VIZSCHEMA – A UNIFIED VISUALIZATION OF COMPUTATIONAL ACCELERATOR PHYSICS DATA*

S. Shasharina[#], J. Cary, M. Durant, D. Alexander, S. Veitzer, S. Kruger, Tech-X Corporation, Boulder, CO, 80303, U.S.A.

Abstract

Data organization of simulations outputs differs from application to application. This makes development of uniform visualization and analysis tools difficult and impedes comparison of simulation results. VizSchema is an effort to standardize metadata of self-described data formats so that the subsets of data needed to visualize physics can be identified and interpreted by visualization tools. Based on this standard, we developed a powerful VisIt-based visualization tool. It allows a uniform approach for 3D visualization of large data of various kinds (fields, particles, meshes) from the COMPASS suite for SRF cavities and laser-plasma acceleration. In addition, we developed a specialized graphical interface to streamline visualization of VORPAL outputs and submit remote VORPAL runs. In this paper we will describe our approach and show some visualizations results.

INTRODUCTION

Visualization is extremely valuable in providing better understanding of scientific data generated by simulations and guiding researchers in designing more meaningful experiments. Scientific models need to be compared with each other and validated against experiments. Consequently, most computational scientists rely on visualization tools. Standardizing on a subset of self-described data formats like HDF5 [1] and NetCDF [2] is a step forward but further standardization on metadata is needed as one needs clues on how to interpret the meaning of data subsets inside these files. For example, how does one recognize that a particular dataset represents a mesh and what kind of mesh is it or how does one indicate that a dataset is mapped to a particular mesh?

In this paper we present a data model (VizSchema) for computational applications dealing with field and particles data in HDF5 files. Based on this model, we implemented a plugin (called Vs) for the powerful visualization tool VisIt [3]. The plugin imports VizSchema compliant data into VisIt. In addition, we developed a graphical user interface that embeds VisIt, allows running accelerator codes locally and remotely and displays default images meaningful to an accelerator physicist who can then further customize them.

Examples of accelerator codes that benefit from this effort include VORPAL [4], a 3D plasma simulation code, and SYNERGIA [5], a multi-particle accelerator code. Both codes are actively used in the COMPASS SciDAC project [6].

VIZSCHEMA DATA MODEL

Principles

The VizSchema annotation identifies the data structures that one needs to expose in order to do minimal visualization. They are not about HOW the visualization is performed (i.e. the type of light or position of the camera); instead, they are WHAT is being visualized (data and geometry) and WHAT needs to be exposed for minimal visualization.

In designing the schema we use the following guiding principles:

- VizSchema assumes that data comes as one of three types: *variable*s, *variables with meshes* and *meshes*.
- The markup uses particular attributes starting with “vs” and specific to type of data.

Variables and Variables With Mesh

A *variable* represents data, which lives on a mesh described outside of the variable array (like magnetic field), while a *variable with mesh* contains spatial information within itself (like particle information having momentum and position typically in one dataset). The suggested markup gives the information to the visualization tool to interpret the data. In the following pseudo-code snippet we show an example of a variable markup:

```
Dataset "phi" {
  Att vsType = "variable"
  Att vsMesh = "mycartgrid"
  Att vsCentering = "zonal"
}
```

Since variables with mesh mix spatial and other data in one dataset, there should be a way to specify the data structure. If the dataset's first N indices specify the coordinates (like in VORPAL), one could use the following markup:

```
Dataset "vorpal_electrons" {
  Att vsType = "variableWithMesh"
  Att vsNumSpatialDims = N
}
```

If the layout of data is different from this order (for example, like in SYNERGIA), one needs to use vsSpatialIndices, which would indicate which indices of the dataset contain spatial informatio

```
Dataset “synergia_electrons” {
  Att vsType = “varibaleWithMesh”
  Att vsSpatialIndices = [0, 2, 4]
}
```

Since the data can be ordered in many various ways, VizSchema also has markup to specify the order of indices starting from the fastest varying. More examples can be found at https://ice.txcorp.com/trac/vizschema/wiki/Variables.

