Kitware SOURCE

SOFTWARE DEVELOPER'S QUARTERLY

Issue 22 • July 2012

TABLE OF CONTENTS

EDITOR'S NOTE

The Kitware Source contains articles related to the development of Kitware products in addition to software updates on recent releases, Kitware news, and other content relevant to the open-source community.

In this issue, Aashish Chaudhary discusses Kitware's mobile develoment efforts with the VES library; Philippe Pebay highlights new work in annotation capabilities in VTK; Yuanxin Leo Liu presents new work on path tracing in VTK; Özgür Güler and Ziv Yaniv showcase their image-guided interventions tutorial that leverages IGSTK; and Tara Lindsley explains how ITK was used in creating a student-accessible brain atlas.

The Source is part of a suite of products and services offered to assist developers in getting the most out of Kitware's open-source tools. Project websites include links to free resources such as mailing lists, documentation, tutorials, 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.

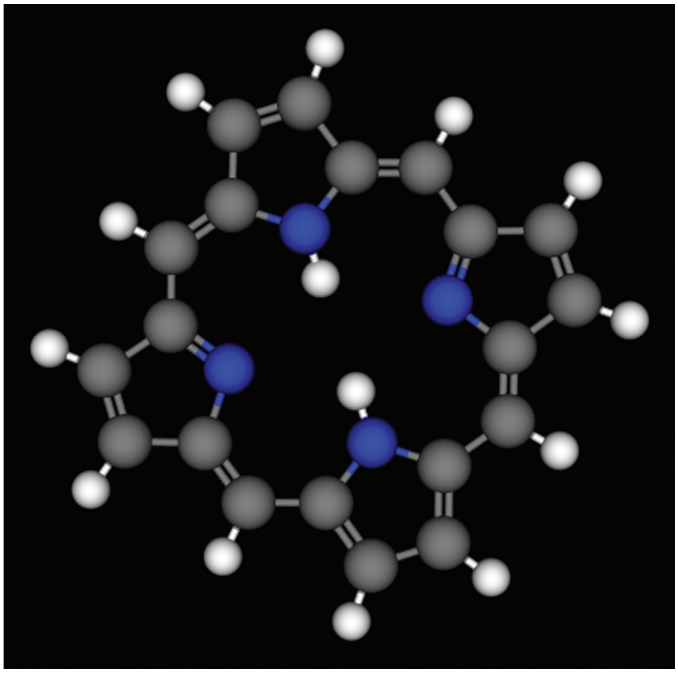
RECENT RELEASES

VTK 5.10

VTK 5.10 was released in May, with new and updated classes, utility improvements, and other enhancements.

A new set of image rendering classes was incorporated for use in the next generation of VTK-based image viewer applications. Like VTK's volume rendering classes, the new image rendering classes consist of separate actor, mapper, and property classes for maximum flexibility. Many of the VTK reader classes were updated, including the LSDyna reader, which resulted in read times for large (100 gigabyte) parallel data sets dropping from multiple hours to several minutes. Additionally, there is a new crop of NetCDF readers, and VTK now has true support for netcdf4 readers

The Google Summer of Code 2011 work by David Lonie and Tharindu de Silva was incorporated into VTK. David developed chemical structure visualization code, which adds accelerated rendering of 3D chemical geometry using standard chemical representations. Tharindu worked on the 2D chart and plot features in VTK, improving chart interaction and adding support for keyboard modifiers to mouse and key events.



In addition to new and updated classes, support for new compilers, such as Visual Studio 2011 and Clang, was added, along with a multitude of code quality improvements. The code quality improvements came from a persistent effort

on the part of the community to plug memory leaks and address compilation warnings that the CDash regression test servers exposed. In a related development, there is now an extension to VTK's leak detection that allows fine-grained profiling of the construction and destruction of VTK objects.

There were several important external development efforts as part of the 5.10 release. To provide the ability to perform arbitrary boolean operations on polygonal meshes, a team from UNC made several meshing improvements, and added a new quad rotational extrusion filter that generalizes beyond triangle meshes. As part of DOE/ASCR and Sandia National Lab's efforts to extend the state of the art in Mathematical Analysis of Petascale Data, the statistics algorithms were revamped and extended.

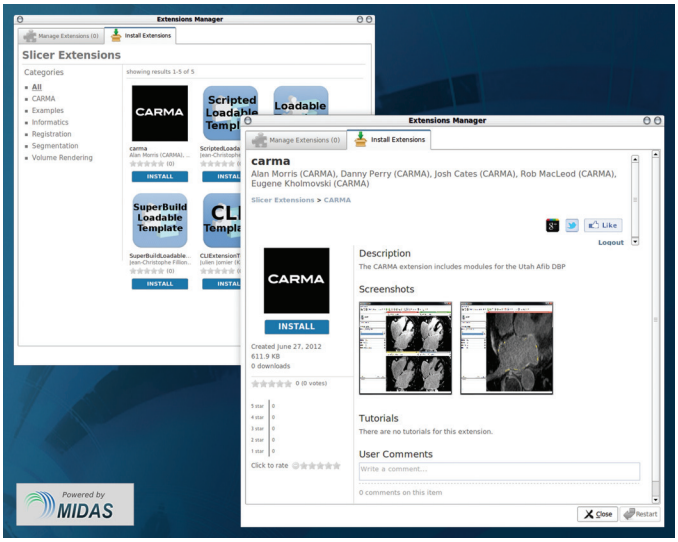
Other improvements include framework and widget updates. The generic framework to handle arbitrary high order cells and custom interpolation functions was enhanced in this release to seamlessly work with the modern VTK pipeline paradigm, and particularly within ParaView. There was also a new hierarchy of versatile multi-state button widgets added to VTK, along with others including a cropping plane, reslice cursor, and broken line widgets.

Also of note, visualization within immersive environments was improved. The ability to perform "Off-Axis" rendering was added to VTK, and can be used either with or without stereoscopic visualization.

SLICER 4.1 AND 4.1.1

Kitware, in collaboration with the National Association for Medical Image Computing (NA-MIC) and the Slicer community announced the releases of Slicer 4.1 and 4.1.1 this spring. These releases brings a new level of stability to Slicer and new features to enhance its usability and functionality.

Notable changes include the addition of the Extension Manager, which enables Slicer extensions to be downloaded and installed from within Slicer. To support extension contributions from the community, an "extension catalog" server (similar to an app store) is running to host uploaded contributions from Slicer developers. Built on Kitware's Midas technology, the "extension catalog" makes it easier for the community to search, find, rate, comment, and download relevant extensions.



In the Slicer 4.1.1 release, these were enhanced and further stabilized. The extensions manager and associated build system provide users and developers a way to quickly extend the functionality and the look-and-feel of Slicer for specific biomedical and clinical research applications. With the extension manager, extensions can be shared with the community in the "app-store" style system that integrates tightly with the Slicer experience. To complement these improvements, extensions-specific documentation was written.

Charting support has been added with a new chart view in the 4.1 release, which is used by the MultiVolumeExplorer module. This new module introduces multi-volume (e.g. time series) support in Slicer. The Cache Settings panel has been ported from Slicer 3, to provide users with controls to display or clear the available cache space used when downloading sample data, and to store temporary filter outputs. Further support for importing VTK unstructured grids is also a new addition.

Several other modules have been updated or added, including the OpenIGTLinks, Welcome, and DICOM modules. The CompareViews and View Controller GUI have been revised and improved. The Modules settings panel has also been enhanced, enabling users to set Prefer Executable CLI loading option to decrease memory consumption by modules and select which module(s) to skip at startup, as well as to customize their Favorite modules toolbar. Many of the icons for the Core Modules have also been updated. Slicer has also been updated to use CMake version 2.8.7.

Lastly, with the 4.1.1 release, the Slicer team also reorganized the overall user and developer documentation, making it easier to get started with Slicer and start developing your own modules and extensions.

MIDAS 3.2.6

The Midas 3.2.6 release features security and stability improvements and other upgrades to enhance the Midas user experience. The community descriptions functionality was enhanced and the upload workflow was streamlined. These two features are vital to Midas, and the updates provide a much better user experience. To further increase the user-friendliness of the system, the team added more batch move, merge, copy, and edit operations to folder views, as well as making the folder view sortable.

For those interested in tailoring their Midas installations themselves, the Midas team added a plugin for landing pages and the ability to render Midas without the default layout, and new web-api functionality to download by hash.

CMAKE 2.8.8

CMake 2.8.8 was released in April, with several significant developments and enhancements including greatly improved CPack documentation and a new Ninja generator on Linux.

This release features an easy way to generate a package config file with CMakePackageConfigHelpers and build MSVC-compatible dlls with MinGW gfortran in CMakeAddFortranSubdirectory. There is new support for Xcode 4.3, which also has a new default location and directory structure.

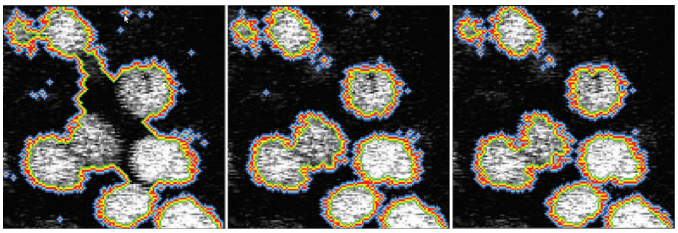
There are several improvements to the find_package command, such as a new module mode and clearer error messages, making CMake more user-friendly. Additionally, the Find modules are much cleaner, with the effort focused on the consistent use of FIND_PACKAGE_HANDLE_STANDARD_ARGS and version number variables.

The CMake team updated libarchive to the new upstream version, and added a new OBJECT_LIBRARY target type to provide CMake-flavored convenience libraries.

ITK 4.1.0 AND 4.2.0

In March the ITK development team announced the release of ITK 4.1.0, followed by the release of ITK 4.2.0 in July. Note that these releases deviate from the "even only" numbering convention of the past due to the transition to Git.

Notable improvements in the 4.1.0 release include better WrapITK support, fixes in the VTK Bridge to adapt to VTK 5, a new deconvolution filtering module, and a new TimeVaryingBSplineVelocityFieldTransform. In addition, there are enhancements to the Registrationv4 and LevelSetsv4 frameworks, a new video filtering module, and updates in InsightApplications. In ITK 4.2.0, compiler support was improved, GPU modules for Finite Difference, Smoothing, Thresholding, and Registration were added, and a new infrastructure to support Remote Modules was added.



Segmentation of cells from a Fluorescent Microscopy image using the new Level Set framework available in ITK 4.1.

IGSTK 5.0

The Image-Guided Surgery Toolkit (IGSTK) team released IGSTK 5.0 in early June. This release, driven by the team at Children's National Medical Center, features support for additional tracking devices, a new example application, and a new version control workflow.

There is additional support for five new trackers: Certus optical Tracker, Polaris Classic, CamBar B2, easyTrack 500, and monocular tracking using ArUco. With support for the Certus optical Tracker, IGSTK now works with all optical and electromagnetic tracking systems from NDI. By adding support for Polaris Classic, an older model of NDI Polaris Tracker, IGSTK supports both the original and current communication APIs for the Polaris tracking systems.

The arsenal of tools compatible with IGSTK is extended through support for the CamBar B2 and easyTrack optical trackers. The CamBar B2 optical tracker is a compact passive optical tracking system available from AXIOS 3D Services, whereas the easyTrack 500 is an active optical tracking system available from Atracsys LLC. Lastly, support for monocular tracking of planar markers using the ArUco toolkit, available from the AVA research group at the University of Córdoba, transforms any computer with a webcam into a tracking system.

Leveraging these trackers, there is a new example application that can be used to log pose information from the NDI Polaris Classic, CamBar B2 and easyTrack500.

Lastly, IGSTK moved to a Git workflow for version control. This new workflow enables the team to track every change that goes into IGSTK and maintain a stable repository.

MOBILE APPLICATION DEVELOPMENT : VES LIBRARY INSIGHTS

According to a report published by the International Telecommunication Union, there were six billion mobile subscriptions at the end of 2011 (see Figure 1). Also, in 2011, sales of smartphone surged to 472 million units. A smartphone is a mobile phone built on a mobile computing platform, with more advanced computing ability and connectivity than a feature phone. The new generation of smartphones offer various multimedia capabilities including interactive games and applications. This is become possible because of the recent improvements in mobile hardware and software. These significant improvements also enabled scientific community to use mobile devices for visualization purposes. For example, Zhou et al [3] has investigated how to achieve volume visualization on mobile devices; Tesch et al [4] developed a system that uses mobile devices for interacting and exchanging data between heterogeneous scientific visualization systems; and Roudaut et al [5] presented their work on interaction techniques for mobile devices. Though technically impressive, none of these offer a general, extensible, and open-source framework for enabling developers to build high performance visualization applications on mobile devices.

