VISIT PYTHON INTERFACE

Zubieda Alali
Kansas State University
Email: zalali@ksu.edu

ABSTRACT

VisIt is an open source application for scientific data analysis and visualization. With VisIt we can visualize the data where it is generated, which reduce the need to move data. Moreover, Python is an open source, interpreted, and object-oriented scripting language that was chosen for VisIt's scripting language due to its ease of use and flexibility. VisIt's Python interface was implemented as Python module and it allows us to enhance our Python scripts with coding to control VisIt.

INTRODUCTION

The most important and challenging question when starting this project is: "Where do I start? One option is to use what is called session files that contain all of the information required to recreate plots that have been set up in previous interactive VisIt sessions, and since these session files contain all of the information about plots, this option would make scripting easier since they can be used to do the hard part of setting up the complex visualization, then to animate through time or make few changes in the plots in some way. The second choice was to Script every aspect of plot initialization, which is exactly what I did in this project. So, the ultimate goal of this project is to launch and control VisIt using Python-based command line interface (CLI) to setup plots and direct visualization operations. VisIt has a rich a command line interface that is based on Python 2.7, and there are several ways to use the CLI for example, we could launch VisIt and run our scripts in Linux using the command below:

```
path/to/visit/bin/visit –nowin -cli -s <my_script.py>
```

Another way I tried in this project is to Launch VisIt so that a visualization window is visible and interactively issue CLI commands. A third way was to use both the standard graphical user interface (GUI) and CLI simultaneously

NOMENCLATURE

Triga Nuclear Research Reactor

PYTHON CLIENT INTERFACE

Two types of files were provided to be visualized in this project; the first one is data.vts which includes the output of a steady state view of a scalar flux density in a reactor. Vts is short for Video Title Sets, which is a rendered file type that contains all of the visual data. The second file is mesh.vtk. To introduce the API in this project I needed to provide a sample of my script that demonstrates VisIt's five primary visualization building blocks, which are listed below:

- Databases: File readers and data sources.
- Plots: Generating image from a 2D or 3D model.
- Operators: Filters implementing data set transformations.
- Expressions: To enable the creation of derived quantities from existing mesh fields.
- Queries: Data summarization operations.

To handcraft any script from scratch we start with opening databases, then creating plots, implementing the right operator, and animating through time. VisIt itself provides the functions such as OpenDatabase function to open a database. Once a database has been opened, we can create plots from its variables using the AddPlot function. The AddPlot function takes a plot plugin name and the name of a variable from the open database. Once we've added a plot, it means that it has not yet been submitted to the compute engine for processing. To make sure that the plot gets drawn, we call the DrawPlots function.

Listing 1: Python script to open and plot data of a scalar field

```
# Step 1: Open the database
OpenDatabase('data.vts')
OpenDatabase('mesh.vtk')

# Step 2: Add plots
AddPlot("Pseudocolor", "phi")
#Slice the volume to show only three
# external faces.
# Step 3: add operators
AddOperator("ThreeSlice")
tatts = ThreeSliceAttributes()
tatts.x = 0
tatts.y = 0
tatts.z = 0
```

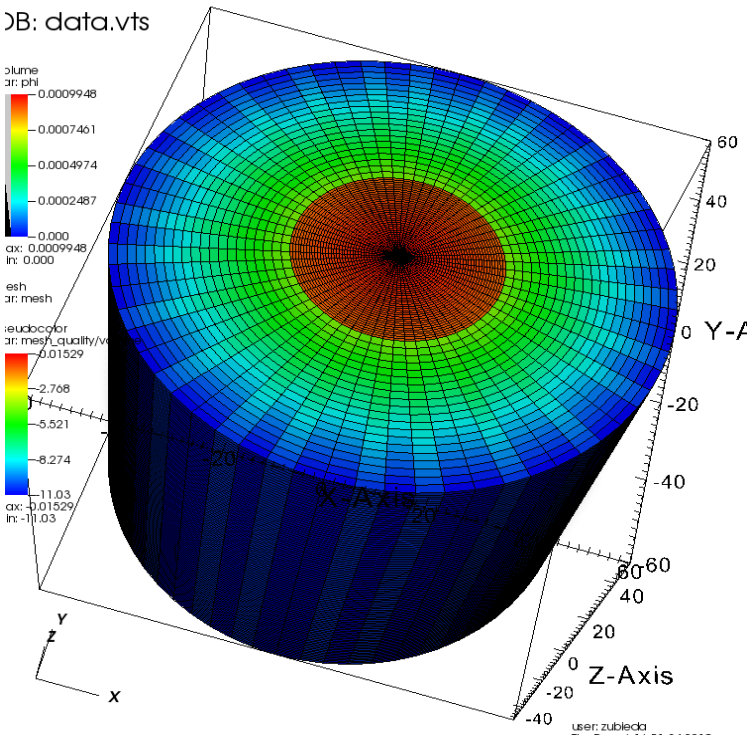

Fig. 1: Pseudocolor and plot setup using the script

Figure 1 above shows a Pseudocolor plot that is created to display the scalar field named ‘phi’. The mesh plot displays the computational mesh over which a database’s variables are defined. It is important to mention that the features of the VisIt interface are dependent on the version of VisIt we are using. The version I am using for this project is VisIt 2.13.2.