Derived Variables

It is often useful to define additional variables, which are not being dumped by a simulation but present an interesting thing to see as well. That is why, in addition to the prime variable described above, we allow defining expressions using regular mathematical symbols. For example, one could define a density of electric energy as follows:

```
  Group anygroupname {
    Att vsType = "variableDefinition"
    Att vsDefinition = "elecEnergyDensity =
(E_0*E_0+E_1*E_1+E_2*E_2)"
  }
```

Meshes

There is no uniform classification of meshes across tools and experiments. Based on our experience with several codes, we determined that the following mesh type categorizations are fairly general (although the list will be growing):

- Structured grid, which is defined by a list coordinates of its points.
- Rectilinear grid, which is defined by the lists of increasing coordinate values for each axis.
- Uniform grid, which has constant distances between nodes in all directions.
- Unstructured grid, which are defined by points and cells of various types.

Examples of mesh markup can be found at https://ice.txcorp.com/trac/vizschema/wiki/Meshes.

Multi-Domain Data

Quite often simulation data comes from multiple domains and uses different names in these domains, while it would be natural to treat it as one variable in a continuous domain. For such cases, we use vsMD attribute, which instructs visualization tools to connect data having the same value of this attribute.

VS PLUGIN

Based on the data model described above, we implemented a C++ data reader class, which creates an object that reflects the structure of an HDF5 file as it is seen by visualization – lists of variables with the meshes that they live on, variables with meshes, derived variables and meshes and all their metadata.

Next we created a VisIt plugin using the reader's API. This plugin is available for the download at https://ice.txcorp.com/trac/vizschema/wiki/WikiStart and will be available upon VisIt installations in the next VisIt release.

EXAMPLES

Several codes adopted VizSchema and now provide the compliant output during I/O. One can also change the files after they have been generated using PyTables [7] (we have successfully using to change data as the schema evolved and also to annotate SYNERGIA files in accordance with the schema).

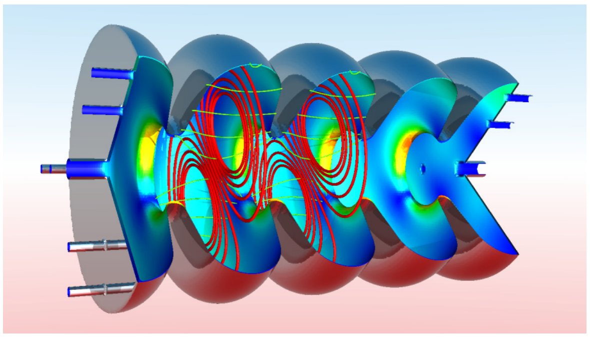


Figure 1: Examples of a visualization of VORPAL data: electromagnetic fields (red and green) and magnetic stress on the cavity (on the walls).

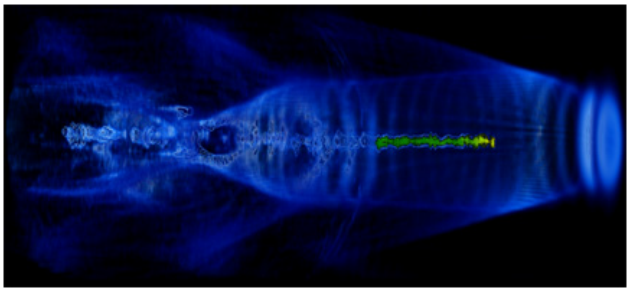


Figure 2: A three-dimensional VORPAL simulation models the self-consistent evolution of the wake resulting from a laser pulse and the acceleration of particles in a laser-plasma particle accelerator. Courtesy of *G.H. Weber and C. Geddes.*

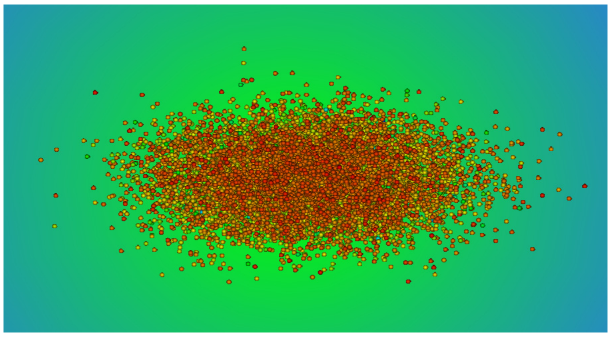


Figure 3: Examples of a visualization of SYNERGIA data: beam colored by the energy of the particles.

Figures 1-3 show some examples of visualizations done using the VizSchema plugin for VisIt. Fig. 1 is a screen capture of OASCR Award for Scientific Visualization at the 2008 Scientific Discovery through Advanced Computation Conference (Seattle) for the video, "Visual Inspection of a VORPAL Modeled Crab Cavity." Fig. 2 has been used as a cover for one of the issues of SciDAC review magazine [8]. Fig. 3 shows visualization for SYNERGIA particle data.

