the open-source VTK and ParaView to better serve the HPC...”
visualization community,” said Dr. Berk Geveci, Kitware’s Senior Director of Scientific Computing.

In addition, as part of the Department of Energy (DOE) Advanced Scientific Computing Research (ASCR) project “XVis: Visualization for the Extreme-Scale Scientific-Computation Ecosystem,” Kitware, in collaboration with government labs and academia, is developing a unified visualization library, named VTK-m, as a single point to collaborate, contribute, and leverage massively threaded visualization algorithms. Kitware will be demonstrating VTK-m in NVIDIA GPU-accelerated applications in in situ mode within ParaView Catalyst.

NVIDIA will add future EGL support to its drivers, eliminating the need to install an X server for graphics output in ParaView. The companies are also exploring bringing OptiX parallel ray tracing and volume visualization capabilities into ParaView for better insight into simulations.

This collaboration will enable VTK and ParaView to fully benefit from the huge computational and graphics performance NVIDIA Tesla and Quadro architectures provide.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012386.

Research reported in this publication was supported by the National Institute Of Biomedical Imaging And Bioengineering of the National Institutes of Health under Award Number R01EB014955. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

SC14 SHOWCASE HIGHLIGHTS PARAVIEW AND VTK, AMONG OTHER TECHNOLOGY

Kitware exhibited recent work in HPC and visualization at Supercomputing 2014 (SC14) in New Orleans, LA. SC14 is one of the premier conferences in the scientific computing field.

As part of its ParaView Showcase, Kitware hosted presentations from renowned collaborators at its booth throughout the conference, where they discussed how they are leveraging ParaView, VTK, and other technology in their work. Featured presenters included Alejandro Ribes (Électricité de France), Sean Ziegler (Engility Corporation), James Ahrens (Los Alamos National Laboratory), Kenneth Moreland (Sandia National Laboratories), Paul Navratil (Texas Advanced Computing Center), Joseph Insley (Argonne National Laboratory), Jim Jeffers (Intel), and Alan Scott (Sandia National Laboratories).

In conjunction with SC14, the Second Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2) was held on November 16, 2014. At the workshop, Marcus D. Hanwell provided an overview of a collaborative paper, “Sustainable Software Ecosystems: Software Engineers, Domain Scientists, and Engineers Collaborating for Science.” This paper focuses on the significance of collaborations when developing advanced scientific software, where sustainability, reproducibility, and extensibility are important.

Kitware also presented two papers at PyHPC 2014: “Python Enabled ParaViewWeb for HPC Analysis and Visualization” and “Scientific data analysis and visualization at scale in VTK/ParaView with NumPy.”

Furthermore, members of Kitware participated in invited talks. On November 16, 2014, Sébastien Jourdain presented a talk on ParaView at the HPC Developer Showcase. Then, Kitware was also involved in this year’s event in a number of tutorials. On November 16, 2014, a team presented the tutorial “Large Scale Visualization with ParaView.” This tutorial showcased the architecture of ParaView and the fundamentals of parallel visualization.

On the following day, Kitware and its collaborators presented the tutorial "In Situ Data Analysis and Visualization with ParaView Catalyst." This tutorial examined the architecture of ParaView Catalyst and the fundamentals of in situ data analysis and visualization. In addition, a group of presenters discussed "An Image-Based Approach to Extreme Scale In Situ Visualization and Analysis" in the Big Data Analysis session.
on November 19, 2014, Robert Maynard explored “VTK-M: Uniting GPU Acceleration Successes” at the NVIDIA booth.

Also on November 19, members of Kitware, as well as several of its collaborators, highlighted “In Situ MPAS-Ocean Image-based Visualization” at the Visualization and Data Analytics Showcase. This video discusses a novel framework for highly interactive, image-based in situ visualization and analysis that promotes exploration.

KITWARE FEATURES APPS DEVELOPED WITH QT FRAMEWORK AT DEVELOPER EVENT

Qt Developer Days 2014 was held in Burlingame, CA, and featured presentations from experts in the Qt community. Throughout the three-day conference, Sankhesh Jhaveri showcased a variety of applications developed using the Qt framework, including ParaView, 3D Slicer, Slicer Extensions, VesselView, Bender, CMake, Avogadro, and VIVIA for Wide Area Motion Imagery (WAMI). Kitware has been one of the leading development teams for many of these cross-platform, open-source projects.

Bender is a free, open-source application for repositioning voxelized anatomical models.

Avogadro is a chemical editor and visualization application.