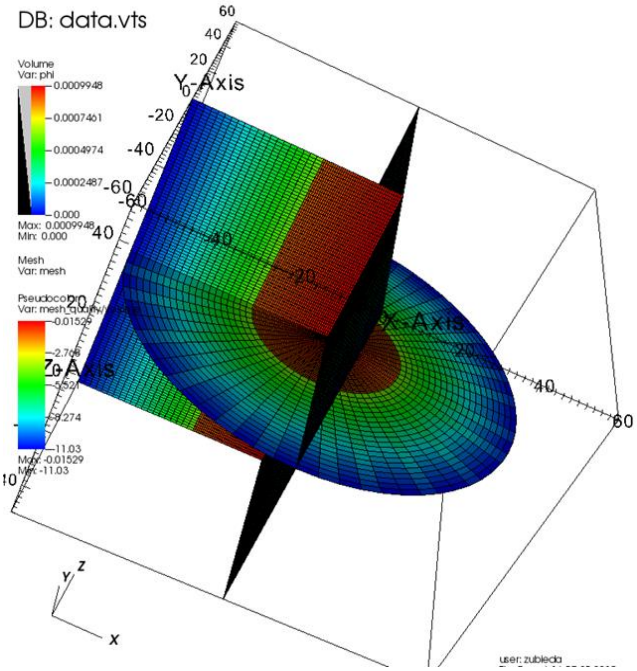

Fig. 2: Pseudocolor and ThreeSlice operator plots

In figure 2 above the mesh is transformed by a ThreeSlice operator to limit the volume displayed by the Pseudocolor plot to three external faces. Clearly, the ThreeSlice operator slices 3D databases using three axis-aligned slice planes and leaves the resulting planes in 3D where they can all be viewed at the same time. The 3D point where all axis-aligned slice planes intersect could be set using the ThreeSlice operator attributes then selecting the x, y, z values. These attributes are easily controlled and modified.

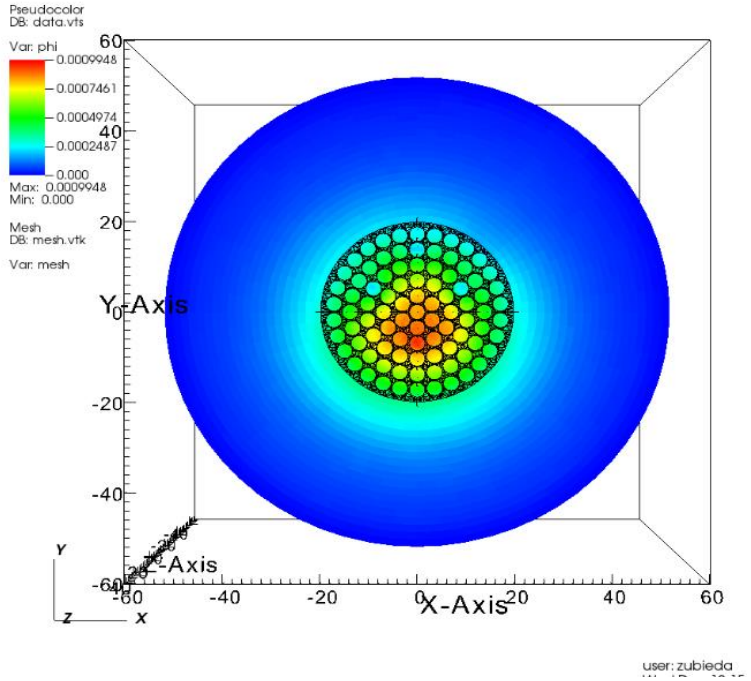

Fig. 3: Pseudocolor and “Slice” operator plot

A Pseudocolor plot to which a Slice operator has been applied to the core data has been shown in figure 3. Furthermore, VisIt allow us to perform a number of different queries based on values calculated about plots for example, to find the maximum value of a scalar field the following was implemented:

Query("Max") val = GetQueryOutputValue()

SETTING THE 3D VIEW

I should point to the fact that 3D view is much more complex than the 2D view. However, the best way to get new views to use in my script is to interactively create the plot and repeatedly call GetView3D() after I finished rotating the plots with the mouse. Then I pasted the printed view

information into my script and modify it slightly to create sophisticated view transitions.