VORPAL COMPOSER

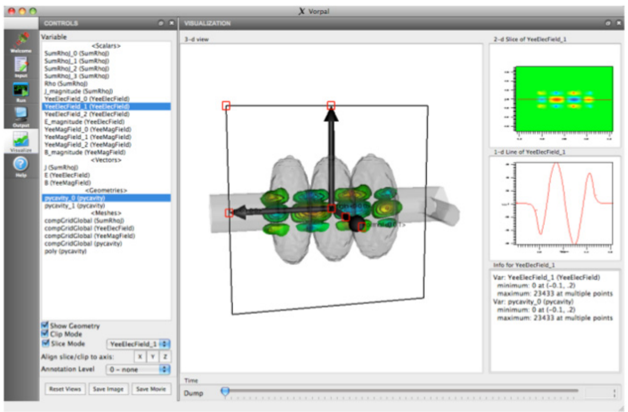
The Composer interface is a customizable application that provides the interface to setup, manage execution, and visualize simulations such as VORPAL. Simulation execution may be local on the user's personal computer or remotely on high performance resources. In the remote case, the application functions as a client connecting to secure shell daemons and visualization servers.

The VORPAL Composer application includes many features useful in producing science from VORPAL simulations. There is a robust editor with validation capability. Run management includes preprocessing, file staging, batch submission script creation, and monitoring feedback. Visualization is provided by embedding VisIt rendering windows. Finally, help is provided based on the user's context within the workflow and simulation contents.

Relying on VisIt to pick out visualization objects from the data files via VizSchema, VORPAL Composer can present the user with an interface to make visualization more efficient especially for the novice. The extra Composer functionality includes:

- A file browser that bundles the hundreds of data files in a typical VORPAL run into one "database" folder.
- Visualization objects can be organized by physical type (fields & particles) rather than by plot-type (pseudo-color & line plot) as seen in the raw VisIt controls.
- Default plots can be automatically rendered for the user with reasonable controls (slice & lineout for example.

Figure 4 shows a screen shot of the VORPAL Composer with the visualization tab displaying three VisIt rendering windows (3D, 2D, & lineout). Note that full VisIt visualization capability is still retained in Composer because the raw VisIt controls can be brought up in a separate window.



Slice Mode: YeeElecField_1
Annotation Level: 0 – none

```
Var: YeeElecField_1 (YeeElecField)
  minimum: 0 at (-0.1, .2)
  maximum: 23433 at multiple points
Var: pycavity_0 (pycavity)
  minimum: 0 at (-0.1, .2)
  maximum: 23433 at multiple points
```

Figure 4: Embedded visualization in the VORPAL Composer application leveraging VizSchema metadata.

CONCLUSIONS AND FUTURE DIRECTIONS

Standardization of the HDF5 output using consistent markup for visualization proved to be useful in accelerator physics applications as well as other domains having notions of fields and particles. The developed VisIt plugin is available for all interested parties.

In the nearest future we intend to extend the schema and the plugin with more detailed metadata for unstructured meshes, bring more applications into the VizSchema realm and extend VizSchema for NetCDF.

ACKNOWLEDGMENT

We would like to thank the VisIt, FACETS and VORPAL teams. We also thank C. Geddes and G. Weber for providing us with images and J. Amundsen for providing us with SYNERGIA data.

REFERENCES

[1] http://hdf.ncsa.uiuc.edu/HDF5/.
[2] www.unidata.ucar.edu/software/netcdf.
[3] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B, J Whitlock and N. Max, A Contract-Based System for Large Data Visualization, *Proceedings of IEEE Visualization* 2005, pp 190-198, Minneapolis, Minnesota, October 23--25, 2005.
[4] C. Nieter and J/ R. Cary, "VORPAL: a versatile plasma simulation code," *J. Comp. Phys.* 196, 448-472 (2004).
[5] J. Amundson and P. Spentzouris, J. Qiang and R. Ryne, Synergia: A 3D Accelerator Modelling Tool with 3D Space Charge, *Journal of Computational Physics*, Volume 211, Issue 1, 1 January 2006, Pages 229-248.
[6] http://www.scidac.gov/physics/COMPASS.html.
[7] http://www.pytables.org.
[8] http://www.scidacreview.org/0903/index.html.