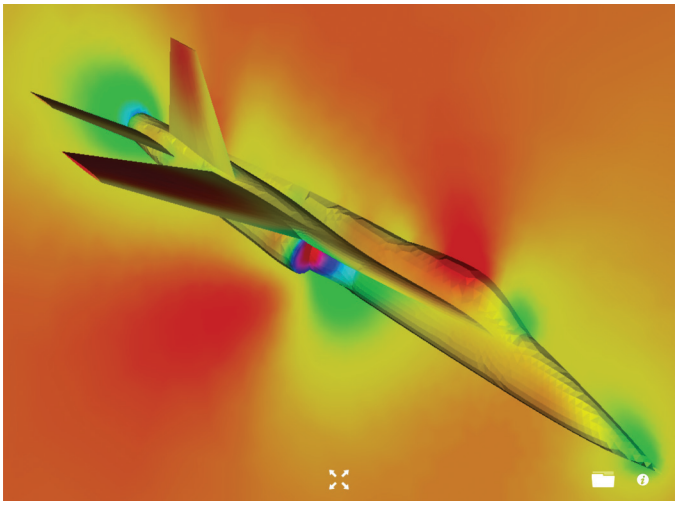


Figure 1: Unstructured Mesh for YF-17 Visualized using VTK and VES Library

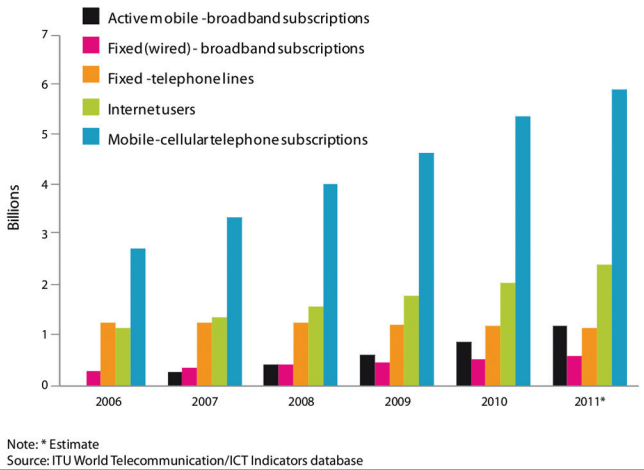


Figure 2: Growth of Mobile Subscriptions

The VES framework [11] has been developed to address these limitations. It puts powerful tools at the fingertips of developers to enable the building of high performance visualization applications on mobile devices. VES is an open-source framework for accelerating mobile visualizations on current generation mobile hardware. VES consists of two core libraries: VES and Kiwi. The VES library provides rendering capabilities and infrastructure for scene management by utilizing OpenGL ES 2.0 [8] whereas the Kiwi library provides an application framework and is built on top of VES and VTK [6]. In this article, we will discuss the architecture and various components of the VES library along with code snippets to explain the API offered by the VES library.

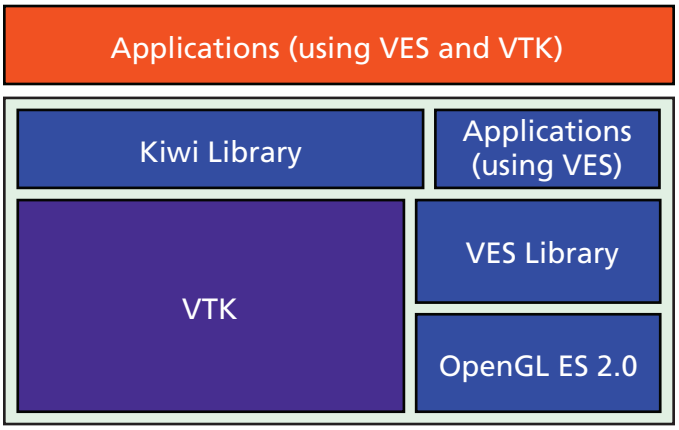


Figure 3: Architecture of VES

ARCHITECTURE & IMPLEMENTATION

As previously mentioned, the VES library provides scene management and rendering functionalities to the applications developed using the VES framework. It uses OpenGL ES 2.0 as the underlying rendering API, which is a subset of OpenGL for desktop and replaces the fixed function transformations and fragment pipeline of OpenGL 1.x. OpenGL does not provide a high-level, object-oriented API for the application development, which the VES library provides to reduce time and effort to build visualization applications.

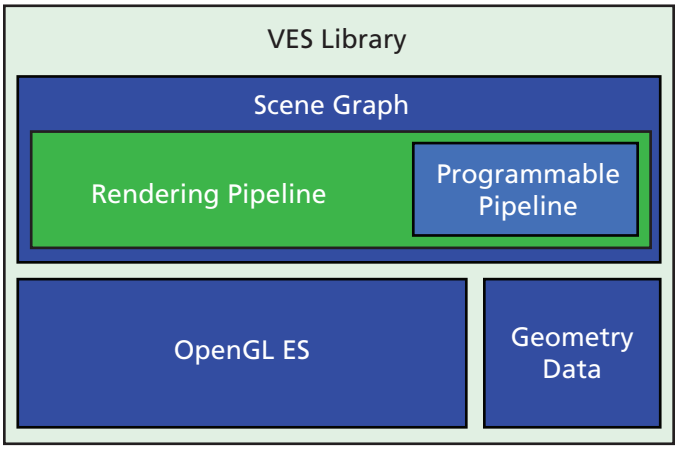


Figure 4: Components of VES Library

As shown in Figure 4, the VES library is composed of multiple components, each of which offers a unique feature within the library. The next few sections will cover some of these components and the underlying technology in detail.

Scene Graph

The VES library uses scene graph data structures to manage scenes efficiently. A scene graph is a data structure that provides spatial and logical relationships between various entities of a scene. A scene graph can be implemented in many ways and some of the open source implementations of a scene graph are inspired by the design of OpenGL Performer, one of the well-known scene graph libraries from SGI [7]. The VES library is built using the same core principles and additionally provides a consistent, easy-to-use API to allow applications to take advantage of programmable pipeline functionality of OpenGL ES 2.0. Figure 5 shows the inheritance hierarchy in the VES scene graph.

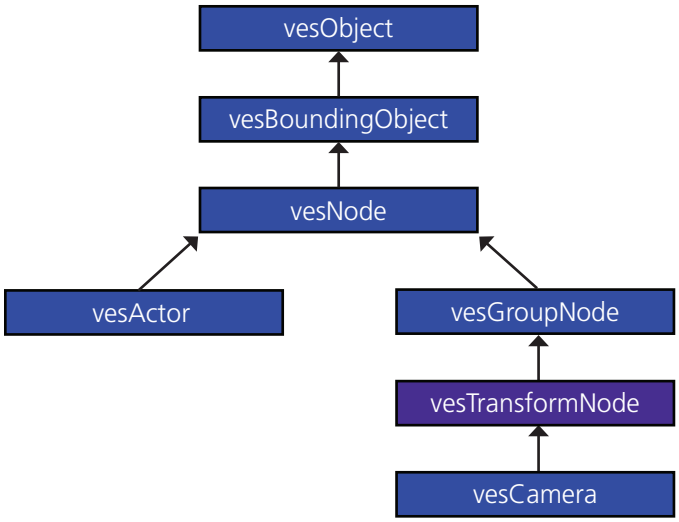


Figure 5: Inheritance Hierarchy in VES Scene Graph

Rendering Pipeline

Rendering in the VES library is a three-pass algorithm as shown in Figure 6. VES renders the scene by traversing it in a depth-first manner. At the API level, the VES library separates the geometry data from the appearance of the geometry. The appearance of the geometry is captured by the notion of a material in the VES library.

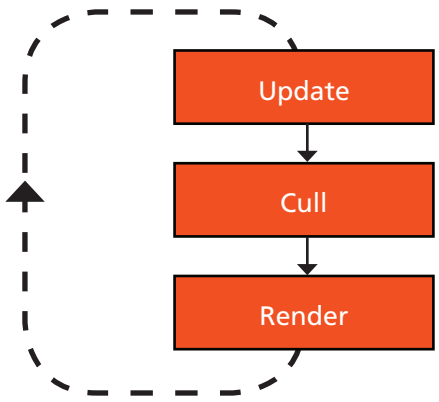


Figure 6: Rendering Model in VES

This separation facilitates reuse of geometry and/or material if required by an application. Since state change in OpenGL is an expensive operation, VES library constructs and renders a scene tree in which geometry data is grouped by the material. Also, the library provides a notion of bins,

where a material and associated geometries with a higher bin number are rendered after the material and associated geometries with a lower bin number.

Programmable Pipeline

OpenGL ES 2.0 replaces the fixed function pipeline of OpenGL 1.x. Figure 7 shows the programmable pipeline of ES 2.0.

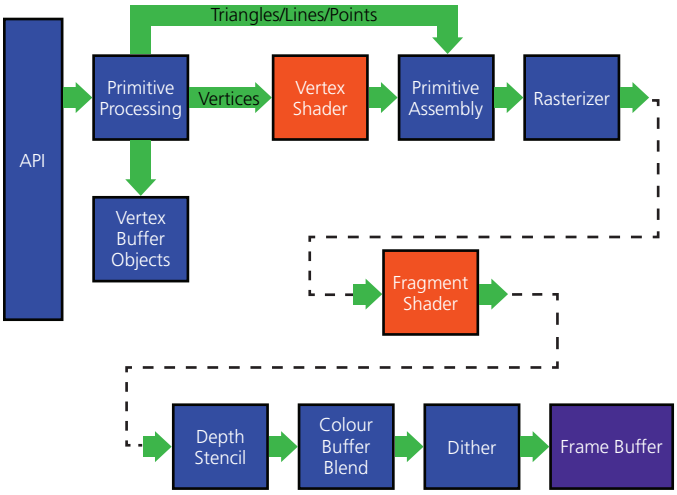


Figure 7: OpenGL ES 2.0 Programmable Pipeline (source: khronos.org)

In OpenGL ES 2.0, it is necessary to provide a vertex and a fragment shader in order to render geometry primitives. Vertex shaders can be used for traditional, vertex-based operations such as transforming the position with a matrix, computing the lighting equation to generate a per-vertex color, and generating or transforming the texture coordinates. The fragment shader is a general-purpose method for interacting with fragments. A simple in-source shader to render geometry in the VES library is shown in Listing 1.

```
// Define vertex shader to be used
// for rendering
const std::string vertexShaderSource =
   "uniform highp mat4 modelViewMatrix;
    uniform highp mat4 projectionMatrix;
    attribute highp vec4 vertexPosition;
    attribute mediump vec4 vertexColor;
    varying mediump vec4 varColor;
    void main()
    {
      gl_Position = projectionMatrix *
        modelViewMatrix * vertexPosition;
      varColor = vertexColor;
    }";

// Define fragment shader to be used for
// rendering
const std::string fragmentShaderSource =
   "varying mediump vec4 varColor;
    void main()
    {
      gl_FragColor = varColor;
    }";
```

Listing 1: Simple Shader to Render a Geometry in the VES Library

The way vertex attributes and uniforms are passed to the pipeline are somewhat different; but due to the fact that they are both inputs to the shader program, it is important to provide a consistent API to bind uniforms and vertex attributes to a shader. In Listing 2, we have shown how to pass uniforms and vertex attributes to the rendering pipeline.

```
// See listing 1 for vertexShaderSource and
// fragmentShaderSource
   ….
   ….
// Create shader objects
vesShader::Ptr m_vertexShader
   (new vesShader(vesShader::Vertex));
vesShader::Ptr m_fragmentShader
   (new vesShader(vesShader::Fragment));
// Set shader sources (from Listing 1)
this->m_vertexShader->setShaderSource
   (vertexShaderSource);
this->m_fragmentShader->setShaderSource
   (fragmentShaderSource);

// Add shader to shader program
this->m_shaderProgram->addShader
   (this->m_vertexShader);
this->m_shaderProgram->addShader
   (this->m_fragmentShader);

// Add uniforms to the shader program
this->m_shaderProgram->addUniform
   (this->m_modelViewUniform);
this->m_shaderProgram->addUniform
   (this->m_projectionUniform);

// Add vertex attributes to the shader program
this->m_shaderProgram->addVertexAttribute
   (this->m_positionVertexAttribute,
    vesVertexAttributeKeys::Position);
this->m_shaderProgram->addVertexAttribute
   (this->m_normalVertexAttribute,
    vesVertexAttributeKeys::Normal);
this->m_shaderProgram->addVertexAttribute
   (this->m_colorVertexAttribute,
    vesVertexAttributeKeys::Color);
// Shader program is an attribute of
// the material.
this->m_material->addAttribute
   (this->m_shaderProgram);
```

Listing 2: Uniforms and Vertex Attributes Binding to the Pipeline

In the previous example, we used predefined vertex attributes keys such as Position, Normal, and Color. These keys are provided for convenience (internally defined as enums) and applications are allowed to add a new key if desired. This design makes it convenient and flexible to bind multiple vertex attributes to the shader, where each attribute is identified by its key. The mechanism for linking vertex attributes to the geometry data using a key is explained in the following section on geometry data.

Geometry Data

The VES library provides a very flexible data structure for defining geometry for the purpose of rendering. Some of the highlights of the VES library geometry data are:

- Support for interleaved or separated data arrays
- Any number of coordinate systems for the point data
- Support for different basic types for the point data
- Separation of point data from the cell data
- Extensible data structure

VES geometry data structure is inspired by the collada file format specifications [9] where geometry is composed of one-or-more sources. Source in the VES library defines the

per vertex data to be used for rendering or computation purposes. For example, in Listing 3, we have shown a code snippet to define a source data, which stores the vertex positions of a geometry.

```
// Define vertex data structure
struct vesVertexDataP3f
{
   vesVector3f m_position;
};

// Define source data structure to store
   vertex positions
class vesSourceDataP3f :
  public vesGenericSourceData<vesVertexDataP3f>
{
public:
  vesTypeMacro(vesSourceDataP3f);

  vesSourceDataP3f() :
    vesGenericSourceData<vesVertexDataP3f>()
  {
    const int totalNumberOfFloats = 3;
    const int stride = sizeof(float) *
      totalNumberOfFloats;

    this->setAttributeDataType
      (vesVertexAttributeKeys::Position,
       vesDataType::Float);
    …
    …
  }
};
```

Listing 3: Source Data to Store Positions

It is to be noted that in the code above,vesGenericSourceData uses vesVertexData3f as the template argument. The VES library provides several such source data types for convenience. Developers can create their own source data type that can use either predefined vesVertexData* types or some other custom vertex data structure. Also in the previous section we mentioned that the vertex attribute binding to the shader uses attribute keys. In order to notify VES library to link a vertex attribute to the vertex source data it is required that they both share the same key value; for example, Position as shown in the example code.

