Application Creation for an Immersive Virtual Measurement and Analysis Laboratory

Wesley N. Griffin* William L. George* Terence J. Griffin* John G. Hagedorn* Marc Olano*
Steven G. Satterfield* James S. Sims* Judith E. Terrill*

Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, US

ABSTRACT

Content creation for realtime interactive systems is a difficult problem. In game development, content creation pipelines are a major portion of the code base and content creation is a major portion of the budget. In research environments, the choice of rendering and simulation systems is frequently driven by the need for easy to use content authoring tools. In visualization, this problem is compounded by the widely varying types of data that users desire to visualize. We present a visualization application creation framework incorporated into our visualization system that enables measurement and quantitative analysis tasks in both desktop and immersive environments on diverse input data sets.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; I.3.4 [Computer Graphics]: Graphics Utilities—Application packages

1 INTRODUCTION

A key part of laboratory science is measurement: taking measurements of data and ensuring those measurements are repeatable. Metrology is the science of measurement and has traditionally meant measurements on laboratory data that are made during the course of a physical experiment. Computational Science is the growing field of applying large-scale numerical simulation codes to develop and expand scientific understanding. With computational science, a mathematical model is simulated on a computer and, while physical experiments are used to validate the model and simulation, there is usually no corresponding physical experiment to the running simulation. This creates the issue of how to perform measurement and analysis on the simulation data.

One possible output from the measurement and analysis process are reference materials which can be used to calibrate or assess other measurement systems or processes [11]. Reference materials can be certified which include certificates attesting to the validity and traceability of the metrology process for creating the reference material. Reference materials are important for national standards organizations in helping industrial companies ensure machines are calibrated.

We have developed a virtual measurement and analysis laboratory in a 3D immersive virtual environment using a CAVE hardware configuration and a software visualization system [4, 7, 13]. One of the challenging aspects during the development and deployment of this system has been managing the widely disparate types of data that scientists want to measure and analyze. This "content creation problem" is not unique to our system. In general, content creation for realtime interactive systems is a difficult problem.

In the game development industry, content creation and "asset pipelines" are a major portion of the development team, time, and budget. The software developed for content creation often has very limited reusability. In research environments, there is often no budget for content creation (be it time or money) and thus the ability to quickly create content often forces researchers to use commercial-off-the-shelf (COTS) rendering systems that provide easy content creation tools. However, being forced to use COTS rendering systems can limit the flexibility of the realtime interactive system.

We present our High End Visualization (HEV) system which is a flexible framework for visualization application creation. Our system design provides two main abstraction points: renderer display output and run-time renderer control. Abstracting renderer display output enables our system to run a single binary on both desktop workstations and immersive environments without change to the visualization application. Abstracting run-time renderer control enables our system to provide a large set of small, flexible tools that can be composed with input data to easily develop visualization applications.

Section 2 introduces our visualization system software architecture. A key feature of our system is the use of a named pipe and text-based control commands for run-time modification of the in-memory scenegraph. This named pipe and set of control commands combine with a large set of application-generic scripts and tools to create our flexible content creation framework with fast iteration for visualization application development. Section 3 describes our content creation framework in detail. Section 4 describes a few of the visualization applications that have been developed with HEV and are currently in use.

2 SYSTEM ARCHITECTURE

Our CAVE consists of three display surfaces, 3D stereoscopic projectors with synchronized glasses, 6 DOF tracking for the user's head, a 6 DOF tracked wand for user interaction, and two channel audio. Two of the display surfaces are rear-projection screens mounted vertically at 90 degrees to each other. The third surface is a front-projection floor suitable for walking or standing. These surfaces are configured as a corner as shown in Figure 1. The corner configuration has proved very effective over the years. It is compact, minimizing floor space requirements, yet provides a comfortable space for collaborative working sessions and demonstrations.

Our visualization system is called HEV (High End Visualization) and consists of a stack of vendor supplied, open source, and in-house software pieces. Figure 2 shows how these pieces fit together. Currently we use OpenSceneGraph (OSG) for rendering and have built two layers on top of OSG: Interpreted Runtime Immersive Scenegraph (IRIS) and IRIS Development Environment for Applications (IDEA) which we describe next.