DR. TUREK DISCUSSES NOVEL FRAMEWORK AT MSS NATIONAL SYMPOSIUM

Dr. Matt Turek, Kitware’s Assistant Director of Computer Vision, presented two papers at the MSS National Symposium on Sensor and Data Fusion, which was held from October 28 to October 31, 2014, in Springfield, VA.

For "Interactive, Content-Based Exploration of Large Visual Archives through Feature Set Fusion," Dr. Turek discussed a novel framework Kitware is developing for visual data search and exploration for social multimedia. The framework combines computer vision, data fusion, and graph-based interactive visualization applied to content commonly found on the web, such as Youtube videos. The paper’s authors are Sangmin Oh, Jeff Baumes, and Anthony Hoogs.

For "Probabilistic Sub-Graph Matching for Video and Text Fusion," Dr. Turek detailed Kitware’s work in associating textual descriptions of entities and activities with automatically extracted visual entities (e.g., tracks) and activities (e.g., “person walking”), while also inferring the presence of missing details. The paper’s authors are Eran Swears, Arslan Basharat, Anthony Hoogs, and Erik Blasch.

DR. ENQUOBAHRIE SHEDS LIGHT ON NEED FOR COMMUNITY INFRASTRUCTURE IN MHEALTH

On October 29, 2014, Dr. Andinet Enquobahrie participated in the National NSF workshop on Computing Challenges in Future Mobile Health (mHealth) Systems, which was held at the National Institutes of Health in Bethesda, Maryland. During the presentation, Dr. Enquobahrie shared Kitware’s experience in developing software and community building tools for the medical imaging and EHR communities.

Dr. Enquobahrie discussed with the workshop’s participants that a publicly accessible mHealth database and a robust community infrastructure will foster collaborations between mobile health researchers and computational scientists. In order to create a robust infrastructure that is sustainable and effective, however, key technical considerations must be taken into account during the design process. Such considerations include user-friendly and flexible access, the capability to handle heterogeneous data, a unified API, and standardized baseline algorithms.

To build a community around the database, it is important to find mutually beneficial incentives for collaboration between public health researchers, computer scientists, computational scientists, clinicians, and healthcare administrators. It is also critical to build trust among the community’s members and to develop a common purpose.

Other key considerations discussed during the workshop were the benefits of a strong collaborative multidisciplinary research community, the major challenges in building a strong community, the best practices to initiate and nurture a vibrant and active community, and the role of open science (open data, open model, and open-source software).
The overall takeaway from Dr. Enquobahrie’s presentation is that a well-designed publicly accessible database and collaboration infrastructure will encourage robust collaboration among researchers and lead to successful data-driven healthcare solutions.

**KITWARE AND TACC COLLABORATE TO BECOME INTEL® PARALLEL COMPUTING CENTER**

Kitware has recently become an Intel® Parallel Computing Center in collaboration with Texas Advanced Computing Center (TACC). This team will focus its efforts on supporting the Intel® Xeon Phi™ Coprocessor in ParaView and in the Visualization Toolkit (VTK).

Today, HPC is widely recognized as a strategic capability of fundamental importance to the economic strength and national security of the United States and other developed countries, applicable to areas ranging from engineering simulation to medical computing to climate modeling. However, it can be difficult to deploy HPC at scale due to the complexity of the programming model and the lack of reference implementations with which to educate and accelerate the creation of new software solutions.

To address these challenges, the team of collaborators will work to add support for the Intel® Xeon Phi™ Coprocessor in VTK/ParaView. Adding this capability to these well-established and widely adopted open-source projects will not only provide easily deployable scalable solutions, but will also serve as a working example for software developers to study and extend.

“The opportunity to become an Intel® Parallel Computing Center will foster a collaboration between Kitware, TACC, and Intel that will enhance ParaView to run more efficiently on the Intel® Xeon Phi™ Coprocessor platform,” Dr. Berk Geveci, Kitware’s Principal Investigator for the project and Senior Director of Scientific Computing, said. “We are honored to be part of such a critical effort in HPC.”

For the project, Kitware and TACC will optimize VTK/ParaView to support Phi’s wide vector processing capability and to take advantage of the large number of computing cores. The team will develop support for both coarse and fine-grained parallelism and will release open-source software to demonstrate the use of these capabilities in a variety of scientific computing applications. By leveraging the multi-core and vector capabilities of the Intel® Xeon Phi™ coprocessors, the team will demonstrate a significant increase in the performance of VTK/ParaView.