VES library source data structure enables applications to define and use interleaved or separate arrays for each vertex attributes. Interleaved vertex arrays are useful to achieve performance in most situations. Information on how to construct cells from the source data is defined by the vesPrimitive. Just like source data, VES library geometry data can contain one or more instances of vesPrimitive. Listing 4 shows a code snippet to define triangle primitives.

```
vesPrimitive::Ptr triangles
  (new vesPrimitive());
vesSharedPtr< vesIndices<unsigned short> >
  indices (new vesIndices<unsigned short>());
indices->pushBackIndices(0, 3, 2);
indices->pushBackIndices(1, 0, 2);
triangles->setVesIndices(indices);
triangles->setPrimitiveType
  (vesPrimitiveRenderType::Triangles);
   triangles->setIndexCount(3);
   triangles->setIndicesValueType
  (vesPrimitiveIndicesValueType::
   UnsignedShort);
```

Listing 4: Creation of vesPrimitive of Type Triangles

The VES library enables developers to use unsigned short or unsigned int type for the indices based on the requirement and whether or not the device and the driver supports unsigned int as the index type.

Appearance and State Management

In the VES library, appearance and rendering state of the geometry is defined by the material used by the node of the scene graph. A material is composed of one or more material attributes. For example, shader program and textures are attributes of a material. Similar to a shader program (see Listing 2), a texture can be added to a material. VES provides support for 1D and 2D textures. Below is a list of data formats and types supported by VES library texture:

Internal and pixel format:
Alpha, Luminance, LuminanceAlpha, RGB, and RGBA.

Pixel data type:
PixelDataTypeNone, UnsignedByte, UnsignedShort565, UnsignedShort4444, UnsignedShort5551.

Control over depth buffer and blending is provided by vesDepth and vesBlend material attributes. The VES library API makes it easy to add a new material attribute in case the functionally is not provided by the existing attributes.

OpenGL ES Extensions

The VES library provides basic support to query OpenGL ES extensions available on a hardware. For example, using this feature, an application can query for the GL_OES_element_index_uint extension and if available, can use an unsigned int type for the index to support large geometry data. However, current implementations do not support applications acquiring function pointer for the extensions.

FrameBuffer Object (FBO)

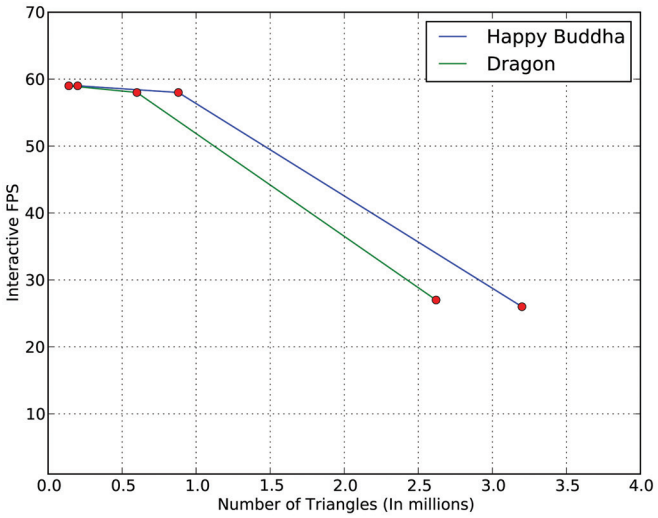
Framebuffer Objects are a mechanism for rendering to images other than the OpenGL default Framebuffer. OpenGL introduced FBO to make rendering to texture objects much more efficient and easier to perform when compared with pbuffer alternatives. The VES library enables applications to use the render to texture feature by creating an instance of vesRenderToTexture, setting appropriate parameters for the render target and setting the render target on the camera. Currently, VES supports attachments of type GL_COLOR_ATTACHMENT0 and GL_DEPTH_ATTACHMENT.

Interfacing with VTK

One of the strong motivations behind VES development is to enable developers to take advantage of state-of-the-art VTK algorithms on mobile devices. In order to provide an interface similar to VTK, the VES library offers vesActor and vesMapper. It is important to mention that the VES library does not depend on VTK as it is expected that an application that requires VTK support will be developed using the application level API provided by Kiwi.

RESULTS

In Figure 8 we show the performance of VES and Kiwi against different size models. We measured the rendering performance of the VES library using KiwiViewer [REF 10] on a third generation Apple iPad. We measured the interactive frames per second (FPS) by averaging the number of times rendering is triggered by the iPad in a 20 second interval. In order to get a consistent performance measure, we didn't include FPS value reported by the application at the start and exit of the application.



Number of Triangles	3263148	2914242
	879696	607560
	201720	143382

Figure 8: Performance Graph of VES Rendering

CONCLUSIONS & FUTURE WORK

The VES library, along with Kiwi, have been very successful in delivering a high performance application on iOS and Android platforms. Since the time we open sourced the VES and Kiwi libraries, we have received many contributions and much feedback from the community. We will continue to improve the current features of the VES library, and look forward to adding support for new features such as volume rendering and vector text rendering.

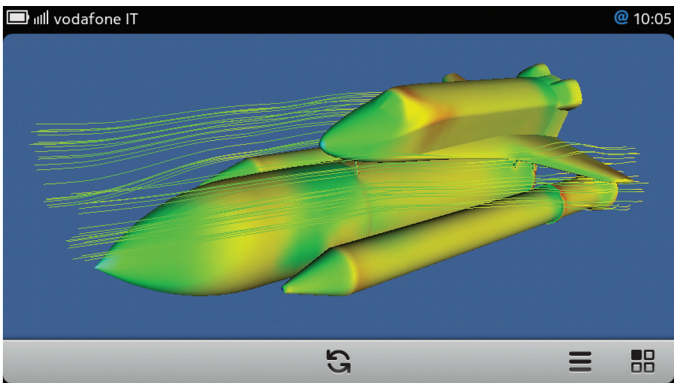


Figure 9: Rendering on Qt based MeeGO Platform

Special thanks to Riccardo Vianello for porting VES to the Qt-based MeeGO platform and providing screen captures for this article.

REFERENCES

[1] International Telecomunicaiton Union (ICT), Facts and Figures, http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf

[2] Be Mindful of OpenGL ES State Variables: http://developer.apple.com/libraryios/#documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/OpenGLESApplicationDesign/OpenGLESApplicationDesign.html

[3] Hong Zhou, Huamin Qu, Yingcai Wu, Ming-Yuen Chan. 2006. "Volume Visualization on Mobile Devices", Proceedings of 14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics'06).

[4] Tesch, J., Dietze, L. & Encarnação, L.M., 2003. Personal Interfaces-To-Go : Mobile devices for Data Exchange and Interaction in Heterogeneous Visualization Environments University for Applied Sciences (FH) Mainz. Distributed Computing, p.2-5.

[5] Roudaut, A., 2009. Visualization and interaction techniques for mobile devices. Proceedings of the 27th international conference extended abstracts on Human factors in computing systems CHI EA 09, p.3153. Available at: http://portal.acm.org/citation.cfm?doid=1520340.1520450.

[6] Visualization Toolkit (VTK), http://vtk.org

[7] OpenGL Performer, http://oss.sgi.com/projects/performer

[8] OpenGL ES, http://www.khronos.org/opengles

[9] COLLADA, https://collada.org

[10] KiwiViewer, http://www.kiwiviewer.org

[11] VES, http://www.vtk.org/Wiki/VES

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

ANNOTATION CAPABILITIES WITH VTK 5.10

At the Direction des Applications Militaires Île-de-France (DIF) of the Commissariat à l'Energie Atomique (CEA), France, an application-specific visualization tool has been developed to post-process the solutions to the numerical simulations calculated at CEA/DIF. On the visualisation side, this tool uses VTK and ParaView servers.

Goals of the collaboration focused on adding new annotation capabilities to VTK, specifically in the areas of extension of the existing axis actor and cube axes actor; addition of a new polar axes actor; and improvement and enhancement of the scalar bar actor.

New mesh manipulation capabilities have also been added, including element selection along a line or a broken line; broken line widget for interactive element selection with a mesh; and quadrangle-generating mesh extrusion from a multi-block polygonal input.

This article will briefly report on these new capabilities, all of which are available in VTK 5.10 (see recent releases).

CUBE AXES

Over the past few years, the 3D cartesian axis capabilities available with VisIt have been extended in order to intro-

duce new capabilities; in particular, the ability to display internal grid lines and / or translucent cut planes within cube axes actors. In addition, about 30 control parameters regarding bounds, titles, colors, and labels were added to the original classes, as well as a specific 2D mode to handle poor title/label appearance for two-dimensional results. We integrated these changes to the VisIt classes into VTK, attempting to retain the pre-5.10 VTK behavior by default whenever possible.

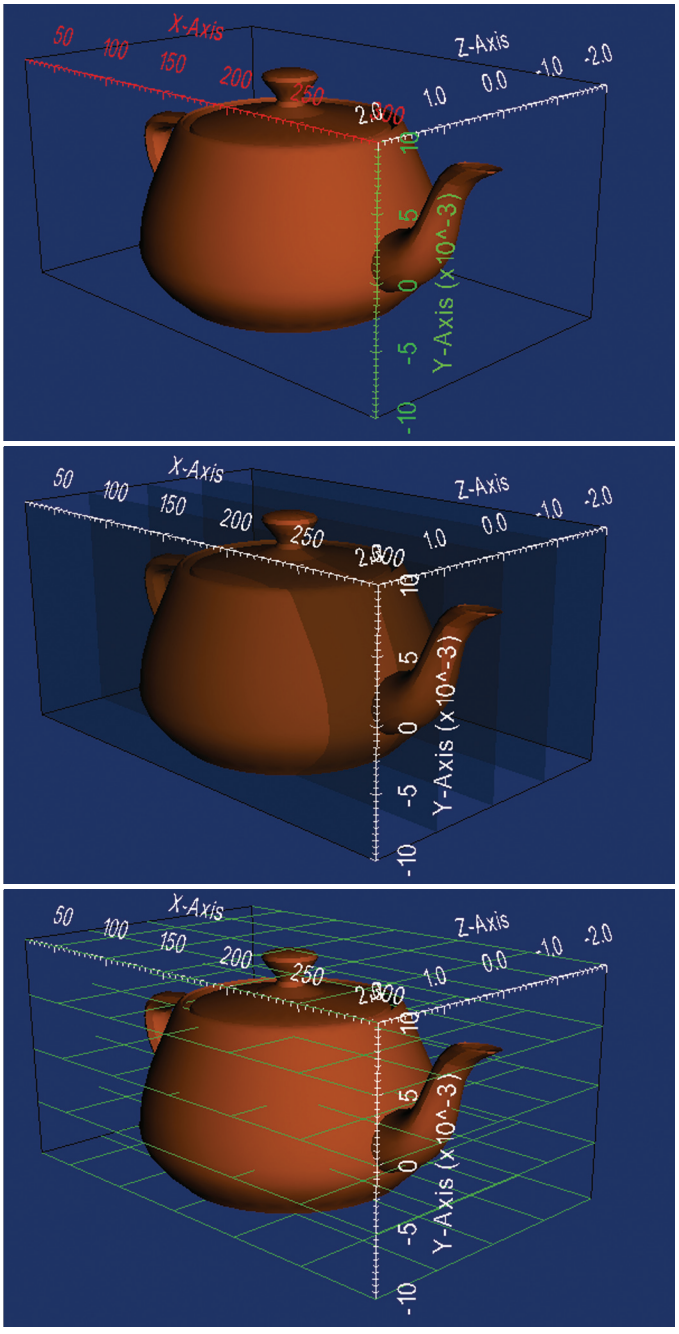


Figure 1: Use of Translucent Inner Planes and Inner / Outer Grids for Cube Axes Annotation

As illustrated in Figure 1, it is now possible to annotate cube axes with translucent inner planes and inner/outer grids. It is also possible to independently control settings for each axis including: axes bounds, text and color of each title, text and color of each axis and its labels, color and display of inner and outer grid lines; and color, translucency, and display of internal planes.

POLAR AXES

In the context of this collaboration, we created a polar axes actor, a feature which did not exist in VTK until 5.10. The main goal of this actor is to allow for the display of a variable number of radial axes, with respect to user-specified origin and polar angle, along with polar arcs attached to the radial tick marks. We developed this new actor from scratch, based on CEA specifications.