USING GRADIENT BACKGROUND COLORS

Clearly, VisIt's default white background may not be the best looking background color for presentations. Adding a gradient background under my plots is an easy way to add a small professional touch to my visualizations. Figure 4 shows before and after adding a gradient background.

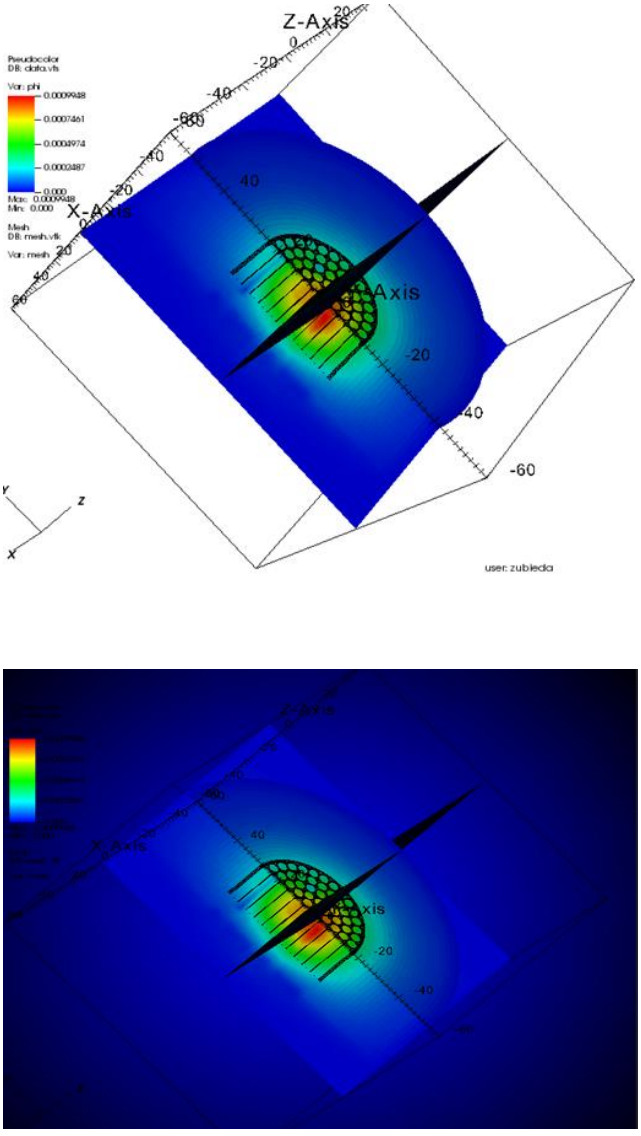

Fig. 4: Before and after image of adding a gradient background plots

ANIMATION THROUGH TIME AND SAVING IMAGES

Animation is used mainly for looking at how databases evolve over time. Creating visualizations using just one time step from the database does not show time-changing behavior. To be more effective, visualizations must be created for all time steps in the database.

Last step would be to create a movie of animated core images. This could be accomplished by saving images of the animation and finally encoding those images into a movie. That requires ffmpeg to be installed and available in my PATH

CONCLUSION

An API is an application programming interface. That includes a set of rules which allows programs to talk to each other. Simply, the developer creates the API on the server and allows the user to talk to it. Unfortunately, there's a high chance developers may come across hard challenges, especially if they've gotten their data from another source, which might be the case here. How do you use the data? What is right tool that you need to use? How to fix the problems and analyze error messages? Those are some of the challenges that I have experienced in this project.

In this project it is crystal clear the huge role that Python plays in VisIt's software infrastructure. The example in this project reveals how VisIt visualizes data by creating one or more plots in a visualization window that includes Mesh plots, and Pseudocolor plots. Plots take as input one or more mesh, material, scalar, or tensor variables. It is possible to modify the variables by applying one or more operators to the variables before passing them to a plot. Examples of operators include arithmetic operations or taking slices through the mesh. It is also possible to restrict the visualization of the data to subsets of the mesh. Many references confirmed

that Python has long been an asset to VisIt as the foundation of VisIt's scripting language.

FOR FUTURE WORK

To write new Databases and Operators in Python.

REFERENCES

- Whitlock, B. et al. VisIt Python Reference Manual. http://portal.nersc.gov/svn/visit/trunk/releases/2.3.0/VisItPythonManual.pdf
- VisIt Python Interface Manual. http://www.visitusers.org/visit/PythonInterface1.5.3.pdf

- https://github.com/Zubieda/Project-ME701
- VisIt Website. https://wci.llnl.gov/codes/visit/