2.1 Interpreted Runtime Immersive Scenegraph

On top of OSG we have developed the Interpreted Runtime Immersive Scenegraph (IRIS) which controls the hardware elements of our CAVE. IRIS creates the rendering contexts to drive the CAVE displays and also integrates the hardware tracker data to build the user view and update the projected images. The IRIS command `iris-viewer` is the main program that initializes the CAVE and

*e-mail: {firstname.lastname}@nist.gov
IEEE 9th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS) 2016
20 March, Greenville, SC, USA
978-1-5090-4275-3/16/$31.00 ©2016 IEEE

Figure 1: The CAVE corner configuration (left) has proved very effective over the years. It is compact, minimizing floor space requirements, yet provides a sufficient space for collaborative sessions and demonstrations and the wide field of view provides good immersion. Using the CAVE (right) to analyze Newtonian and Non-Newtonian flows in a pipe. See Section 4 for discussion of this specific visualization application.

performs all of the rendering. IRIS has two key features to enable rapid and flexible visualization application development: Display Object and Control Pipe.

Display Object IRIS uses dynamic shared objects loaded at runtime to configure the rendering contexts and provide the graphics windows for rendering. By separating the display devices into dynamic shared objects, IRIS can execute the same binary on both the CAVE hardware and development workstations which have standard 2D monitors. This enables visualization application developers to sit at their desks and develop applications on top of HEV that can then run in the CAVE with no changes needed. For testing and debugging, IRIS provides a simulator mode that simulates the CAVE hardware on a workstation with multiple graphics windows.

Control Pipe IRIS also listens on a named pipe for text-based control commands. These commands can modify the in-memory scenegraph thus enabling separate processes to dynamically update the rendered visualization. Since the control pipe can be written to by any other process or script, the majority of our visualization applications make heavy use of control commands. Using scripts to modify the scenegraph at run-time provides fast iteration on visualization application development and content creation.

2.2 IRIS Development Environment for Applications

On top of IRIS, HEV provides the IRIS Development Environment for Applications (IDEA). This is an in-house collection of commands, scripts, and file formats. IDEA is developed using the traditional UNIX philosophy of small tools focused on one simple task that is useful for a wide variety of applications. A nice feature of the IRIS Control Pipe is that IDEA can leverage standard software packages. IDEA uses X11 and a standard window toolkit for providing the visualization application graphical user interface (GUI). The 2D GUI windows are overlaid on top of the 3D rendering window and thus easy to manipulate in the CAVE. Figure 3 shows how the wand controls GUI menus.

As an example, the IDEA tool `irisfly` is a shell script that wraps and extends the IRIS command `iris-viewer`. `irisfly` is a general tool for displaying a wide variety of data with all of the capabilities of the CAVE. As needed, an application-specific file loaded can be created and added to `irisfly` and `irisfly` can be easily combined with other HEV tools to assemble specific visualization applications.

Another example of a widely used generic IDEA tool is the IDEA clipping controls. IRIS configures eight clipping planes in the scenegraph with no specific geometry that are disabled by default. The clipping control tool implements several different clipping geometries, such as a windshield, corner, or box, and dynamically controls the clipping planes using the Control Pipe.

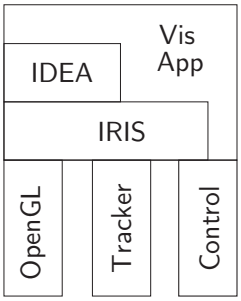

Figure 2: Software stack.

Figure 3: Wand control of GUI menus.

2.3 Visualization Applications

The Visualization Applications sits atop the software stack and are frequently assembled by combining tools from IDEA and issuing commands to the Control Pipe. All current applications are based on `irisfly` and implemented without modification or recompiling the core iris-viewer code. Since separate processes can write to the Control Pipe, visualization applications can leverage additional software packages such as the R statistical analysis tool, or the D3.js information visualization tool. Section 3 discusses in more detail how IDEA and the IRIS Control Pipe combine to create a flexible content creation pipeline with fast iteration. Section 4 describes a few of the visualization applications that have been developed with HEV and are currently in use.

3 Visualization Application Creation

In HEV, a new visualization application typically starts as a shell script that processes the input data files. For input data types that have been used in the past, there is likely an IDEA tool for processing the files. If the input data has not been used with HEV before, our process is to develop any application-specific tools for processing the data and then abstract those tools into general tools for future use.

```
FUNC RgbaLut1D myTF2
    VAL_TRANSFORM log10
    RGBLUT
      1.2e-20   0.4 0.1 0.7
      3.9e-15   0.2 0.2 0.6
      6.8e-07   0.1 0.6 0.6
      5.9e-02   0.1 0.6 0.2
      1.0e+01   0.1 0.7 0.1
    END_RGBLUT
    ALPHALUT
      1.2e-20   0.0
      3.9e-15   0.4
      6.8e-07   0.9
      5.9e-02   1.0
      1.0e+01   1.0
    END_ALPHALUT
END_FUNC myTF2
```

Figure 4: Example of the transfer function description. The volume data is assumed to have a single scalar value at each voxel location.