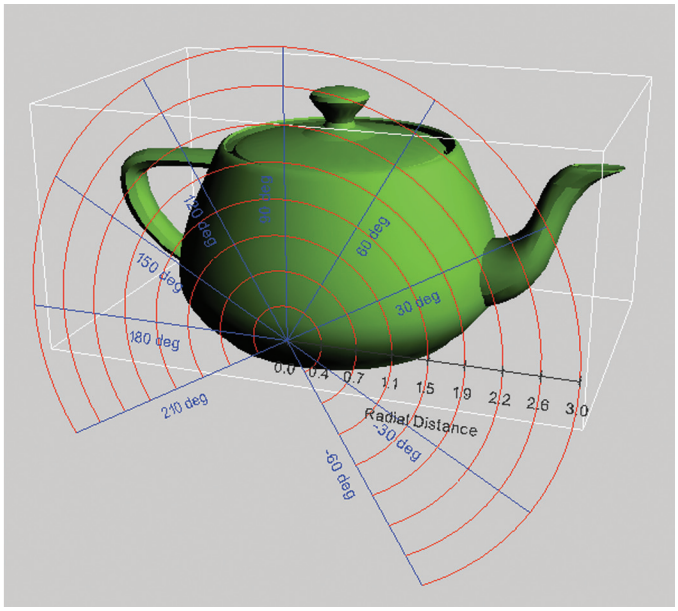


Figure 2: Multiple Parameters of the Polar Axes

As illustrated here, a number of parameters of the polar axes can be independently tuned, such as origin and polar angle; number of ticks and polar arcs (or auto-calculation option); number, display, color, and length of radial axes; number, display, and color of polar arcs; text attributes of axes and arcs titles and labels; and level-of-detail parameters.

SCALAR BAR

The class vtkScalarBarActor in VTK had been extended at CEA into a new class called vtkCEAScalarBarActor, in order to 1) modify the title and label placement within the scalar bar actor, in order to ensure that all text be placed within the interior of the specified 2D actor size, and 2) the addition of optional frame and background color, possibly translucent.

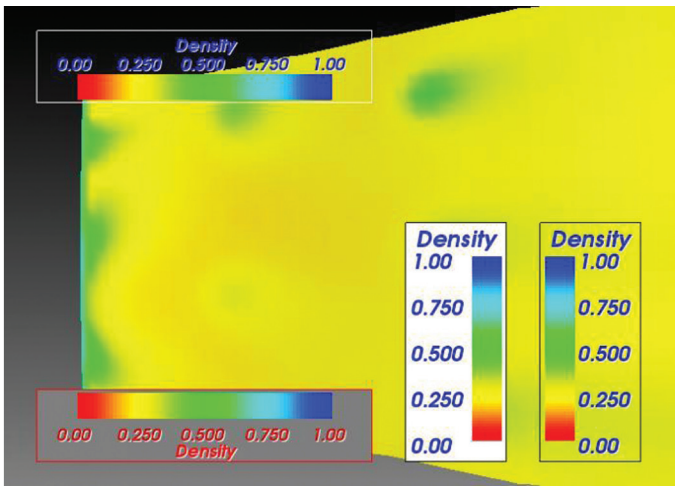


Figure 3: Scalar bar actor in VTK

As shown here, we ported these new features into the current VTK scalar bar actor. As a result, we had to modify its default behavior as it was not ensuring that all text remained within the frame, which led to visual defects when this option was added. We also used this opportunity to fix title placement behavior, which was exhibiting collisions with the color bar when placing letters such as 'q' or 'y'. We implemented these changes for both horizontal and vertical bar layout.

MESH SELECTION AND EXTRUSION CAPABILITIES

Linear Selector

A class for the selection of cells along a segment defined by its endpoints, within a composite dataset, had been developed at CEA/DIF. As part of our collaboration, we developed a new subclass of vtkSelectionAlgorithm, called vtkLinearSelector, which extends the method to support selection along broken lines within an input composite dataset. In particular, the API of this new selection algorithm allows for two different ways of defining a broken line: either by passing a list of vtkPoints, or by only specifying two endpoints in order to preserve backwards-compatibility for the CEA/DIF visualization application.

In addition, we enhanced the original method by allowing for the exclusion of line endpoints in the selection process; i.e., selection along open segments. We noticed with early experiments that typical use cases with CEA/DIF datasets were leading to numerous unwanted cells being selected in cases where a segment endpoint coincides (numerically) with a cell vertex within the input mesh. This exclusion method is based on an IncludeVertices option, which when turned off, makes use of a user-defined relative tolerance; thus defining a closed segment contained within, and numerically approaching, the actual open segment.

Incidentally, we discovered that the current implementation of the IntersectWithLine() method of vtkTriangle does not specifically handle the case of degenerate intersection; i.e., the (unstable) case where a segment is both coplanar and secant to a triangle. The current implementation works effectively as a dichotomy between either secant or coplanar, with the degenerate intersection belonging to the former case. This design is not faulty per se, for it was conceived as such for speed in the context of facet picking within a mesh. However, when using VTK in the context of topological/geometric manipulation, this becomes a problem as we discovered here. We initiated a discussion amongst VTK developers as to whether a modification of vtkTriangle::IntersectWithLine() is warranted in order to support the ternary division between generic intersection, non-intersecting coplanarity, and degenerate (co-planar) intersection. Due to the potential ramifications of such a change, we have decided to not not modify vtkTriangle::IntersectWithLine() until a consensus is reached in this regard. In the meantime however, this degeneracy does not cause any trouble for our CEA/DIF partners, for their visualisation application uses a version of vtkTriangle which they modified to handle this numerical issue in particular.

Broken Line Widget

Following the development of the broken line selection mechanism, we implemented a broken line widget to allow for interactive selection of cells within an input mesh. For consistency within VTK, we devised the API and behavior of the new vtkBrokenLineWidget to be similar to those of the vtkSplineWidget, so the developer already familiar with the latter will find it natural to use the former. In particular, mouse/button events produce the same results in terms of handle addition/deletion, panning, zooming, etc.

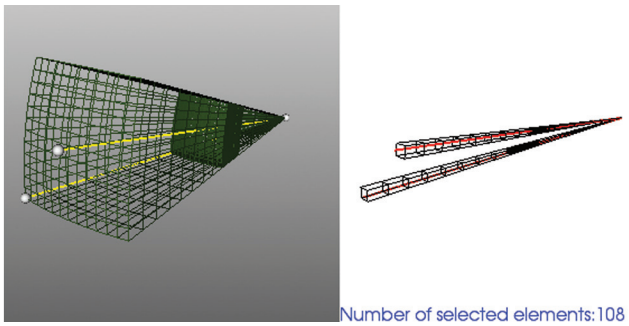


Figure 4: Broken Line Widget

The behavior of the new selection mechanism combined with broken line widget interaction can be interactively tested with the SegmentAndBrokenLineSources.py script, which we put under Examples/Graphichs/Python/ in VTK 5.10. It demonstrates the operation of the widget and lets the user interactively select cells within an example mesh, chosen for its peculiarity at the apex with multiple coincident pyramids, thus illustrating the use of endpoint elimination during the selection process.

Rotational Extrusion in Quadrangles

Among the modeling capabilities of VTK is the class called vtkRotationalExtrusionFilter, which takes polygonal data as input and sweeps it around the z-axis to create new polygonal primitives. These primitives form a "skirt" or swept surface. For example, sweeping a line results in a cylindrical shell, and sweeping a circle creates a torus. A number of control parameters are available: for instance, one can decide whether the skirt sweep of a 2D object (i.e., polygon or triangle strip) is to be capped with the generating geometry. It is also possible to control the angle of the rotation, and whether translation should be performed at the same time as rotation (screw action). The output skirt is generated by locating certain topological features: free edges (edges of polygons or triangle strips only used by one polygon or triangle strips), lines and polylines generate surfaces, whereas vertices generate lines.

This class, however, has the following limitations:

1. It can only generate triangle strips as output.
2. It can only sweep around the z-axis.
3. It does not allow native processing of multi-block inputs, with independent rotation angles specified for each block.

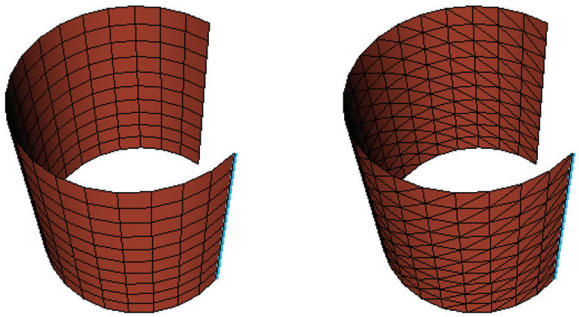


Figure 5: Rotational extrusion of a line parallel to the z-axis, in triangles (left) and quadrangles (right)

In order to address these limitations, a new class called vtkRotationalExtrusionFilter has been developed at CEA/DIF. In particular, the output of this new filter is a skirt made of quadrangles, as illustrated in Figure 5: left, the result of a rotational extrusion in triangles from a polyline containing 10 segments using vtkQuadExtrusionFilter; right, the output of vtkQuadRotationalExtrusionFilter applied to the same input, with the same parameters.

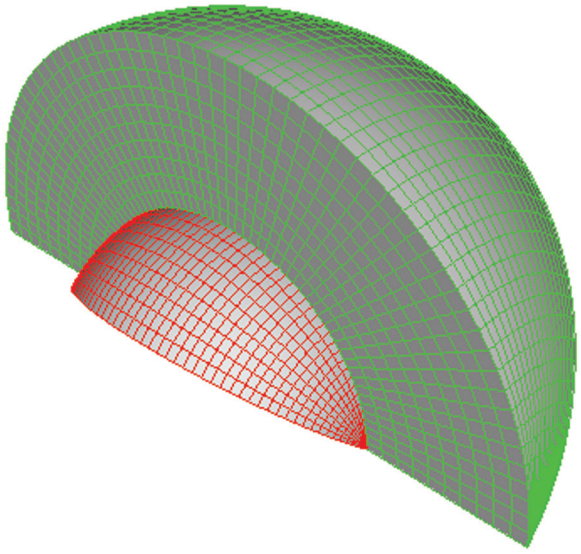


Figure 6: Rotational extrusion in quadrangles of a 2-block data set with a different angle for each block

As part of this collaboration with CEA/DIF, we ported the new class to VTK 5.10. We made a number of changes to make it easier to maintain and extend. In particular, we decided to keep it separate from vtkRotationalExtrusion, as opposed to merging the capabilities of both classes into a single new one; as illustrated above, vtkQuadRotationalExtrusionFilter is intrinsically a vtkMultiBlockDataSet algorithm, so we thought that for backwards-compatibility it was desirable to keep the triangle strip generating class as a vtkPolyDataAlgorithm, at least until a compelling reason to do otherwise would become manifest. The new, quadrangle-generating, multi-block filter is now fully-available with VTK 5.10, see Figure 6 above.

ACKNOWLEDGMENTS

This work was made possible thanks to a contract with CEA, Direction des Applications Militaires Île-de-France (DIF), Bruyères-le-Châtel, 91297 Arpajon, France. We extend special thanks to Daniel Aguilera, Thierry Carrard, and Claire Guilbaud for this fruitful collaboration. We are looking forward to continued collaboration with CEA/DIF in this and other areas of scientific visualization.

Philippe Pébay *is the Director of Visualization and HPC of Kitware SAS, the European subsidiary of the Kitware group. Pébay is currently one of the most active developers of VTK, an open-source, freely available software system for 3D computer graphics, image processing, visualization, and data analysis. He is in particular the main architect of the statistical analysis module of VTK and ParaView.*

IMPROVEMENTS IN PATH TRACING IN VTK

Given vector fields, or time-varying vector fields (flow fields), a common visualization approach is to "release" a set of particles into the vector field, move them according to the vectors, and observe their paths of travel. Specifically,

- Streamlines are paths taken by particles in a vector field.
- Pathlines are paths taken by particles in a flow field.
- Streaklines are curves created by releasing particles continuously into a flow field at fixed seed positions and connecting the particles released from common seed positions.

We have improved path tracing support in VTK:

- Created a faster parallel algorithm for streamline tracing.
- Improved usability of particle tracing filters for flow fields.
- Added a new filter to create streaklines.

FASTER PARALLEL STREAMLINES

The previous parallel algorithm for stream tracing is very simple and has the same asymptotic running time as the sequential algorithm. We have replaced it with a more sophisticated algorithm that scales reasonably well according to the number of processors. The algorithm starts by dividing the seed particles between the processes according to the data; each process then traces the streamlines one by one, and, in between, sends and receives particles that travel between processes. For AMR data, by using the metadata information, the particles can be sent to exactly the right processes; for other types of data, the particles are iteratively forwarded to the process next in MPI rank until they arrive at the right processes.

The algorithm is implemented by the class vtkPStreamTracer, which previously existed as an abstract super class. The child class vtkDistributedStreamTracer is depreciated. Conveniently, thanks to the VTK modular changes, the user does not have to decide whether to use vtkStreamTracer or vtkPStreamTracer depending on the presence of MPI; the object factory will instantiate a vtkPStreamTracer object whenever MPI is enabled.

In experiments, we observed that the algorithm scales reasonably well by the number of processes. Figure 1 shows the output of the parallel stream tracer, with an AMR Flash input dataset from an astrophysics simulation. The data set is 24G and distributed over eight machines with eight cores each. The colors of the trace lines are mapped to the processor ids.

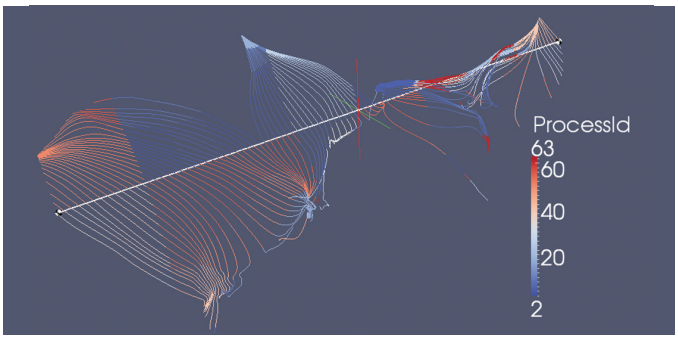


Figure 1: Parallel Stream Tracer with AMR Flash Input Dataset

EASIER-TO-USE PARTICLE TRACING

Previously, the main algorithm for performing particle tracing for flow field data was implemented in the class vtkTemporalStreamTracer [1]. This algorithm moves a set of particles one time step at a time according to a linearly-interpolated velocity field. It also supports parallel execution using MPI. In this work, we kept the algorithm mostly intact but focused on resolving a set of usability issues. Because the algorithm always takes one time step, the user has to manually step through the time steps by incrementing the time step parameter starting from zero and calling update(). Not starting from zero, skipping time steps, or decreasing time steps all result in errors. If particle paths are desired then the user has to connect an additional filter, vtkTemporalPathLineFilter, to run in tandem. The parallel support for vtkTemporalPathLineFilter is incomplete; it leaves gaps from particles hopping between processes. There is no streak line support.

