VisIt and Imaging Guide 2016

Sean Connelly

June 10, 2016

1 Intoduction

VisIt is a complex and incredibly useful tool for scientific visualizations. That being said, it has a very large list of functions and capabilites, which can get confusing at times since the documentation isn't always very clear. It is my goal that this guide will help you become familiar with all of the functions in VisIt that we typically use in this research group, plus a few more that are also occasionally useful. However, I haven't explored every nook and cranny of VisIt, so a complete list of the functions and how to use them can be found here: http://visit.ilight.com/svn/visit/trunk/releases/2.10.0/VisItPythonManual.pdf.

Secondly, there will be advice on to how to make "good" images, that is, ones that will end up being put into the papers that Stu and the post-docs will write. But images can always be improved, so take my advice as a guide and improve upon it.

Note: Currently this group uses HDF5 files for the data, .3d files for the black holes, and Python for the scripts, so I will be using that language over the course of this guide.

2 File

The following functions can be accessed through the "File" button at the top of the VisIt menu. There are a few items on the dropdown menu that I will talk about, which involve opening and looking at databases, saving sessions, and saving images and movies. I'll mention common errors as well.

2.1 Opening Files

As of this writing, most of the data (as HDF5 files), is organized into folders (3d_data_201511141451, or something like that), and in each folder is a bunch of h5 files (labeled as Bx.file_0.h5, Bx.file_1.h5, etc). All of the files of a certain type, say Bx, form a database (or dataset), which you want to load in all at once. All of the files in a database are sections of a total volume in space, so if you have Bx.file_0.h5 through Bx.file_119.h5, you have 120 sections in space, which is why you need to load them all in at once if you want to properly draw things. If your database contains multiple time steps, each file will contain all the time steps. In order to grab all the data in the database, make sure that the file grouping is set to "Smart". But you can only grab one file at a time if you set it to "Off" (this is useful when you want to only grab one .txt file, or something like that).

Databases can also take a long time to load into VisIt, depending on the size of the file and the speed of the computer. So if it is taking 5-10 minutes to load in a database, that is normal. But if it is taking more than 30 or 40 minutes to load in a database, you should probably look into splitting up your data into smaller folders.

If you want to see more details about a file/database, you can select "File Information". This gives a very detailed description of the active source file. Some of the more useful details are where the file is located on your computer, and the cycles and times of each timestep. I'll come back to these where I talk about the clock in the "Objects" section of the Annotation menu.

There are a number of errors that you can get due to loading in databases. The common ones are due to the dataset not being there, missing files inside the folder, or trying to load too much data into VisIt at one time. The last one is sneaky, as VisIt will tell you that it can't read the database, when really, there just is too much data in VisIt. Also, when this happens, VisIt will refuse to load in any more datasets unless you restart it. But this only happens when you try and load in more than 5 databases or so, totaling more than $\tilde{2}$5GB (not 100% sure about the number). One way you can get around this is by not loading in some files in your database (like Bx.file_160.h5 through Bx.file_200.h5 or whatever) that show up outside your field of view, but try not to do this, since there is a chance it can mess with the resulting image (this depends on the case that you are visualizing). Magnetic field lines seem to be especially vulnerable.

2.2 Sessions

Sessions should become your best friend. In creating images, you may load in 5 or so databases that are all plotted at the same time with various parameters. Sure, you can write a Python script to create and draw everything (more on that later), but if you save the session, then all you have to do is load in the databases and open the session file, and VisIt will restore itself to exactly as it was when you first saved the session. There is an option to "Restore session with sources", but in my experience, that takes a really long time to finish, so you are better off loading in your databases and then restoring your session.

2.3 Saving Images

Now that you have created a pretty image, you want to save it! VisIt will, as a default, save it to your home directory, so if you want to name it a certain way or put it in a certain place, you'll need to open "Set Save Options" and do that yourself. Typically we save images in as high quality as possible (as .png's with a resolution of 1920x1080). However, this means that your image will extend farther left and right than what you may see in your VisIt window, so keep that in mind. Also, saving images takes much longer than drawing images (sometimes much, much, much longer), so if you want a quick image, you're better off just doing a printscreen. But you should save a final image uses VisIt's saving method.

Also if you want to create a 3D movie, then you just have to turn on the "stereo" option, and VisIt will save two images, that when put together, will create a 3D image.