3.1 Input Data Processing

Input data processing results in geometry files for rendering. The geometry files are frequently saved as OSG binary files which consist of the OSG scenegraph bits to render the geometry. Occasionally, other geometry files are used and HEV supports any geometry file that OSG can read (e.g. Inventor or OBJ). HEV also supports a custom text-based format, SAVG, that is a simple description language for geometry. The SAVG file format is a historical artifact of the evolution of our system but is still useful as HEV provides a large suite of command-line tools for creating geometry in the SAVG format and modifying SAVG files.

For very common cases, such as volume visualization of spatial volume datasets (e.g. microtomography), there is an IDEA tool that can generate a volume visualization application for the input data. This tool `vol-visBuilder` takes the volume data files and a transfer function description and outputs a set of geometry files, shader sources, textures, and shell scripts which can be immediately viewed. Figure 4 shows an example of the `RgbaLut1D` transfer function description where the volume data set is assumed to have a single scalar value at each voxel. The `VAL_TRANSFORM` keyword specifies a value transform, the `RGBLUT` and `ALPHALUT` tables define the RGB and alpha lookup up tables respectively.

3.2 HEV Tools

After input data file processing, the geometry files can be immediately viewed using `irisfly`. However, a visualization application needs additional functionality to be useful and HEV provides a large set of generic tools for adding this functionality. These tools range from a simple "gnomon" orientation tool, to a "light-editor" that can modify the pre-set lights in the scenegraph, to a geometry clipping control (described above), and a few different animation controllers that have evolved depending on the type of "animation" being used.

These tools all share several common features: 1) they have command-line interfaces, 2) they are small in scope and focus on a single feature, and 3) they communicate by sending their output to the console which is then redirected to the Control Pipe. These three features enable fast iteration in the visualization application development process by providing flexible and composable visualization control, measurement, and analysis tools. The command-line interface enables the tools to be run while the visualization application is under development and also run from shell scripts for additional flexibility. The small scope of each tool ensures that each tool remains simple to implement. Console output allows the tools to interactively modify the in-memory scenegraph through redirection to the Control Pipe.

```
LOAD wirebox wirebox.savg
ADDCHILD wirebox world
EXEC hev-gnomon > $IRIS_CONTROL_FIFO
BACKGROUND 1 1 1
```

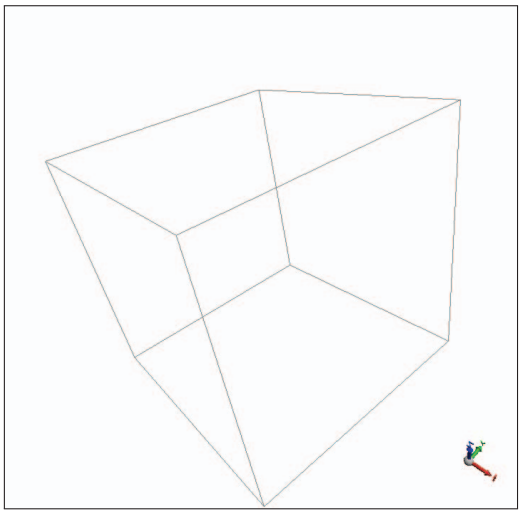

Figure 5: Example of an IRIS file and the associated rendered view.

The development process continues by deciding which of these tools are most useful for the specific visualization application and incorporating them. If a visualization requires new functionality, a new tool can be developed and if it has the above three features, then it will integrate well with the existing HEV tools. As the visualization application is expanded an IRIS file is created that encompasses loading the geometry, running any tools, and adding user interface elements. IRIS files consist of the same text-based control commands for the Control Pipe interpreted by `iris-viewer`. IRIS files are the key component of our application creation framework.

Figure 5 is a basic IRIS file and corresponding rendered view that `LOAD`s a wireframe cube into a node named *wirebox*, adds the node to *world*, `EXEC`s the `hev-gnomon` tool to provide an orientation gnomon in the scene, and sets the `BACKGROUND` of the render window to white. The `$IRIS_CONTROL_FIFO` is a environment variable that specifies the location of the Control Pipe.