Kitware’s scientific computing team will collaborate with TACC under the leadership of its Principal Investigator, Dr. Paul A. Navrátil. Dr. Navrátil is a Research Associate and the Manager of TACC’s Scalable Visualization Technologies group. He is an expert in high-performance visualization technologies and advanced rendering techniques, including algorithms for large-scale distributed-memory ray tracing.

**KITWARE FUELS SCIENTIFIC DISCOVERY WITH ASCR AWARDS**

Kitware recently announced its collaboration on three Advanced Scientific Computing Research (ASCR) awards from the Department of Energy (DOE).

The awards are part of the ASCR Computer Science program, which supports research that enables computing at extreme scales and the understanding of extreme-scale data from both simulations and experiments. Recently, the program announced nine new awards under Scientific Data Management, Analysis & Visualization at Extreme Scale 2.

“We are honored to be selected for three of the ASCR awards,” Dr. Berk Geveci, Kitware’s Principal Investigator on two of the projects and the company’s Senior Director of Scientific Computing, said. “The Computer Science program is important because it addresses key areas in the field of scientific computing, including high-performance computing (HPC) and extreme-scale simulations.”

For the first project, “XVis: Visualization for the Extreme-Scale Scientific-Computation Ecosystem,” Kitware will collaborate with Sandia National Laboratories, under the leadership of Principal Investigator Ken Moreland; Oak Ridge National Laboratory; Los Alamos National Laboratory; University of California Davis; and University of Oregon.

The team will build a unified visualization library, named VTK-m, as a single point to collaborate, contribute, and leverage massively threaded visualization algorithms. The motivation behind the project is the shift in HPC toward the increasing use of accelerators and other processor technologies that use greater concurrent threading to overcome physical limitations of power and latency.

Building on top of this foundation, the XVis project will research techniques for flyweight in situ components, advanced data models, new domain challenges, and post-hoc interaction techniques.

Secondly, on the “Scalable Analysis Methods and In Situ Infrastructure for Extreme Scale Knowledge Discovery” project, Kitware will collaborate with Lawrence Berkeley National Laboratory, under the leadership of PI Wes Bethel; Argonne National Laboratory; Georgia Institute of Technology; and JMSI, Inc.

The project will address a set of research challenges for enabling scientific knowledge discovery within the context of in situ processing at extreme-scale concurrency. The work is motivated by a widening gap between Floating-point Operations Per Second and input/output (I/O) capacity, which will make full-resolution, I/O-intensive post hoc analysis prohibitively expensive, if not impossible. The team will focus on developing new algorithms for analysis and visualization that are suitable for use in an in situ context aimed specifically at enabling scientific knowledge discovery in several exemplar application areas of importance to the DOE.
Lastly, Kitware will collaborate with Los Alamos National Laboratory, under the leadership of Principal Investigator Pat McCormick; Sandia National Laboratories; University of Utah; and Stanford University on the project “A Unified Data-Driven Approach for Programming In Situ Analysis and Visualization.”

The overarching goal of the project is the study of a unified data-driven approach for programming applications, as well as in situ analysis and visualization. In particular, the team will focus on the interplay between data-centric programming model requirements at extreme scale and the overall impact of those requirements on the design, capabilities, flexibility, and implementation details for both applications and the supporting in situ infrastructure.

“We are looking forward to collaborating with world-renowned organizations on projects that will significantly advance scientific discovery,” Geveci said.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Numbers DE-SC0012387, DE-SC0012388, and DE-SC0012386.

VIRTUAL SURGERY SYSTEM AIMS TO HELP SURGEONS TREAT CRANIOSYNOSTOSIS

Kitware is developing software for treatment planning and evaluation of craniosynostosis through robust, quantitative, and reproducible methods that assess cranial shape.

Craniosynostosis is the premature fusion of cranial sutures. It occurs in approximately one in 2,000 live births and results in cranial malformation that can lead to elevated intra-cranial pressure and brain growth impairment. It can also lead to developmental deficiency.

While the most common treatment option for craniosynostosis is surgery, today’s surgical treatment planning is mostly qualitative, subjective, and irreproducible. In addition, although virtual planning has been successfully introduced in niche areas of craniofacial surgery, such as corrective jaw surgery applications, there exists a need for clinical tools that provide accurate and reproducible evaluation of cranial morphology to guide cranial vault remodeling.

“To cover the gap in current clinical practice, we will develop personalized technology for preoperative planning for infants with craniosynostosis,” Dr. Andinet Enquobahrie, Kitware’s Principal Investigator for this project and Assistant Director of Medical Computing, said. “The technology will help decrease operative time and blood loss to reduce perioperative morbidity and facilitate an optimized and more durable long-term outcome.”