If you have multiple time steps, you can also use the "Save Movie" option to get a movie, but usually it'll be easier to just generate all of the images and

stitch them together using mencoder.

Occasionally you'll get errors telling you that VisIt cannot save the image. When you get this error, double check that the folder you want to save to actually exists, and that you can write files to that location.

3 Controls

These function can be accessed through the "Controls" dropdown menu. Here we will look at Annotations, Color Tables, Correlations, the Command Line Interface (CLI), Expressions, Lighting, and Picks.

3.1 Annotations

In Annotations, you can set a bunch of parameters that has to do with all of the details of the image that have nothing to do with the data. I'll split it up by the important tabs.

3.1.1 General

Here there is the user information, database information, and an option to turn all legends on and off. We never want to see the user and database information, so that is always kept off, but sometimes we want to see the legends, so you can leave that on and turn it on or off on a plot by plot basis.

3.1.2 3D

Here you have the option to toggle the triad, bounding box, and the axes. The triad is very useful when manually rotating the image, and the other two are useful when you want to get the precise size of the object. But we don't want to see them in the final image, so make sure to turn them off before printing out the final image! You also have the option to put the bounding box wherever you'd like, and the axes will move with it.

3.1.3 Colors

The most useful setting in this section is the background color. At the time of writing this guide, we typically use a blue defined by $(r, g, b) = (55, 118, 255)$, since the black hole and disk show up well aganist it. You could also set a background image (such as a starry space background), but a black hole wouldn't show up so well aganist it, so we don't usually use this option.

3.1.4 Objects

There's a number of useful things that you can do with this tab. In every plot you create, a legend will be put in this section. If you want to edit that legend, this is where you would go. You can move the legend, change its size, change

the number of tick marks and what they say, the color of the text, and a few other things.

You can also create text that you can place on the image. The most important way that we use this in creating labels for a plot and the clock. The clock is just a piece of text that says what the time of the image is, in units of t/M. But how do you go between code units (cycle number) and t/M? The easiest way is to take the time that VisIt gives you, and divide by the ADM mass. To be more general, currently we get the time by using this formula: $\frac{t}{M} = \frac{cycle \times a}{sep \times mass}$, where *cycle* is the cycle number (also know as iteration number), a is the size of the timestep between frames in VisIt, *sep* is the size of the timestep in terms of iteration number (typical 256), and *mass* is the ADM mass. Using the text options, you can set the position, size, and color of this clock, just make sure that you update it with each new timestep that you plot.

3.2 Color Tables

Color Tables can help to create a better image. There are a large number of default color tables that have various different colors and opacities. Typically, these don't quite give us the nicest images, but they are a good place to start when creating a new color map. Also be careful if you use this window to create a new color table, as if you don't change the name, you'll overwrite an existing color table.

3.3 Correlations

There are two useful things to mention here, Data-Level Comparisons and Database Correlations. Data-Level Comparsions take data and put it on the same grid (such as putting together Bx, By, and Bz), whereas Database Correlations connect databases together so that they evolve through time together. You can use the dropdown menu and put all of the databases into one grid, but its much easier to just create an expression. I'll mention more about this in the sections on the CLI and Expressions.

3.4 Command Line Interface (CLI)

This is probably one of the most important sections, next to Sessions. The Command Line Interface (CLI) is an incredibly powerful tool that allows you to "script" VisIt. Scripting means that you create a Python code that sets up all of your settings and plots, and then you run it in VisIt and it does all of the work for you. This means that you don't have to go back and click lots of buttons if VisIt crashes or something else happens. The CLI is a Python interface, so you will use Python commands to create plots and set options. Also the interface is a very old school interface, so the arrow keys don't work in it. Probably one of the most useful features is to type commands such as "vol = VolumeAttributes()", which will provide you with a list of settings for the Volume plot. If you change any settings, then you will have to set them, which for the volume plot can

be set with "SetVolumeAttributes(vol)" (and similarly with other plots and operators). If you want to see every VisIt function that you can use in the CLI, just go into the CLI and type "dir()". Using these commands, you can write a script to set up any plot that you choose. I'll mention a bit more of the settings of each as I talk about each plot later. Also, in order to run a Python script in the CLI, just type in "Source("/path/to/file/script.py")", where you'll insert the appropriate path and file.