We refactored the original vtkTemporalStreamTracer into two new classes, vtkParticleTracerBase and vtkPParticleTracerBase, for the serial and parallel version. We also augmented the original algorithm to internally step through the time steps from the user-defined starting time until the termination time. To be efficient, the new algorithm caches output so that if the only parameter and input change is the increased termination time, then a new run can start from the previous termination time instead of all the way from the starting time. Changing any upstream pipeline or any other parameters would invalidate the cache.

We created three new particle tracing filters that all can be used on their own. These filters are derived from the base class previously described and essentially do the same thing; they move a set of seed particles from the starting to termination time, but the outputs are different:

vtkParticleTracer/vtkPParticleTracer: The outputs are the particles at the termination time. In ParaView, we expect the user to use this filter to animate the moving particles. Therefore, the termination time is not exposed on the GUI. Rather, it is updated according to the pipeline time (vtkStreamingDemandDrivenPipeline::UPDATE_TIME()), which can be changed using the animation controls.

vtkParticlePathFilter/vtkPParticlePathFilter: The outputs are a set of polygonal lines representing the paths particles took.

vtkStreaklineFilter/vtkPStreaklineFilter: The outputs are a set of polygonal lines representing the streaklines.

Figure 2 shows the three particle tracers running on a dataset of ten time steps.

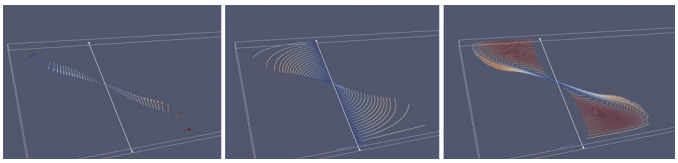


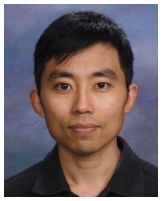
Figure 2: From left-to-right: a snapshot of particle tracing; pathlines; and streaklines. The colors are mapped to the magnitude of the velocity vectors.

ACKNOWLEDGEMENT

We would like to thank Paul Sutter and Mohamad M. Nasr-Azadani for providing the data used to produce the images used in this article.

REFERENCES

[1] John Biddiscombe, Berk Geveci, Ken Martin, Kenneth Moreland, David Thompson: Time Dependent Processing in a Parallel Pipeline Architecture. IEEE Trans. Vis. Comput. Graph. 13(6): 1376-1383 (2007)



Yuanxin "Leo" Liu *joined the Kitware team in January, 2012 as an R&D engineer. Prior to joining Kitware, he was a senior R&D software engineer at Geomagic Inc. and developed algorithms to process point clouds and polygonal meshes.*

IMAGE-GUIDED INTERVENTIONS TUTORIAL WITH IGSTK

The Image-Guided Surgery Toolkit (www.igstk.org) is a free, open-source toolkit that provides a framework for the development of image-guidance applications. In addition to providing an infrastructure, one of its goals is to serve as an educational resource for training newcomers to the field [1]. While this has been one of our goals from the beginning, the toolkit itself does not address it per se. Most often, newcomers find it intimidating to implement a brand new program using the toolkit components. To ease the burden, the toolkit includes several example applications. Most users prefer to start their development by modifying these examples. While this is helpful for application developers, these programs are not appropriate for teaching common concepts in image-guided interventions (IGI).

Our goal in this tutorial was to provide a set of programs that allow students to gain hands-on experience with IGI, enhancing their understanding of theoretical concepts. When teaching IGI, we describe algorithms based on mathematical models of the physical world. Students understand the models, can reiterate the derivations, but often do not seem to connect the models with the real world. For example, the fact that Fiducial Registration Error (FRE) does not reflect the quality of registration in paired point rigid registration is understood. But, when a student sees it happen with a navigation system in the physical world, there is an "Aha!" Moment. It is these moments that solidify their understanding of the theoretical concepts. Theory has been validated by practice. This tutorial was developed with the intention of eliciting these "Aha!" Moments.

Finally, in an era of rising higher education costs, we were resolved to provide an accessible tutorial with minimal cost to the users.

ACCESSIBILITY

Two hurdles stood between us and providing a tutorial that is accessible to a wide audience; the cost and availability of imaging and tracking devices, key components required by many IGI systems. To address the former issue, we followed the example of Pace et al. [2]. We too provide CT scans of a simple LEGO phantom. Given that LEGO blocks are manufactured in a consistent manner, the scans can be readily used once the user builds the corresponding phantom.

To address the second issue, we realized that our initial observation that low cost tracking systems are not widely-available is mistaken. These types of systems are all around us. If you are reading this article on your laptop, then most likely you are staring directly at a tracking device: your computer's webcam. All we have to do is calibrate these consumer-grade cameras to transform them into monocular tracking devices. Figure 1 shows two tracking devices that represent the opposite extremes of optical tracking, as compared using a variety of evaluation measures (i.e. accuracy, cost, refresh rate, etc.): a consumer grade webcam and a high end tracking system.

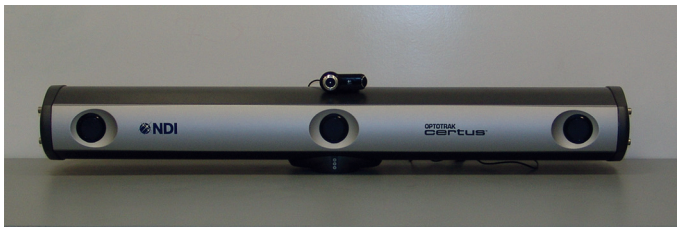


Figure 1: The two extremes of optical tracking, a webcam, QuickCam Pro 9000 from Logitech Inc., and the Optotrack Certus from Northern Digital Inc.

In the context of the Image-Guided Surgery Toolkit, all tracking devices are used in a similar manner with the only difference being in the initialization code, as illustrated by the following code snippets.

Initialization of the webcam based tracker:

```
/* Instantiate the tracker */
  igstk::ArucoTracker::Pointer tracker;
  tracker = igstk::ArucoTracker::New();
/* Setup tracker */
  tracker->SetCameraParametersFromYAMLFile
    (cameraCalibrationFile);
  tracker->SetMarkerSize(50);
/* Instantiation and setup of tracker tool */
  igstk::ArucoTrackerTool::Pointer trackerTool =
    igstk::ArucoTrackerTool::New();
  trackerTool->RequestSetMarkerName("102");
/* Common code {*/
/* Frequency setting and opening communication */
  tracker->RequestSetFrequency(30);
  tracker->RequestOpen();
/* Tracker tool configuration, tracker attachment
   and start tracking */
  trackerTool->RequestConfigure();
  trackerTool->RequestAttachToTracker( tracker );
  tracker->RequestStartTracking();
/* } common code*/
```

Or the equivalent for the NDI Certus tracker:

```
/* Instantiate the tracker */
  igstk::NDICertusTracker::Pointer tracker;
  tracker = igstk::NDICertusTracker::New();
/* Setup tracker */
  tracker->SetIniFileName(CertusSetupFile.c_str());
  tracker->rigidBodyStatus.lnRigidBodies = 1;
  tracker->rigidBodyDescrArray[0].lnStartMarker = 1;
  tracker->rigidBodyDescrArray[0].lnNumberOfMarkers
= 4;
  strcpy(tracker->rigidBodyDescrArray[0].szName,
    rigidBodyFile.c_str());
/* Instantiation and setup of tracker tool */
  igstk::NDICertusTrackerTool::Pointer trackerTool =
    igstk::NDICertusTrackerTool::New();
  trackerTool->RequestSetRigidBodyName
    (rigidBodyFile.c_str());
/* Common code above */
```

TUTORIAL STRUCTURE

Our tutorial is comprised of a set of programs allowing the user to carry out a simulated biopsy procedure using a needle-like tool. The end goal is to provide guidance so that the needle tip is inserted to the desired target point inside the "body," without going through critical structures. In many aspects, this procedure is an advanced version of the children's game "Operation" (http://www.hasbro.com/games/en_US/operation/?page=history).

The tutorial programs are built using IGSTK and Qt, with the software architecture shown in Figure 2. To carry out the biopsy procedure the user performs setup and system calibration, procedure planning, and navigation, as described in the following sections.

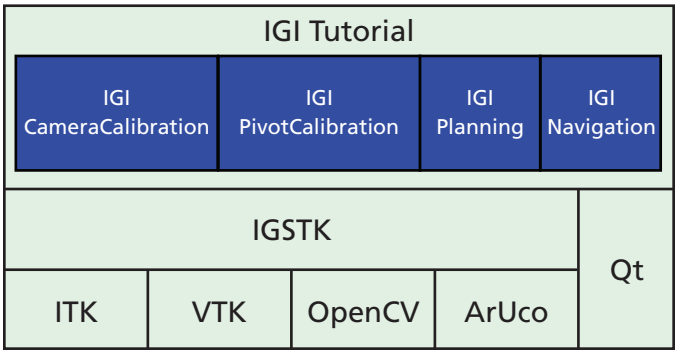


Figure 2: IGI-tutorial software architecture

SETUP AND CALIBRATION

This component is specific to our tutorial as most actions are not required or desired in clinical practice. Initially, the user has to build the phantom, print calibration and tracking patterns, and construct a tracked pointer tool. The camera is then transformed into a tracking system using the provided calibration software. Finally, the pointer tool is calibrated; that is, the location of the tool tip relative to the attached marker is estimated.

The software components used in this logical step are:

IGICameraCalibration: This application is used to calibrate the camera in order to use it as a tracking device. After calibration there is a calibration quality assessment or validation step. A pattern with known distances is detected and the estimated distance is overlaid onto the pattern's image.

IGIPivotCalibration: This application is used to calibrate a pointer tool in order to track the tool tip.

PLANNING

The preoperative planning software allows the user to specify the system setup, and identify registration fiducials and target points in the CT. The software component used in this step is the IGIPlanning program.

The user first selects the image dataset and the camera and pointer calibration files. Then, tracking setup is performed by selecting the marker used for the dynamic reference frame (DRF), the pointer tool, and additional arbitrary tools that will be displayed in the navigation's 3D view. The DRF is then rigidly attached to the phantom, allowing us to freely move the phantom during the navigation phase as all measurements will be relative to this coordinate frame which maintains its relationship with the phantom. Finally, the user identifies registration fiducials and target points by

scrolling through the volumetric dataset, which is displayed using the standard radiological reformatted views. Figure 3 shows the user interface for selecting the tracked markers and the fiducials.

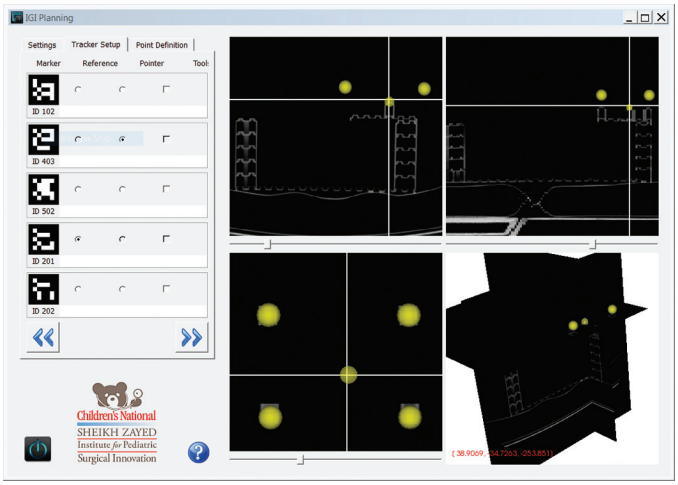


Figure 3: Planning interface. On left, user interface for tracker setup. On right, multiplanar views showing the image dataset with the registration fiducials overlaid as yellow spheres.

NAVIGATION

The software component used in this step is the IGINavigation program. This application displays standard radiological views with the displayed planes corresponding to the location of the tracked pointer tool's tip.

To perform navigation, the user loads the configuration file generated in the planning phase. Next, they need to perform paired point rigid registration. At this point, we already have the fiducial locations in image space, so we need to identify the corresponding points in the physical space, which are obtained with the tracked pointer tool. Lastly, the two point sets are registered and the user can start navigating. The navigation display shows axial, sagittal, and coronal planes corresponding to the pointer tool's tip location.