The goal of the Phase I Small Business Technology Transfer (STTR) project funded by the National Institutes of Health is to design, develop, and validate a virtual surgery system for optimal treatment planning for cranial remodeling. To accomplish this goal, the team will create a normative multi-atlas database that captures a wide breadth of normal variations on cranial shape and patient ages. The team will also develop image processing, statistical shape analysis, and biomedical modeling algorithms to identify desirable postoperative skull shapes.

The project is a collaboration between Kitware and Children’s National Health System in Washington, DC. Children’s National is an internationally recognized pediatric hospital, with over 140 years of experience, and a center of excellence in the treatment of birth defects and the clinical management of craniosynostosis. Children’s National is also home to the Sheikh Zayed Institute for Pediatric Surgical Innovation, which is devoted to advancing the frontiers of pediatric surgery.

“The impact of this technology is reduced perioperative morbidity and lower treatment costs,” Dr. Marius Linguraru, the Principal Investigator for the project and a Principal Investigator and Director of the Quantitative Imaging Group of the Sheikh Zayed Institute for Pediatric Surgical Innovation, said. “The technology will also enable the precise, quantitative comparison of measurements before and after cranial vault reconstruction to determine the efficacy and durability of specific reconstructive techniques.”

The project will build upon Kitware’s extensive experience in the field of surgical guidance. As the primary developer of Image Guided Surgery Toolkit (IGSTK), Insight Registration and Segmentation Toolkit (ITK), and 3D Slicer, Kitware has demonstrated leadership in the creation, integration, and support of state-of-the-art medical data visualization and analysis technology.

Research reported in this publication was supported by the Eunice Kennedy Shriver National Institute Of Child Health & Human Development of the National Institutes of Health under Award Number R41HD081712. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

KITWARE PRINTS NEW BOOKS ON DEMAND

Kitware is pleased to announce that the newest versions of the ITK, ParaView, and CMake books are now available through Amazon CreateSpace. In an effort to best contribute to the open-source communities in which Kitware participates, the books are now being published as print-on-demand offerings, enabling Kitware to provide an updated book with each software release.

The ParaView Guide is updated for the 4.3 release, and Mastering CMake is updated for the 3.1 release. Additionally, The ITK Software Guide has been completely rewritten and split into two books: 1) Introduction and Development Guidelines and 2) Design and Functionality.

More information, including links for ordering the books, is available on the Kitware website. Check out the updated books now!
NEW EMPLOYEES

Sujin Philip

Sujin joined Kitware as a Research and Development Engineer on the Scientific Computing team. Sujin received his B.E. from the University of Pune and his M.S. in Computing – Graphics and Visualization from the University of Utah. Prior to joining Kitware, Sujin was a Research Assistant at the Scientific Computing and Imaging Institute. There, he developed a scalable, parallel, out-of-core solver for gradient domain processing of large, gigapixel-sized images using multithreading, GPGPU, and MPI. Sujin also developed a scalable, parallel, out-of-core technique for computing seams for gigapixel-sized panoramas using multithreading and MPI.

Javier Ortega

Javier joined Kitware as an Assistant Office Administrator. Javier attends Hudson Valley Community College, where he is studying respiratory care. Prior to joining Kitware, Javier was a Customer Service Representative at Pioneer Bank and a Hyperbaric Technician at Mobile Hyperbaric Centers. Javier was also a Teacher Associate at the Center for Discovery in Harris, NY, where he assisted in teaching children with developmental disabilities skills for achieving higher levels of independence.

Dženan Zukic

Dženan joined Kitware as a Research and Development Engineer on the Medical Computing team. Dženan has defended his thesis and is currently finishing up his Ph.D. dissertation at the University of Siegen. He is studying vertebral body segmentation and diagnosis in magnetic resonance images. Prior to joining Kitware, Dženan was a System Engineer at BSTelecom, where he designed and implemented a web interface for a database-backed inter-telecom billing system. Dženan also has experience as a developer at aNET. As such, he designed a database, a GUI, and a web interface for the technical examination and registration of vehicles.

UPCOMING EVENTS

Naval Future Force Science And Technology (S&T) Expo
February 4 to February 5, 2015
Washington, DC

The expo is sponsored by the Office of Naval Research (ONR). Topics to be addressed at the event include objectives of the revised Naval S&T Strategy and new research opportunities within the Navy and Marine Corps community. The status of key programs will also be highlighted.

Presentation on Open Source at the College of Charleston
February 6, 2015
Charleston, SC

Stephen Aylward and Jean-Christophe Fillion-Robin will present “Accelerating Medical Visualization Research via Open-Source Platforms and Practices.” This talk will introduce the tools that enable the creation of effective open-source platforms, discuss the costs and benefits of creating and using open-source platforms, and identify the trends of open-source visualization in medicine.