3.5 Expressions

Expressions are powerful and can help you plot exactly what you want to see. However, they also can be a source of most of the errors that you encounter. If you go into the expressions window, you'll see a list of expressions that you can add to by pressing "New". Also, in creating expressions, stay in the Standard Editor window. You can use the other window, I just don't know how. There are a huge amount of functions that come with VisIt, which can help you plot exactly what you need. Here is a list of the ones that we typically use, with descriptions on how to use them.

- abs
  - Format: abs(value)
  - Pretty straightforward, it takes the absolute value of whatever variable you put into the expression.
- log10
  - Format: log10(value, default)
  - This function takes the base 10 logarithm of the value that you put into the expression. You can also put in an optional default value, which the function will substitute in for negative arguments.
- sqrt
  - Format: sqrt(value)
  - This just takes the square root of whatever you stick in the expression.
- coord
  - Format: coord(mesh)
- conn_cmfe
  - Format: conn_cmfe(database,mesh)
  - This is an important one. You use this function if you want to create a database correlation. The format is that you take a database and insert it into a mesh. So to give an example, currrently we use the Carpet AMR grid as our mesh, and then load in databases (Bx, By, etc),

and put them in the Carpet AMR grid. So that would look like this: conn_cmfe(</path/to /Bx.file_* database[0]id:MHD_EVOLVE- -Bx>, <Carpet AMR-grid>) where we are putting the Bx datbase into the Carpet AMR grid.

- lt
  - Format: lt(valL, valR)
  - This is equivalent to valL $<$ valR.
- gt
  - Format: gt(valL, valR)
  - This is equlivalent to valL $>$ valR.
- if
  - Format: if(condition, op1, op2)
  - This means that if the condition is true, then do op1. Otherwise, do op2.
- and
  - Format: and(cond1, cond2)
  - This should also be pretty straightforward, but if you need more than one condition to be satisfied, then use this function. Using it in an if statement would look like this: if(and(cond1,cond2), op1, op2).

If you end up typing your expression incorrectly, VisIt won't tell you until you try and plot your data, at which point it probably won't plot anything. Also, when you change your expressions, you'll have to redraw your plot in order to see the effect. If you get errors, check to make sure that all of your paths are correct, then make sure that you have the correct amount of ()'s, then check to see that all of the expression names are correct and such. Also, make sure that the type of your expression is correct, that is, if you want the result to be a scalar, choose "Scalar Mesh Variable", and if you want the result to be a vector, choose "Vector Mesh Variable". To give an example of an expression, sometimes you may want to know the strength of the magnetic pressure above the black hole. One way we can do this is but measuring a parameter know as $\log \frac{b^2}{2\rho}$. To draw this in VisIt, we need to put together two databases, smallb2 (b^2) and rho_b, and then draw the expression. So as expression, it would look like: log10(<smallb2>/(2*<rho_b>)).

3.6 Lighting

In this section, you can change where the lighting is coming from. In the past we haven't done much with this, but I'm mentioning it for completeness sake.

You can click and drag to make the light come from any direction you want, and you can ever have multiple lights at the same time. You can also change the color of the lights too.

3.7 Picks

This is a useful section of which I have barely scratched the surface. VisIt seperates a volume of space into "nodes" and "zones" (As of this writing, I'm not sure of the difference or how they are organized in space). If you "pick" a specific node/zone, you can learn about the information that that region in space, such as the data value and incident nodes/zones. This is useful if you want to look at a specfic region of space, for example, where the density is 1E-6. A useful function here is "NodePick(coord=(x,y,z))", which will grab the nearest node to that physical position. There's plenty more too that I haven't explored, so just look at the VisIt manuals to learn more about them.

3.8 View

An important topic in plotting data in 3D is the view, since where you view an image can introduce a slight bias, since the view angle highlights important details in the image. There are a number of different view box, but I am going to focus on the 3D view. If you open the View menu underneath the Control dropdown menu, you'll see a whole bunch of parameters that you can set:

- Focus: This is the center of the image, if you move or rotate your image, it will move around this point.
- View Normal: This is your "line of sight" vector, it points from you (the viewer) to the focus of the image.
- Up Vector: This vector points straight up, and together with the View Normal vector, defines a specific viewing angle. If you want to change where you are looking from, you'll want to change these two values. But make sure that the two vectors are orthogonal, or else the image will look weird.
- Angle of View: How wide of a view you have when looking at the image. I have always left this at 30.
- Near Clipping: If you are viewing a cube, it is the face of the cube that is facing you and is closest to you. Anything that would be closer to you than that gets cut off.
- Far Clipping: Same as Near Clipping, except this is the face of the cube that is furthest from you. Anything that would be farther gets cut off. Typically I leave the Near and Far Clipping at the values that VisIt assigns.