3.3 User Interface

As previously mentioned, IDEA uses standard X11 windows for the visualization application GUI. A useful testing utility, `hev-wiggleNav` is a simple GUI tool which causes the virtual environment to rock back and forth, providing enhanced depth cues in desktop windows by means of motion parallax. The tool, when run, displays a window as shown in Figure 7. While running the tool directly is useful for development, a better user experience is provided if the wiggle nav UI can be shown or hidden by the user when necessary. To enable this, the Master Control Panel (MCP) seen in Figure 6 can be extended with additional buttons and menus via MCP files, which are text-based descriptions of UI elements. Figure 8 shows an example MCP file that adds the *wiggle* button to the MCP in Figure 6. When clicked, this button toggles the wiggle nav UI.

There is also a query control command for querying the scenegraph. These query commands allow for more complex tools to interact with IRIS over the Control Pipe. One example that uses the query interface is an interactive "light editor" that enables run-time

Figure 6: The Master Control Panel (Section 3.3).

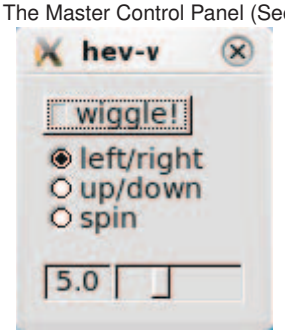

Figure 7: The wiggle nav UI (Section 3.3).

```
BUTTON wiggle
  FIRST AFTER REALIZE EXEC hev-wiggleNav \
      -title wiggle > $IRIS_CONTROL_FIFO

  FIRST WAIT irisfly-addAndShowWindow wiggle
  ON WAIT irisfly-addAndShowWindow wiggle
  OFF WAIT irisfly-removeAndHideWindow wiggle
```

Figure 8: The wiggle MCP file (Section 3.3).

editing (adding, removing, orienting, etc.) of the lights in the scene. Figure 9 shows the user interface of this tool.

We have also implemented a server running in Node.js that listens on an HTTP WebSocket and forwards control commands to the control pipe as well as query replies back across the WebSocket. This server has enabled us to develop web-based visualization applications leveraging D3.js running in a web-browser and interacting with the data being rendered in the 3D immersive environment. This can be seen in Figures 14 and 15.

Specifically, in Figure 14, the yellow arrow is the HEV probe tool which updates a shared memory object with the 3D location of the probe. Figure 10 shows the complex MCP file for the probe. In this listing, the `FIRST` commands create the state for the probe when the button is first selected:

1. `probeRun.sh` creates a 2D message box that is updated with the current probe position in 3D space.
2. `irisfly-select` updates a share memory "selector" for other processes to determine when the probe is active.
3. `pointerGlyph.iris` and `plus3d.osg` are the geometries for the probe.
4. the `hev-shmOnOff` commands execute actions for the left and right wand buttons.
5. `irisfly-addAndShowWindow` ensures the probe windows are visible.

The `ON` commands enable all processes on the `probe` selector, enable the geometry node, and show the probe windows. The `OFF` commands reverse the `ON` commands.

This section has only covered a few of the tools available in HEV. As previously mentioned, there is a clipping control tool that supports several different types of clipping geometries and also animation control tools that support a few different types of animations for time-varying datasets. All combined, we have implemented over