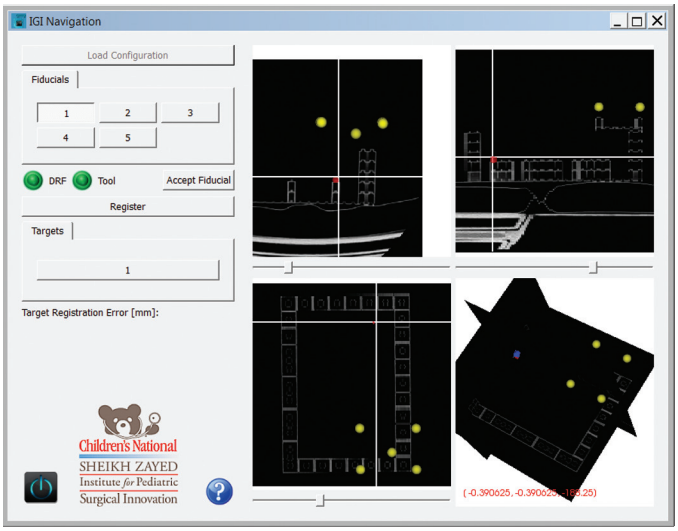


Figure 4: Navigation software. Yellow spheres represent registration fiducials. Red sphere represents the target. Cross-hairs follow the tooltip (tool tip is on the defined target in physical world) and reslices the planes accordingly.

The quantity describing the root mean square error of the fiducial locations after registration, the Fiducial Registration Error (FRE), is reported immediately after registration. Clinically this is irrelevant as we are interested in the error at the target locations, the Target Registration Error (TRE). During planning, the user can select a target in the program's interface and the distance between that and the actual target defined in the image; the current visualized tip position according to the tool tip position in the physical space is displayed. Figure 4 shows the program during navigation.

Having given a general description of the tutorial, the following sections proivde two examples of using it to enhance the students' learning experience.

EXAMPLES OF "AHA!" MOMENTS

FRE does not reflect TRE

In solving the paired point rigid registration problem the registration algorithm finds the transformation that best aligns the two point sets in a least squares sense. The quality of the alignment with regard to the points used for registration is given by the FRE. The quantity we are actually interested in is the TRE, which is uncorrelated with FRE [3]. To drive this point home, we have the students perform registration twice: once with the phantom constructed correctly, and then with a shifted fiducial configuration. In both cases both FRE and TRE are estimated. Figure 5 shows results from this experiment, exhibiting considerable changes in TRE even though there is minimal change in FRE.

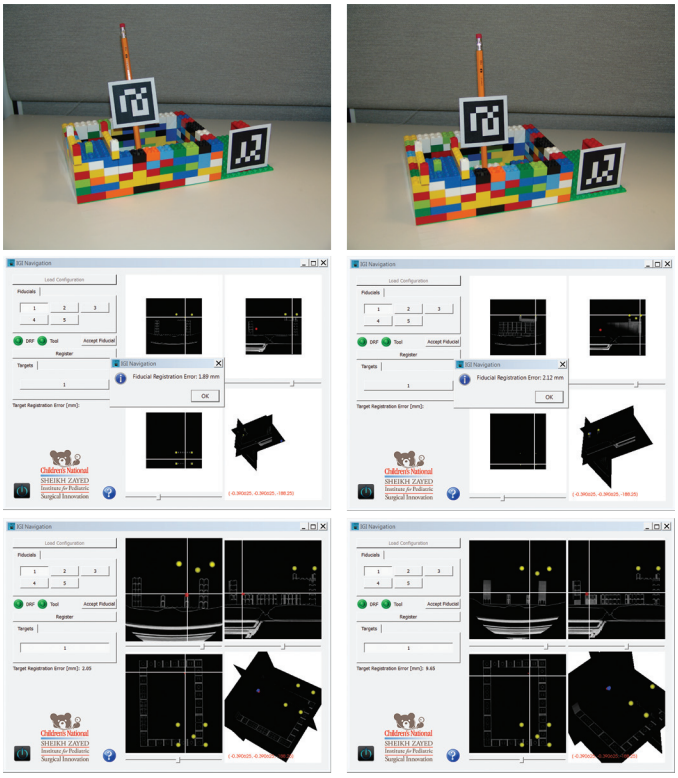


Figure 5: On the left we have a correctly (corresponds to the CT data) constructed phantom, resulting in a FRE of 1.89mm and TRE of 2.05mm. On the right, the fiducial configuration is shifted one step from the correct location. The FRE is 2.12mm, very similar to that obtained for the correct configuration. On the other hand, the TRE is 9.65mm, exhibiting the difference between the CT data and the physical world.

The effect of rotational errors on point location (lever effect)

The effect of rotational errors on point location is dependent upon the distance of the point from the center of rotation. This is often referred to as the "lever effect:" the closer the point is to the origin, the less it is affected by the rotational errors. This is illustrated by the pivot calibration procedure used to estimate the tip of a pointer tool relative to a tracked marker.

When performing pivot calibration, the pointer is pivoted and rotated while keeping its tip at a fixed location. The marker's poses are used to estimate the tip location using a linear least squares approach. Due to the lever effect, calculating the tip offset from the marker origin is less accurate the larger the distance is between the two. To drive this point home, we have the students perform pivot calibration twice, once with the marker close to the tip, and once far from it. The calibration software reports the root mean square error which reflects the tip estimation error. Figure 6 shows results from this experiment, exhibiting a larger error for the larger distance.

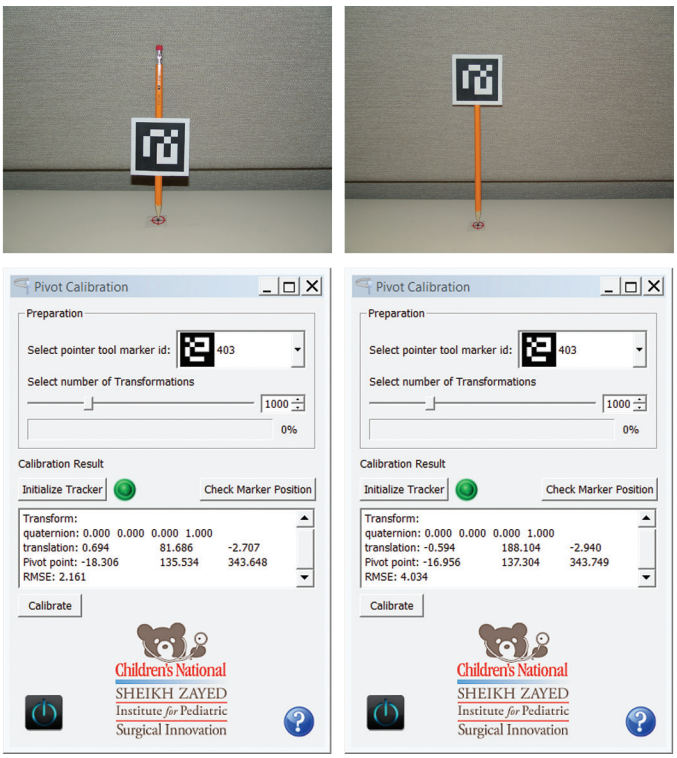


Figure 6: Effect of distance between the tracked marker's origin and the pointer tip (a.k.a lever effect) on accuracy. The shorter distance results in a RMS of 2.16mm vs. the 4.03mm for the longer one.

CONCLUSIONS

We have described a hands-on tutorial for illustrating various concepts in image-guided interventions. This educational tool is intended to enhance the understanding of theoretical concepts taught in class. Download the tutorial from:

http://public.kitware.com/IGSTKWIKI/index.php/IGI_Tutorial

REFERENCES

[1] "IGSTK: Building High Quality Roads with Open Source Software", L. Ibanez, A. Enquobahrie, M. Turek, J. Jomier, R. Avila, P. Cheng, Z. Yaniv, F. Lindseth, K. Gary, K. Cleary, Workshop on Systems and Architecture for Computer Assisted Intervention in conjunction with MICCAI, 2008.

[2] D. F. Pace, R. Kikinis, and N. Hata, "An accessible, hands-on tutorial system for image-guided therapy and medical robotics using a robot and open-source software," Open Science Workshop in conjunction with MICCAI, 2007.

[3] J. M. Fitzpatrick, "Fiducial registration error and target registration error are uncorrelated", SPIE Medical Imaging: Visualization, Image-Guided Procedures, and Modeling, 2009.

Özgür Güler *is a post-doctoral fellow at Children's National Medical Center. His research interests include development, assessment and application of image guided surgery systems, medical image processing, medical registration and registration error assessment, and software development. He is one of the primay developers of the Image-Guided Surgery Toolkit.*

Ziv Yaniv *is a principal investigator at the Sheikh Zayed Institute for Pediatric Surgical Innovation, Children's National Medical Center. His main areas of interest are image-guided interventions, medical image analysis, computer vision, and software engineering. He actively supports the development of open source software, and is one of the lead developers of the free open source Image-Guided Surgery Toolkit and a contributor to the Insight Registration and Segmentation toolkit (ITK).*

COMMUNITY SPOTLIGHT

The following article highlights how some of our community members are utilizing Kitware's open-source toolkits and products in their own development environment. If you would you like to be featured as a Community Spotlight, send your article ideas and materials to editor@kitware.com.

CREATING A VIRTUAL BRAIN ATLAS FOR MEDICAL EDUCATION USING ITK

Question: How can a pea-sized brain tumor cause such seemingly incongruous symptoms as double vision, abnormal bone growth, and infertility? Answer: Because anatomic neighbors are frequently affected by the same tumor, stroke, or injury. This is why it is crucial that medical students learn the complex spatial relationships of functionally important brain structures and pathways, most of which are not visible on the surface. After many years teaching medical neuroanatomy, we know that students find it very difficult to create a mental picture of these spatial relationships using traditional resources, such as atlases of stained-sections of brain tissue and plastic models. A few years ago, given exciting advances in 3D medical imaging, we thought surely a digital atlas of the human brain had been developed that provided direct visualization of internal brain anatomy. After an extensive search, we were surprised to find that none of the available resources met our students' needs; in particular, all of them failed to show dozens of important structures that are too small or lack sufficient contrast to be resolved on MRI or CT scans.

IT TOOK A TEAM

In retrospect, it shouldn't have been very surprising that details were missing from digital 3D brain models, since manual segmentation of such 'invisible' structures requires a lot of time and patience combined with a detailed knowledge of neuroanatomy, two things that don't often coexist in one person. We believed that a team approach could be successful, so we decided to work together to create a virtual brain atlas tailored to our curriculum and designed to help students learn human brain anatomy. Our team included a first year medical student willing to spend his summer doing manual segmentation (Joseph Rozell), a neurosurgeon (Charles Kite), a neuroanatomist (Norman Strominger), and a neuroscience educator and grant writer (Tara Lindsley). Since high resolution MRI images are essential for accurate segmentation and rendering, we obtained a head MRI of a normal volunteer on a powerful GE GENESIS-SIGNA 3 Tesla MRI, courtesy of the Neurosciences Institute at Albany Medical College.

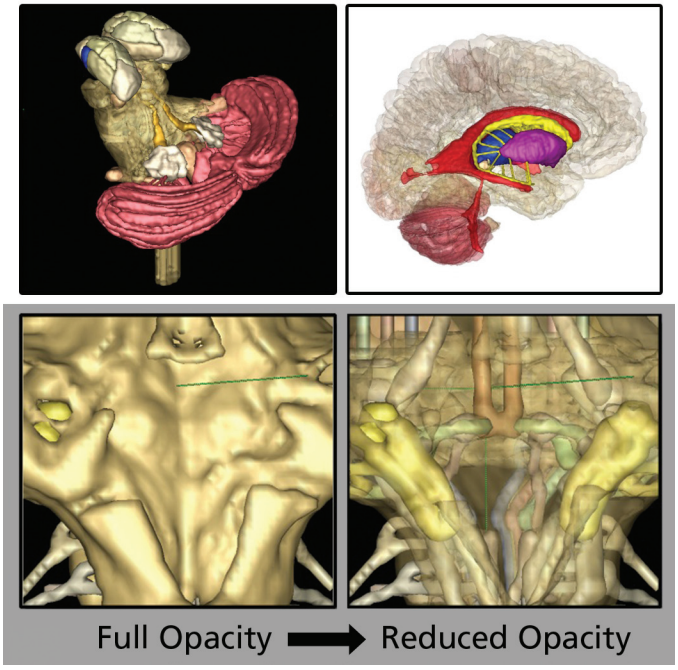
ITK-SNAP WAS THE IDEAL SOFTWARE

The team agreed that our overarching goal was to create the missing resource, study how it is used, verify its effectiveness in our curriculum, and then make it as widely available as possible. The intention to share our work led naturally to a search for open-source software. We found that ITK-SNAP had all the visualization, image processing, segmentation, and registration algorithms we needed and was easy enough to download and use, even for a team without IT support or programmers. ITK-SNAP is mainly developed by a team at the University of Pennsylvania, and is based on the ITK, VTK, and FLTK toolkits. Several other programs had similar capabilities, such as zooming and rotating the 3D models. However, several features made ITK-SNAP ideally suited for use by medical students and the way we imagined they would use the models: 1) the student could build custom models and save and share them; 2) they could 'see through' superficial structures to appreciate how they are spatially related to deeper structures using the opacity settings; and 3) the student could see how structures in the 3D window correspond to their appearance in MRIs in standard clinical views, making it more relevant for future physicians.

OUT OF 24 MILLION PIXELS, 79 STRUCTURES AND SOME PROBLEMS

The raw MRI data files contained 326 axial, 248 sagittal, and 300 coronal slices, which corresponded to a staggering combined total of roughly 24 million pixels. Seventy-nine structures and pathways were manually segmented in the sagittal slices by moving serially through the stack and completing one structure at a time. This was arduous and required lots of educated approximation of the location of structures not discernible on MRI. Members of the team and print atlases were consulted frequently to ensure accuracy [1,2]. A prototype of the software that used ITK-SNAP to view the image database was distributed to 135 first-year medical students and 6 faculty the following year. When we asked them about their experiences with the software, we learned that most users were thrilled with the visualization capabilities and thought the models were extremely effective for learning the spatial relationships of brain structures.