- Image Zoom: Lets you zoom in and out of the image. Typically when changing views, Image Zoom, View Normal, and Up Vector are the three values that get changed.
- Parallel Scale: Changes the relative size of everything.
- Shear: Streches the axes in one or more direction. I haven't found a reason to change this yet, so I leave it at the default value of (0 0 1).
- Eye Angle: This is useful for 3D movies. It starts at a default value of 2, but you can make it higher to increase the level of 3D in your 3D movies.

It is important to note that you need to have all of these parameters set in your script, or else you won't get the right image. For example, if the parallel scale gets changed from 1800 to 1500, then your 30 zoom will give different images. Also, to get all of these settings in the CLI, just type in "View3DAttributes()".

4 Plots

Now we reach the fun stuff. In this section, I will discuss how to plot data using the pseudocolor, streamline, vector, and volume plotting methods, when it is best to use each one, and how to get a nice and pretty picture with each. I will assume for this section that all of your databases have been loaded in and your expressions are created correctly. Also, for all of these plots, you can save your attributes as an XML file if you click on the Save button in the bottom right of the attributes box. That way, you can load them in later and don't have to redo all of your hard work.

4.1 Pseudocolor

Creating a plot is pretty simple, and the Pseudocolor plot is the simplest plot that we commonly use. In order create it, you click on add, move down to Pseudocolor, and chose which variable (database or expression) that you want to plot. A Pseudocolor plot is a color plot in 3D space, so it can only plot scalar variables. Here's a couple useful things to help get you started with the Pseudocolor plot:

- The color is set with a color table (see section above on color tables).
- You can see the maximum and minimum of the plot, where it plots linearly or logarithmically, as well as a number other nifty little things.
- Currently we use it to plot black holes (along with the Delaunay operator, see the next section) and if we want to get a basic idea for values in a dataset.
- A useful tool to see inside the plot is to use the Box operator, which is outlined in the next section.

- You can use this plot to get a "Pseduo-2D" plot if you combine it with a Box operator and only look at one plane (for example, the xz plane).
- In order to see all the functions that you can play with for the Pseudocolor plot, just go to the CLI and type in "PseudocolorAttributes()", and it will provide you with a list, as well as the current values for each .

4.2 Streamline

Streamlines are incredibly useful, and can show the direction of particles in a 3D space. Because of this, you can only use streamlines for vector variables. Currently, we use them for the magnetic field lines of a system, which can look very different for a system depending on where you choose your seed points, from which the magnetic field lines will be drawn. As a result, I'll use the term Streamline and field line interchangeably.The trickiest part of the Streamline plot is dealing with broken field lines, since in reality, those don't exist, but VisIt is a free piece of software, so it has limitations that we must deal with. I'll explain some tools and tricks for dealing with these and other features of the Streamline plot:

- There are a few different ways in which you can seed points. The most common ones used are circle, sphere, and point list.
- Point List is the most powerful since it's very easy to write code for the positions of points in any of the other shapes (circles, spheres, etc), and you can use the point list to have seed points in shapes that VisIt doesn't provide you (like cylinders), as long as you can write the code for it.
- The "Force Node Centering" option can help to connect Streamlines at the $z = 0$ plane, but it also slows down the time it takes to draw the image immensely, and doesn't always work. For these reasons, I suggest to keep this option off.
- Turn "Integration Direction" to "Both" so that the lines go both backwards and forwards in time (which will allow them to connect).
- There are a number of different numerical integration algorithms, but Dormand-Prince seems to give the best results.
- Keep your tolerances high so that they better approximamte what the fields will actually look like, but don't push them too high, or else there will be major errors in the integration. Setting both the absolute and realtive tolerance to 1E-10 seems to do the trick.
- I would suggest setting the maximum number of steps to 50,000 since that will allow the integration to go as far as possible. Setting the number of steps higher than this hasn't seemed to make a difference to me.
- There is an option to use a color table for the Streamlines, but typically it is better to use a solid color, like white.