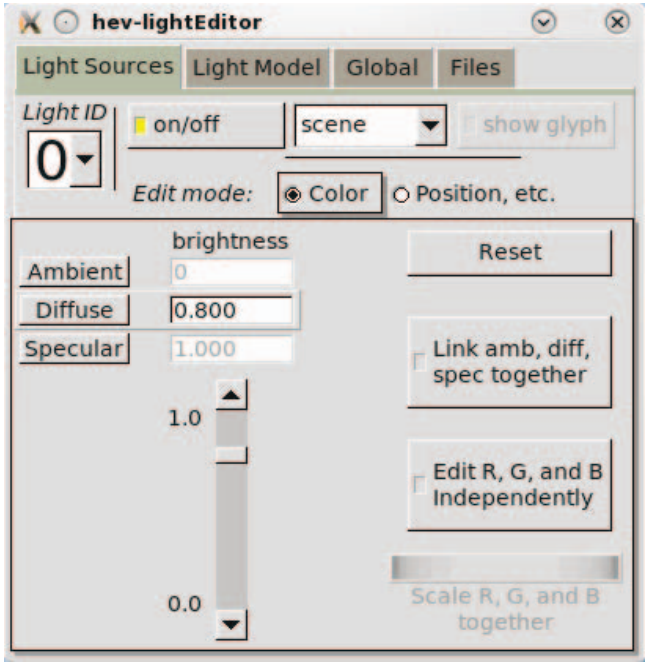

Figure 9: The light editor user interface. This tool enables run-time editing (adding, removing, orienting, etc.) of the lights in the scene.

```
BUTTON probe
  FIRST EXEC $HEV_IDEA_DIR/etc/hev-probe/bin/probeRun.sh
  FIRST WAIT irisfly-select probe

  FIRST LOAD \
    ${HEV_IDEA_DIR}/etc/hev-probe/data/pointerGlyph.iris
  FIRST LOAD irisflyPlus3d plus3d.osg
  FIRST NOCLIP irisflyPlus3d

  # what to do when the wand buttons are pressed
  FIRST EXEC hev-shmOnOff --selector probe \
    idea/buttons/left \
    < $HEV_IDEA_DIR/etc/hev-probe/data/eventLeft.onOff \
    > $IRIS_CONTROL_FIFO
  FIRST EXEC hev-shmOnOff --selector probe \
    idea/buttons/right \
    < $HEV_IDEA_DIR/etc/hev-probe/data/eventRight.onOff \
    > $IRIS_CONTROL_FIFO

  FIRST WAIT irisfly-addAndShowWindow probePosition

  # make the probe visible and turn on button routing,
  # make message windows visible
  ON WAIT irisfly-select probe
  ON NODEMASK irisflyPointerGlyph ON
  ON WAIT irisfly-addAndShowWindow probePosition
  ON WAIT irisfly-addAndShowWindow probe@

  # make the probe and its message windows invisible
  # and turn off button routing
  OFF NODEMASK irisflyPointerGlyph OFF
  OFF WAIT irisfly-removeAndHideWindow probePosition
  OFF WAIT irisfly-removeAndHideWindow probe@
  OFF WAIT irisfly-deselect
```

Figure 10: The probe MCP file discussed in Section 3.3.

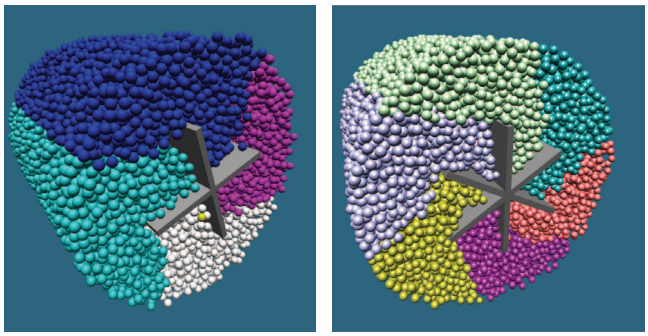

Figure 11: Simulation snapshots of a proposed mortar standard reference material in a 4-blade (left) and a 6-blade (right) rheometer. The spheres are color-coded based on their starting position relative to each vane blade.

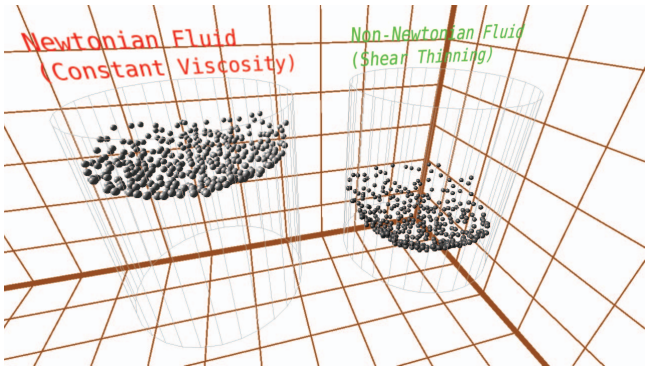

Figure 12: Side-by-side comparison of Newtonian and Non-Newtonian fluids flowing through a pipe. One layer is isolated for visual speed comparison during animation of the simulation data.

250 different tools over the lifetime of HEV. By ensuring each tool is small and focused on a single task and sends its output to the console, these tools are easily composed into larger visualization applications. The next section presents some of these applications that we have developed.