Specific examples will be given throughout the talk, drawn from ongoing research and development projects at Kitware, including recent enhancements to 3D Slicer and research into USB-based ultrasound probes and automated ultrasound analysis software for in-field brain and abdominal trauma assessment.

The talk will conclude with a live demonstration that highlights innovations in 3D Slicer for interactive disease visualization and treatment planning.

Rensselaer Polytechnic Institute Spring 2015 Career Fair
February 11, 2015
Troy, NY

Members of Kitware will be recruiting candidates for job and internship opportunities at the career fair. For a list of available positions, please visit http://jobs.kitware.com/opportunities.html.

SPIE. Medical Imaging
February 21 to February 26, 2015
Orlando, FL

The conference will consist of sessions, keynote addresses, panels, workshops, poster presentations, and interactive demos. Topics of discussion will include image processing, computer-aided diagnosis, image-guided procedures, ultrasonic imaging and tomography, and digital pathology.

For more information, please visit the conference’s website at http://spie.org/medical-imaging.xml.

SIAM Conference on Computational Science and Engineering
March 14 to March 18, 2015
Salt Lake City, UT

Kitware has been actively involved in planning this event, as Patrick O’Leary is on the Organizing Committee. Kitware will also have a strong presence at the event, beginning with Will Schroeder’s keynote address. In addition, members of Kitware will present several minisymposiums, including “Software Quality with the Open Source Tools CMake, CDash, CTest” and “Computational Model Builder and ParaView Catalyst: Empowering HPC Workflows.”

Kitware will also present a minisymposium titled “Software Process for a CASL Sustainable Simulation Software Solution” and a minitutorial titled “Python Visual Analytics for Big Data.” Furthermore, Kitware will attend the Career Fair, which will take place on March 14.

For more information, please visit the conference’s website at http://www.siam.org/meetings/cse15/index.php.
GPU Technology Conference
March 17 to March 20, 2015
San Jose, CA

The conference will host scientists, programmers, researchers, and other professionals. It will feature keynote addresses, exhibits, receptions, sessions, hangouts, and labs.

At the conference, Robert Maynard and Marcus D. Hanwell will present the paper "Visualization Toolkit: Faster, Better, Open! Scientific Rendering and Compute."

For more information, please visit the conference's website at http://www.gputechconf.com/.

IEEE Virtual Reality 2015
March 23 to March 27, 2015
Arles, Camargue – Provence, France

Patrick O'Leary is an organizer of the Third Workshop on Immersive Volumetric Interaction (WIVI 2015). According to the conference's website, the goal of the workshop is to unite researchers, developers, and users from several communities in Virtual Environments, 3D User Interfaces, and Immersive Visualizations. At the workshop, attendees will talk about issues and present ideas in regards to the challenge of interacting with volumetric data in immersive visualizations. Multiple objects defined as a volume in virtual environments will also be discussed.

For more information, please visit http://ieeevr.org/2015/.

EMPLOYMENT OPPORTUNITIES

Kitware is seeking talented, motivated, and creative individuals to fill open positions. Kitware has an immediate need for software developers and researchers who will work on cutting-edge research and join us in our mission to develop and deliver state-of-the-art software products and services using advanced software quality methods and technologies.

The impact of your research at Kitware will extend far beyond our organization, as our open-source business model will allow you to become part of the worldwide communities that surround our projects. Kitware employees enjoy a collaborative work environment that empowers them to pursue new opportunities and to challenge the status quo through novel ideas.

In addition to providing an excellent workplace, Kitware offers comprehensive benefits including: flexible hours; a computer hardware budget; health, vision, dental, and life insurance; short- and long-term disability; visa processing; a generous compensation plan; and free drinks and snacks.

For additional information, please visit our employment website at jobs.kitware.com. Interested applicants are encouraged to submit their resumes and cover letters through our online portal.

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 interns assist in developing foundational research and leading-edge technology across five business areas: HPC & visualization, computer vision, medical computing, data and analytics, and quality software process. We not only offer our interns a challenging work environment, but we also provide them with the opportunity to attend advanced software training.

Contributors: Lisa Avila; Jeff Baumes; J. Dexter Bird, III; Roni Choudhury; Casey Goodlett; and Adamos Kyriakou

Graphic Design: Steve Jordan
Editor: Sandy McKenzie

This work is licensed under an Attribution 4.0 International (CC BY 4.0) License.

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