However, some students complained about the many steps that are required to download and install ITK-SNAP, and problems loading the image files. More worrisome was the feedback indicating that many users unintentionally applied edit functions with unhelpful results. Suggestions for improving the software were addressed the following summer, taking advantage of the open source nature of the software package, by creating a simplified version of ITK-SNAP [4], without the 3D rendering capability. This required adding a team member with some computing skills (Aaron Asch, an undergraduate).



AN EVEN BETTER BRAIN MODEL (V2.0)

Aaron used a CVS client and open-source utilities, such as CMake and GCC along with the Visual C++ Express IDE to compile and work with ITK-SNAP, and relied on useful instructions in the source code. The GUI editor "Fluid" from the FLTK toolkit was used to rearrange components of the user interface, adding menu buttons to simplify commonly used tasks and to introduce new educational functions. In keeping with open-source practices, this modified code was posted back to the developer's website. Key features of this user-only version of the Virtual Brain Model (v2.0) for education are illustrated in Figure 2:

- Software binaries, image database, and instructions constitute a file size of 150 megabytes.
- Download and install takes just 2 clicks, then the software can be launched from the desktop icons.
- A simplified Main Screen retains the clinical MRI views and 3D window (Fig. 2A), but has new menu buttons, including:
  - Center Axes
  - Show All or Hide All structures
  - Open pre-built models corresponding to specific learning objectives (Fig. 2B)
  - Save Model and My Models to simplify saving custom models and accessing them later
- A Selected Structure Description window names the structure at the crosshairs (Fig. 2C).
- The structure browser includes a Groups list to select all subsets of certain larger structures (Fig. 2D).
- Easy access pull-down menus, including a step-by-step tutorial in the Help menu

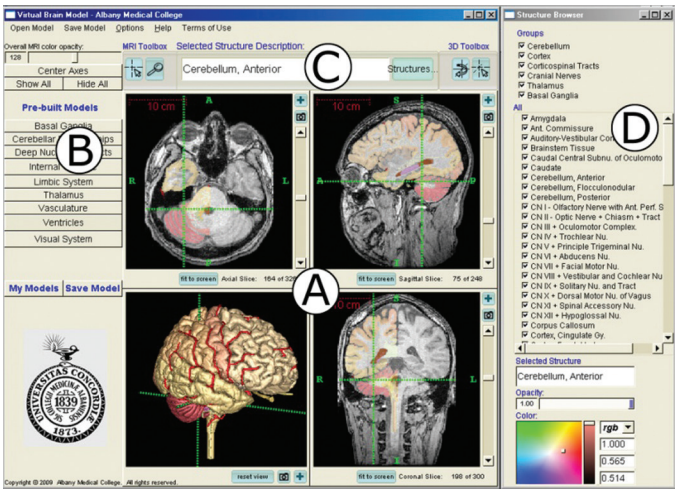


Figure 2: AMC Virtual Brain Model Software Main Screen

This version of the software has been used by three classes of medical students, and just as the original version of ITK-SNAP, it is open-source and available for download at www.amc.edu/academic/software. An abstract describing this work has been published [3], and an analysis of learning outcomes is underway.

WHAT'S NEXT? KITWARE ON THE TEAM

We plan to publish a paper describing the development and use this software and will submit the resource to the MedEd Portal to target its dissemination to medical and allied health professions schools. But first, we recognize a few additional changes are needed. Users around the world are asking for compatibility with other OS, an improved tutorial with guided exercises, and other changes. We are excited about the next version of the Virtual Brain Model software which will benefit from the open-source and programming expertise of Kitware. We hope the ITK open-source community will take the AMC Virtual Brain Model v2.0 for a test drive and offer us some suggestions for an even better brain.

REFERENCES

[1] Miller, RA and Burack, E. Atlas of the Central Nervous System in Man. 3rd Ed. Williams & Wilkins, Baltimore, MD. 1982.

[2] Haines, DE. Neuroanatomy: An Atlas of Structures, Sections, and Systems. 6th Ed. Lippincott, Williams & Wilkins, 2004.

[3] Rozell J, Strominger NL, Lindsley TA, Kite CH. Program # 23.5 2009 Neuroscience Meeting Planner. Chicago, IL: Society for Neuroscience, 2009. Online.

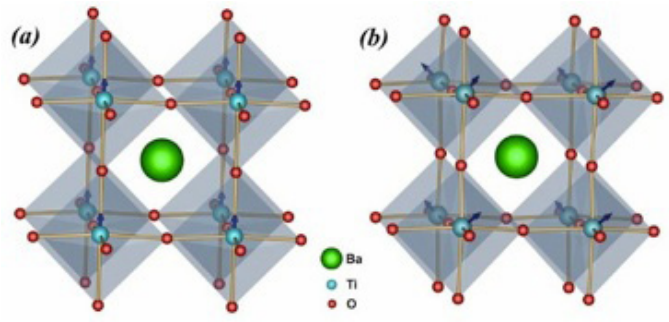
[4] ITK-SNAP: http://itksnap.org/pmwiki/pmwiki.php?n=Main.Credits

Tara Lindsley *is a Professor and the Thelma P. Lally Chair of Neuroscience Education at Albany Medical College in Albany, NY. She is a member of a team of faculty and students developing open-source virtual human anatomy atlases for education by creating manually segmented image databases and simplified, user-only software on the ITK infrastructure. Release of virtual female pelvis model software is planned for 2013.*

KITWARE NEWS

KITWARE HOSTS IEEE SCIVIS CONTEST DATA

Kitware is hosting the IEEE SciVis Contest 2012 data on an installation of Midas, Kitware's open-source toolkit that enables the rapid creation of tailored, web-enabled data storage. It is optimized for efficiently centralizing, indexing, and storing massive collections of data, providing the foundation for computational scientific research.



In this regard, Midas is perfect for supporting this year's IEEE SciVis contest, which targets the field of computational material science, particularly atomic configurations. The goal of this year's contest, part of IEEE VisWeek 2012, is to devise a visualization that allows for exploring the phase transitions of a particular ferroelectric material while decreasing the temperature gradually.

A repository with this and previous years' contest data is available in easily-navigable communities. Further contest details and the public data for the contest are available at http://midas3.kitware.com/midas/community/1#tabs-info.

INNEROPTIC AND KITWARE COLLABORATION

InnerOptic Technology and Kitware announced NIH Phase II SBIR funding for the development of a needle guidance system for hepatic tumor ablation. The operating room ready system will provide novel 3D visualizations for needle guidance in soft tissues. Brian Heaney is the PI and will lead and oversee the project at InnerOptic, and Stephen Aylward will lead the Kitware effort.

Using InnerOptic's Spotlight™ technology, which was developed during Phase I of this grant, intra-operative ultrasound images will be fused with pre-operative computed tomography (CT) images. The Spotlight system renders opaquely and in sharp detail only the portions of the CT data that are in the vicinity of the ultrasound probe or the needle trajectory. This is analogous to a spotlight on stage: illuminating the scene of interest, while the rest of the stage is transparent and out-of-focus to be less distracting.

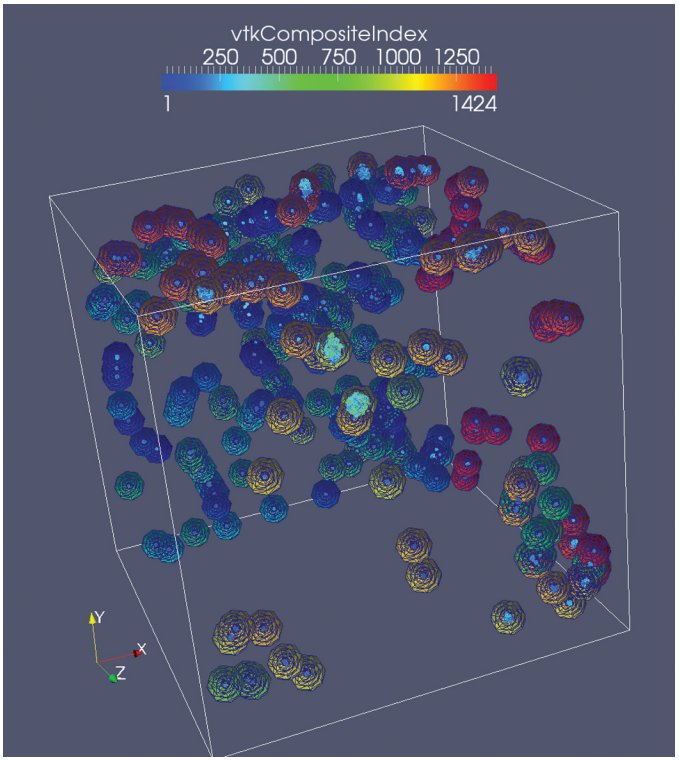
During liver lesion ablations and other image-guided procedures, surgeons and interventional radiologists currently must rely on mentally-integrated information from several imaging modalities. While CT imaging has excellent diagnostic value, breathing and surgical manipulation can cause tissues to move and deform. Additionally, intra-operative ultrasound images are available in real-time but have a limited field-of-view and can be less effective than CT at distinguishing tissues and pathologies. Physicians must therefore alternate between viewing annotated pre-operative CT

images or live ultrasound images on separate monitors, with no interaction between them. The awarded Phase II grant will extend Spotlight with registration algorithms that will keep the CT and ultrasound images continuously aligned. The work will result in a radically improved workflow for using CT and ultrasound images in image-guided soft tissue procedures.

National Institutes of Health Acknowledgement and Disclaimer: Research reported in this publication was supported by the National Cancer Institute of the National Institutes of Health under Award Number R44CA143234. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

KITWARE WINS THREE DOE SBIR AWARDS

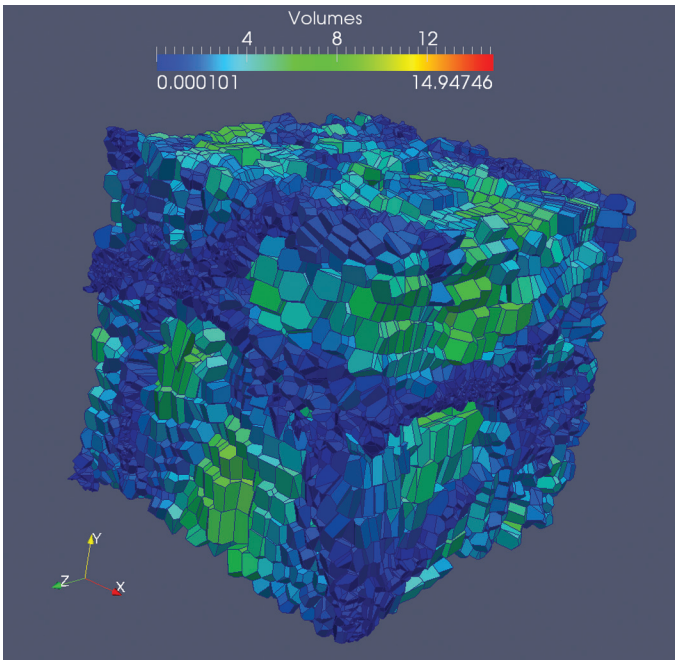
In April, Kitware announced $450,000 in Department of Energy SBIR Phase I awards. The awards will fund three projects focused on scientific computing: "In-Situ Analysis of Cosmology Data," "Cloud Computing and Visualization Tools for KBase," and "Graphical HPC Application Suite for Supporting the Product Simulation Lifecycle."



This DensityProfile shows the set of concentric spheres centered at each halo center and DensityProfilePlot shows the corresponding particle density, i.e., number of particles, within each sphere.

In-Situ Analysis of Cosmology Data

Dr. Berk Geveci will lead the development of a cosmology analysis framework for in-situ analysis and data-reduction of cosmological simulations. Such simulations play an important role in the DOE's High Energy Physics Cosmic Frontier program and are critical for understanding dark energy. This project requires an evolutionary shift in the way cosmological predictions are obtained, and therefore addresses critical challenges such as workflow I/O and the lack of domain-specific data analysis algorithms. An infrastructure will be developed that provides data reduction for minimizing input and output, robust and efficient halo-extraction methods, online tracking of halos to capture halo formation dynamics, and in-situ and co-visualization capabilities.



The voronoi tesselation used in identifying voids in cosmological simulations. This is being developed by Peterka et al. (http://www.mcs.anl.gov/uploads/cels/papers/P2087-0512.pdf)

In addition to its use in cosmology simulations, the developed framework will have wide applicability in industries that use particle-based simulation techniques, including astrophysics, ballistics and defense, volcanology, oceanology, solid mechanics modeling, and a range of maritime applications. The adaptability of this framework for use with large-data simulations will drive new levels of innovation in the computational sciences and facilitate the necessary transition from terascale work to peta- and exascale computing.