4 Visualization Applications

HEV is currently in active use and visualization applications have been developed over the years. Each application involves at least one domain scientist and the visualization is tailored to the specific scientific data. The applications focus on providing quantitative measurement and analysis of the data.

4.1 Rheology of Dense Suspensions

Understanding the mechanisms of dispersion or agglomeration of particulate matter in complex fluids, such as suspensions, is important in many industries such as pharmaceuticals, coatings, and construction. These fluids are disordered systems consisting of a variety of components with disparate properties that can interact in many different ways. Modeling and predicting the flow of such systems requires large-scale simulations. A large-scale simulation tool [8, 9, 10] is used in the development of standard reference materials (SRM) for the calibration of vane rheometers used to measure the flow properties of fresh concrete. Figure 11 shows snapshots of the rheometer simulations for a proposed mortar SRM.

A second flow of interest to industry is the flow of dense suspensions through a pipe. Flow of suspensions in pipe or channel systems is important for a wide variety of applications including

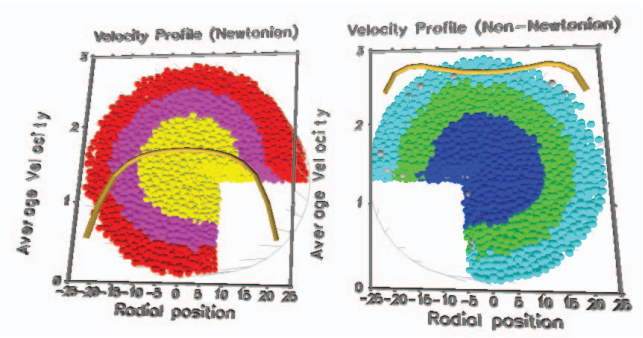

Figure 13: Velocity profile plots show velocity of the solids near the pipe wall is much greater for Non-Newtonian fluids.

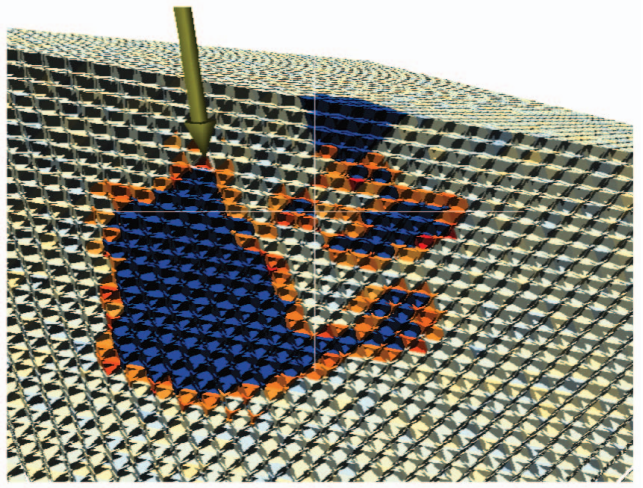

Figure 14: Immersive display of the growth of C3S particles (rendered in blue) hydrating in a calcium hydroxide solution. Regions of orange are where hydration reactions have produced solid calcium hydrate reaction product.

pumping of concrete and slurries, micro-fluidic devices, and biological systems. A visualization application was developed to compare the results from two pipe flow simulations. One data set was for a Newtonian fluid and the other for a Non-Newtonian fluid. Figure 12 shows the side-by-side comparison of the two pipe flow simulations. Controls allow playing the simulation forward and backward. A single layer can be isolated to better compare the velocity of the two data sets. For this visualization application, an analysis tool was developed to show the average velocity of each flow as a line graph. These average velocity graphs are displayed in a heads-up fashion and are dynamically updated corresponding to both the simulation time step and layer position. Figure 13 illustrates this new analysis tool.

4.2 Cement Paste Hydration and Microstructure Development

When cement powder is mixed with water, a hydration process occurs that transforms the cement-water paste from a fluid suspension into a hardened solid. This process involves complex chemical and micro-structural changes. Understanding and predicting the rates of these changes is a longstanding goal. Computational modeling of the hydration of cement is challenging because it involves a large number of coupled non-linear rate equations that must be solved in a highly irregular 3D spatial domain. HydratiCA is a stochas-

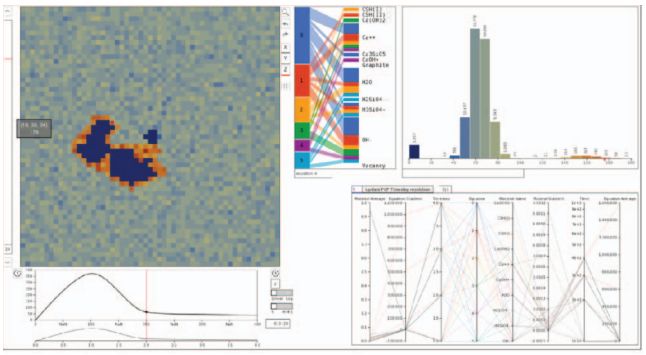

Figure 15: Quantitative graphs of the simulation output corresponding to the 3D visualization in Figure 14. The graphs are dynamically updated as the timestep is changed in the immersive environment.

tic reaction-diffusion cement hydration computational model that addresses these challenges [1, 2, 3, 6].

Figure 14 shows the growth of C3S particles (rendered in blue) hydrating in a calcium hydroxide solution. Regions of orange are where hydration reactions have produced solid calcium hydrate reaction product. This figure is the 3D immersive display of the simulation tool output. For this visualization application, a hybrid tool was developed [5] that provides a 3D immersive view and 2D quantitative view of the same dataset. The 2D quantitative view is implemented in D3 and runs in a web browser. Figure 15 shows the quantitative graphs. Key to this hybrid visualization application is that the immersive 3D display and 2D quantitative display can communicate through the Control Pipe. This enables two-way probing of the data to be dynamically reflected in both displays.

4.3 Body Area Networks

With recent advances in microelectronics, the technology to build very small and extremely low power wearable and implantable devices is clearly within our reach. However, commercial success of this technology depends on the widespread adoption of a global standard for its communication protocol. Knowledge of radio frequency (RF) propagation is a critical step in this process, enabling RF engineers to optimize their physical layer design to achieve better communication performance. Such information is typically gathered by conducting physical experiments and processing the measured data to obtain propagation channel characteristics. Obtaining sufficient data to study various scenarios is difficult for wearable body area network sensors. Moreover, obtaining data through physical experimentation is extremely challenging or impossible for implantable devices. Therefore, a 3D modeling and visualization platform that is capable of emulating physical experiments is required in understanding RF propagation in body area networks [14, 12].

The modeling tool enables researchers to place a custom designed antenna at the desired location of the human body, set the operating frequency of the node, and simulate the RF propagation in and around the human body. The 3D immersive platform, shown in Figure 16 then visualizes the propagation data as a heatmap of signal loss. Several IDEA tools are used to enable probing of the data such as dynamically-updated point values based on wand position or a line-segment tool that uses R to generate summary plots of the data along the line-segment.

5 Conclusion

Content creation is difficult for rendering systems and visualization compounds the problem by needing to visualize widely disparate types of data. In research settings, easy content creation frequently

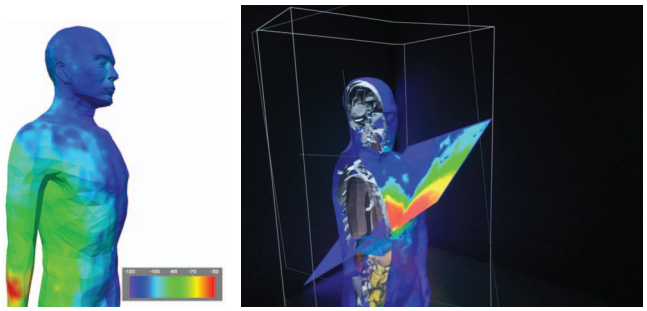

Figure 16: Heatmap of RF propagation signal loss. Propagation on the body (left) and on a plane inside the data volume (right). The right figure also uses clipping planes to show the inside of the body.

drives the choice of rendering and simulation systems. We have developed the High End Visualization (HEV) system that incorporates a visualization application creation framework built on top of a control command interface that can dynamically update an in-memory scenegraph. Small, focused tools that send output to the console are composed using the Control Pipe and IRIS files. This framework has enabled rapid and flexible visualization application development that uses a single rendering executable to run on both a CAVE system and desktop workstation.

We have been migrating our code base to git and will be publishing the software on Github in the near future. We are also evaluating replacing the rendering backend to leverage the new low-overhead graphics APIs such as Vulkan. Before replacing the rendering backend, however, we will implement rendering 'unit tests' to verify that changes made to the rendering system do not have adverse consequences on the rendered output.