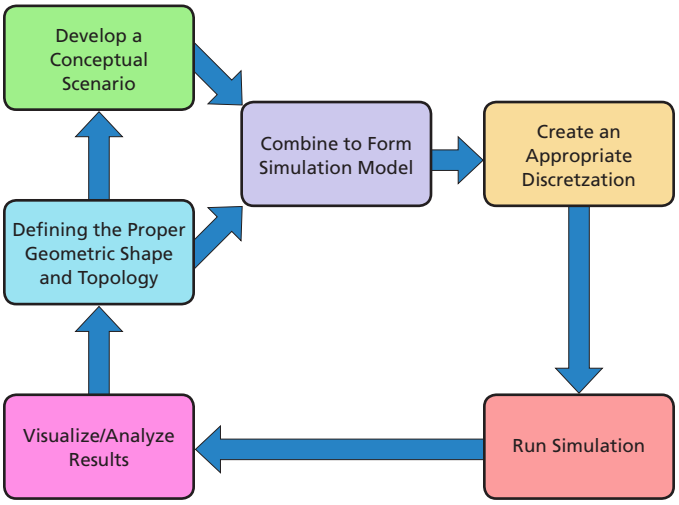
Cloud Computing and Visualization Tools for KBase

Dr. Jeffrey Baumes will lead the work focused on the DOE's Systems Biology Knowledgebase (Kbase), a community-driven cyberinfrastructure for sharing and integrating biology and genetics data. Kitware will collaborate with The Ohio State University on this project, which will specifically address the critical need of extracting meaningful knowledge from ever-growing volumes of scientific data by making improvements to Kbase. Three distinct improvements will be made to Kbase: the design and incorporation of a new cloud-based architecture for genetics algorithms, integration of new visualization tools, and the ability to link algorithms to existing databases.

A fully functional, visualization-enabled Kbase system could be adopted by genetics and systems biology laboratories at academic, government, and industrial research centers due to its customizable, open-source nature. Organizations can use the system to deploy their own components and efficiently produce, share, and run their algorithms from the cloud, providing easy collaboration with researchers from both within and outside of the agency.

Graphical HPC Application Suite for Supporting the Product Simulation Lifecycle

Bob O'Bara will lead the HPC application suite project, which will focus on enabling widespread adoption of high performance computing (HPC) functionality for small and medium-sized manufacturing firms by providing a suite of open-source applications that address the complete simulation lifecycle. The project will address current hurdles of technology integration, interaction, and scalability. Many of the current tools used to address parts of the simulation lifecycle are rigid, expensive, and cumbersome for non-simulation experts; Kitware's open-source solution will make it more realistic for organizations of all sizes and budgets to leverage HPC resources.



Simulation lifecycle using conceptual scenarios

The project will address these issues and streamline manufacturing workflows by developing a suite of open-source applications based on a flexible framework and built upon existing open-source HPC toolkits such as CGM, OpenCASCADE, MOAB, Meshkit, and ParaView. This project will specifically target the field of computational fluid dynamics; however, the applications are interchangeable, customizable, and can be easily modified to address various vertical markets. A consistent, intuitive graphical user interface will be used in the different applications in the suite, making it easy to use by experts and non-experts alike.

KITWARE COLLABORATES ON SDAV INSTITUTE

Kitware is a collaborating team member in the Scalable Data Management, Analysis, and Visualization (SDAV) Institute, supported by the U.S. Department of Energy's Scientific Discovery through Advanced Computing (SciDAC) program. The project, led by Lawrence Berkeley National Lab, is a collaboration between prominent government laboratory, university, and industry partners for the purpose of creating a comprehensive set of tools and techniques that are essential for knowledge discovery on the DOE's various computing platforms over the next five years.

The SDAV Institute's primary goals are to actively work with application teams and assist them in achieving breakthrough science; and to provide technical solutions in data management, analysis, and visualization, all of which are broadly used by the computational science community. Such expertise is required to address the ever-increasing size, scale, complexity, and richness of scientific data.

Current users of DOE computing systems are faced with managing and analyzing their datasets for knowledge discovery, and must often rely on antiquated tools more appropriate for the teraflop era. While new techniques for interpreting large data have been developed, many in the field believe that these new algorithms and implementations will not work on their systems as we move into the many-core era. Without addressing these issues and beliefs, there will be a widening gap between computational and I/O capacity.

The SDAV Institute is working with DOE researchers to develop a comprehensive approach that will address these issues by encompassing all the stages of data analysis from initial data generation, orchestration of analysis tasks, and effective visualization of the results.

Kitware is supporting the DOE's scientific teams with large-scale data analysis and visualization. This involves making enhancements and extensions to current tools, such as ParaView and VisIt; introducing and supporting new technologies leveraging many-core and multi-core architectures; and coupling data analysis capability with simulation codes for in-situ analysis and co-processing.

TOM MUNNECKE VISITS KITWARE

In April, Kitware hosted Tom Munnecke, one of the world's leading experts in health information technology and an original member of the VistA team, who gave a presentation titled "Towards a Federal Health Information Space." He discussed his observations and lessons learned as one of the original architects of VistA and CHCS, and focused on the need for an open, adaptable health information space. To this end, he presented an architectural conceptual model based on an "Information Space" rather than a "Integrated System" that approaches the federal health information as a large-scale, fine-grained network of information and interactions. His recorded presentation is available on Kitware's webinar page, www.kitware.com/products/webinars.html.

KITWARE PARTICIPATES IN THE UNIVERSITY AT ALBANY OPEN SOURCE FESTIVAL

Will Schroeder, Bill Hoffman, and Luis Ibáñez presented at the University at Albany Open Source Festival in March. The event, organized by the UAlbany Student Chapter of the American Society for Information Science & Technology (ASIS&T), brought together many groups invested in open source, along with others interested in learning more about the Way of the Source.

Will gave a presentation titled "Open Source Business Models," during which he spoke about the variety of business models that have been used to monetize open-source software and communities. Will discussed a variety of business approaches ranging from open source as a hobby to consulting, dual licensing, and collaborative R&D, which is a major component of the Kitware model.

Bill presented on "Open Source Quality Processes" and discussed how open-source practices result in higher-quality software due to the many eyes theory. Although not all open-source projects will benefit from this theory as much as the Linux kernel, there are open tools and processes that enable developers to share and test software regardless of community size. Tools including Git, CMake, CTest, and CDash are examples of freely-available tools for managing, building, and testing high-quality software – while also reducing maintenance costs and maximizing community involvement.

Lastly, Luis Ibáñez gave two presentations, one on "Open Source in Healthcare" and another focused on "Open Source in Education". In speaking on open source in healthcare, Luis introduced the economic relevance of healthcare costs in the U.S. He described how the open-source community can contribute to reducing the cost of healthcare while simultaneously increasing its quality by participating on the open-source initiatives surrounding the development of electronic health record (EHR) systems, such as the Veterans Health Information Systems and Technology Architecture (VistA) and the open-source EHR Agent (OSEHRA).

In the "Open Source in Education" talk, Luis presented on how Kitware has partnered with Rensselaer Polytechnic Institute (RPI) since 2007 to offer an Open Source Software Practices class. In this class, students are exposed to basic principles of economics; models of peer-production; social principles of collaboration; the legal basis of copyrights, patents, and trademarks; principles of software licensing; and the motivational aspects of cooperation. Simultaneously, students are able to practice interaction skills with open-source communities through use of revision control systems, mailing lists, forums, wikis, code peer-review systems, software quality control, and documentation.

KITWARE TO OPEN NEW MEXICO OFFICE

In mid-June, Kitware announced the opening of our third U.S.-based office in Santa Fe, New Mexico. The office will be headed by Patrick O'Leary, a specialist in high performance computing, who will be leading the office as Assistant Director of Scientific Computing. Before joining Kitware, Dr. O'Leary was the Director of the Center for Advanced Modeling and Simulation at Idaho National Laboratory; as well as Director of Operations of WestGrid, one of the seven partner consortia that make up Compute Canada.

Similar to Kitware offices in Clifton Park, NY; Carrboro, NC; and Lyon, France, the new location will provide support and services for scientific computing, particularly focused on high performance computing and visualization.

"I could not be more excited about this venture," said Dr. O'Leary. "New Mexico is a great location to focus on HPC and visualization as the National Labs there are leading the field. With this expansion, Kitware will now be able to provide more direct support and consulting to these organizations."

"Kitware recognizes the increasing importance of computational science, analytics, and large data, and the necessity of growing our HPC capabilities," said Will Schroeder. "We are thrilled to have Dr. O'Leary on our team and are looking forward to exploring new collaboration opportunities that help us impact and improve scientific research."

ISBI 2012: ITK-FOCUSED HACKATHONS

Kitware attended the International Symposium on Biomedical Imaging (ISBI) and participated in two ITK-related hackathons: integrating the NiftyReg library from University of College of London into ITK and integrating more ITK functionalities into Osirix. Over the two days of hackathons, Luis Ibáñez, Xiaoxiao Liu, and Matt McCormick met with several group to discuss ITK, new ideas, and potential collaborations.

The NiftyReg hackathon focused on integrating the NiftyReg tool as an ITK module. NiftyReg is a package that performs Image Registration by taking advantage of GPUs. The NA-MIC / Slicer community has been interested in using this library; by including it as an external ITK module, it will be easier for the NA-MIC and ITK communities in general to get take advantage pf NiftyReg. The proposed course of action for the ITK team includes assisting with the configuration of an external module and testing.

In the Osirix hackathon, Kitwareans worked with Jan Perhac from the Osirix team in Geneva and two researchers from the Polytechnic University of Catalunya (UPC): Raul Benitez and Oriol Carrillo. The hackathon focued on building an example ITK plugin for Osirix and discussing using ITK pipelines in Osirix for Image Guided Surgery (IGS) applications. The Osirix team explained that the plugin infrastructure is in flux, but that once it is settled, there will be abundant opportunities for extending Osirix via ITK, IGSTK, and TubeTK plugins.

KITWARE ANNOUNCES SPRING PROMOTIONS

This spring, Kitware announced several promotions of its team members. Theresa Vincent is now an Accounting Administrator in recognition of her growing authority and responsibility in driving the Accounts Payable operations. There are two new Technical Experts, Brad King and Roddy Collins, recognizing them as individuals who have made and continue to make sustained and broad technical impacts across Kitware. Kitware also promoted two new Technical Leaders, Patrick Reynolds and Marcus Hanwell, acknowledging their contributions to building cross-cutting technology and developing a new initiative in computational chemsitry, respectively. Lastly, Anthony Hoogs was promoted to Senior Director of Computer Vision for his significant contributions to Kitware, and Stephen Aylward was promoted to Senior Director of Operations - North Carolina for his work in growing and managing the North Carolina office.

Congratulations to the Kitwareans on their achievements!

UPCOMING CONFERENCES AND EVENTS

OSCON 2012

July 16-20 in Portland, OR. Kitware will give two presentations at the event, "Mobile 3D Visualization" and "OSEHRA - Building an Open Source EHR for All."

SciPy

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

SIGGRAPH 2012

August 5-9 in Los Angeles, CA.

American Chemical Society Meeting

August 19-23 in Philadelphia, PA. Marcus Hanwell and Kyle Lutz will present two talks, "Avogadro, Open Chemistry and Chemical Semantics" and "Exploring Large Chemical Data Sets: Interactive Analysis and Visualization."

BioImage Informatics

September 16-19 in Dresden, Germany. Luis Ibáñez , an invited speaker, will present a talk titled "Reproducible Image Analysis for Research and Education."

MICCAI 2012

October 1-5 in Nice, France. Kitware will sponsor the "Young Scientist Publication Impact" award.

SIAM Conference on Mathematics for Industry

October 2 in Denver, CO. Will Schroeder is a plenary speaker and will present a talk focused on the value of leading-edge visualization for industrial research and development.

NEW EMPLOYEES

David Lonie

David Lonie joined Kitware as an R&D Engineer on the scientific computing team at the Clifton Park office. David is completing his Ph.D. in chemistry at the State University of New York at Buffalo, where his thesis focuses on the evolutionary algorithm prediction of novel crystalline structures.

Patrick O'Leary

Patrick O'Leary joined Kitware as the Assistant Director of Scientific Computing and leader of Kitware's newest office in Santa Fe, New Mexico. Dr. O'Leary's expertise includes large-scale scientific computing, visualization, and modeling; high performance computing, and leadership.

Kevin Zimmerman

Kevin Zimmerman joined Kitware as a Systems Administrator at the Clifton Park office, bringing more than 17 years of experience to the team. Kevin earned his M.S. degree in management with an MIS concentration from Rensselaer Polytechnic Institute in Troy, NY.

Ricardo Ortiz

Ricardo Ortiz joined the medical team as an R&D Engineer at the Carrboro office. He earned his Ph.D. in applied mathematics and computational sciences from the University of Iowa. Ricardo is also a Postdoctoral Fellow Associate in the mathematics department at the University of North Carolina at Chapel Hill.

KITWARE INTERNSHIPS

Kitware is pleased to welcome ten new and returning interns this summer. The five interns at the Clifton Park office are assisting the computer vision and medical teams and include Ilseo Kim, Tianyang Ma, Sean Tozier, Syed Zain Masood, and Dan Gnoutcheff.

At the Carrboro office, Vikas Shivaprabhu, Christopher Mullins, and Nathan Taylor are assisting the computer vision and medical teams. Célia Pansard and Guillaume Sala are long-term interns and will be with Kitware for the next year.

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.

Our interns assist in developing foundational research and leading-edge technology across six business areas: supercomputing visualization, computer vision, medical imaging, data management, informatics and quality software process. We offer our interns a challenging work environment and the opportunity to attend advanced software training. To apply send, your resume to internships@kitware.com.

EMPLOYMENT OPPORTUNITIES

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

At Kitware, you will work on cutting-edge research alongside experts in the field, and 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; yearly bonus; and free drinks and snacks. For more details, visit our employment site at jobs.kitware.com

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

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

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

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

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

Contributors: Lisa Avila, Stephen Aylward, Aashish Chaudhary, David Cole, David Demarle, Andinet Enquobahrie, Jean-Christophe Fillion-Robin, Özgür Güler, Luis Ibáñez, Tara Lindsley, Yuanxin Leo Liu, Philippe Pebay, Patrick Reynolds, and Ziv Yaniv .

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

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