Acknowledgements

We would like to thank our scientific collaborators on these projects: Jeffrey W. Bullard, Nicos S. Martys, and Kamran Sayrafian-Pour as well as all of the other NIST staff that have worked with us on these projects. We would also like to thank the reviewers who helped improve this paper with their comments.

DISCLAIMER Certain commercial products are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.

References

[1] J. W. Bullard, E. Enjolras, W. L. George, S. G. Satterfield, and J. E. Terrill. A parallel reaction-transport model applied to cement hydration and microstructure development. *Modelling and Simulation in Materials Science and Engineering*, 18(2):025007:1–16, 2010.

[2] J. W. Bullard, E. J. Garboczi, W. L. George, N. S. Martys, S. G. Satterfield, and J. E. Terrill. Advancing the materials science of concrete with supercomputers. *Concrete International*, 33(1):24–29, Jan. 2011.

[3] J. W. Bullard, T. Ley, J. G. Hagedorn, R. Desaymons, W. N. Griffin, J. E. Terrill, Q. Hu, S. G. Satterfield, and P. Gough. Direct comparisons of 3d hydration experiments and simulations. In *6th Advances in Cement-Based Materials Conference*, 2015.

[4] J. E. Devaney, S. G. Satterfield, J. G. Hagedorn, J. T. Kelso, A. P. Peskin, W. L. George, T. J. Griffin, H. K. Hung, and R. Kriz. Science at the speed of thought. In Y. Cai, editor, *Ambient Intelligence for Scientific Discovery*, volume 3345 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2005.

[5] W. N. Griffin, D. Catacora, S. G. Satterfield, J. W. Bullard, and J. E. Terrill. Incorporating d3.js information visualization into immersive

virtual environments. In *Electronic Proceedings of IEEE Virtual Reality (VR) 2015*, 2015.

[6] J. G. Hagedorn, J. W. Bullard, R. Desaymons, , W. N. Griffin, J. E. Terrill, T. Ley, Q. Hu, S. G. Satterfield, and P. Gough. A parallelized numeric model of cement hydration. In *XSEDE 2015*, 2015.

[7] J. G. Hagedorn, J. Dunkers, S. G. Satterfield, A. P. Peskin, J. T. Kelso, and J. E. Terrill. Measurement tools for the immersive visualization environment: Steps toward the virtual laboratory. *Journal of Research of NIST*, 112(5):257–270, 2007.

[8] N. Martys. Study of a dissipative particle dynamics based approach for modeling suspensions. volume 49, 2005.

[9] N. Martys, D. Lootens, W. George, and P. Hebraud. Contact and stress anisotropies in start-up flow of colloidal suspensions. *Physical Review E*, 80, 2009.

[10] N. S. Martys, W. L. George, B.-W. Chun, and D. Lootens. A smoothed particle hydrodynamics based fluid model with a spatially dependent viscosity: Application to flow of a suspension with a non-newtonian fluid matrix. *Rheologica Acta*, 49(10), Oct. 2010.

[11] National Institute of Standards and Technology. SRM Definitions, 2016. http://www.nist.gov/srm/definitions.cfm. Accessed January 8, 2016.

[12] K. Sayrafian-Pour, W. Yang, J. Hagedorn, J. Terrill, K. Yazdandoost, and K. Hamaguchi. Channel models for medical implant communication. *International Journal of Wireless Information Networks*, 17(3–4):105–112, Dec. 2010.

[13] J. E. Terrill, W. L. George, T. J. Griffin, J. G. Hagedorn, J. T. Kelso, M. Olano, A. P. Peskin, S. G. Satterfield, J. S. Sims, J. W. Bullard, J. Dunkers, N. S. Martys, A. O'Gallagher, and G. Haemer. Extending measurement science to interactive visualisation environments. In R. Liere, T. Adriaansen, and E. Zudilova-Seinstra, editors, *Trends in Interactive Visualization*, Advanced Information and Knowledge Processing, pages 287–302. Springer London, 2009.

[14] W. Yang, K. Sayrafian-Pour, J. Hagedorn, J. Terrill, K. Yazdandoost, A. Taparugssanagorn, M. Hamalainen, and J. Iinatti. Impact of an aortic valve implant on body surface uwb propagation: A preliminary study. In *Proceedings of the Fifth International Symposium on Medical Information and Communication Technology (ISMICT)*, Mar. 2011.