- There are a couple different options for the opacity. Typically we want it to be a constant value for every point on the magnetic field lines, whether that be 80% or 20% depends on the situation. If they are just in free space, we can set them to 80% or 100%. If they are inside/just outside a a star, we may want the opacity to be lower so that we can still see the structure of the star.
- Currently we use a line width of 5, which keeps the Streamlines bright and clear even if we shrink the image.
- There are a number of options related to seeing the seeds, which is useful for testing purposes. But showing the seeds also significantly slows down the time it takes to draw the image.
- The Advanced section of the Streamline options allows you to suppress warnings.
- As with the Pseudocolor plot, you can get all of the options for the Streamline plot by typing "StreamlineAttributes()" into the CLI.

Those were all pretty general tidbits about Streamlines themselves. Here are a couple more points of advice that are useful when drawing magnetic field lines, stars, and looking for jets:

- A sphere of points around a star give a nice dipole shape, but if it is lacking in "exterior lines", then you can add a large circle of points way outside the star, on the equator of the star.
- The best way to check for a jet is to have a circle of points above and below the black hole, as that is where the field will twist up to create the signature collimated shape.
- You'll have to find a balance between too many and too few field lines. That line is somewhere around where you'll have a lot of field lines in order to show the structure of the field, but not too many so that you'll block out the structure of the star or object underneath the field lines.
- Broken field lines are bad, and in my expererience, changing the options of the Streamline plot doesn't really make a difference. So the easiest way to fix this is to nudge your seeds a bit (whether that be up, down, in, or out) until your get some nice field lines that aren't broken.
- It's okay to have a few broken lines, just try not to have the major lines be the ones that are noticeable. Something you can do is use a program like Gimp to edit out broken lines, but try to only do this if it's one or two broken lines. There is a fine line between getting rid of some bad data and changing the science of the image by accidently introducing a bias into the image by removing lines, so be careful. Also, if you do choose to remove some lines, be careful to do it so that it isn't noticable that you removed the lines, such as a discoloration in the background.

4.3 Vector

A Vector plot is a nice way to show vector data using arrows placed all over a 3D volume. Here is some information and advice to help get you started:

- Typically we use a fixed number of vectors, but whether that is 100 vectors or 1,000,000 vectors depends on the situation.
- You can set the color of the arrows to be a solid color, or if you want the color to vary depending on your vector field, then you can use a color table.
- The scale changes the size of all the vectors, and for most cases, you'll want to scale by magnitude.
- The size just changes the size of the arrow heads, and the width changes the thickness of the lines on the arrows.
- When you first plot a Vector plot, you may get arrows everywhere, which probably won't help you very much. In order to get something that looks nicer, you can use a Box operator (described in the next section) or you can use an expression to focus on a smaller area. For example, you could plot velocity vectors when $\log \frac{b^2}{2\rho} > -2.1$.
- But even with this you may not have the nicest image. In order to do that, you may have to nudge your Box, number of arrows, or expression a bit (just make sure you Redraw the plot after you change your expression or else you won't see a change!)
- In order to see all of the settings that you can change for the Vector plot, type in "VectorAttributes()" into the CLI.

4.4 Volume

The Volume plot is focus of the image, it is usually what makes the image look pretty, so I will try and explain it in detail, as well as how to actually make it look nice. As a comparison, you can think of a Volume plot as a version of the Pseudocolor plot that allows for much more detail, allowing for a much higher quality image. Currently we use the Volume plot to draw the density of matter in a system, whether that be stars or disks of matter. Here is some information and advice to get you started:

- There are five different rendering methods that I will seperate into three different categories. The first contains the Splatting rendering method, which is the fastest method, but gives you the lowest quality image. For this method you set the number of samples, and you will need at least 50,000 to give you anything resembling the higher quality methods. I would use this rendering method for testing purposes, since it is a very fast method.

- The second category is the medium speed, medium quality rendering methods, which are the 3D Texture and SLIVR methods. Both of these methods will work a bit slower than Splatting, but create much higher quality images. However, these methods have their limits. For one thing, they look odd and a bit more cartoon-y than the other methods, and they don't work well with other plots. For example, if you are drawing magnetic field lines, they'll be drawn on top of disks of matter instead of inside, which is where they should be.

- The third category is the slowest speed, but highest quality methods. The category contains the Ray Casting methods. VisIt provides you with two, one in color and one in greyscale, but we would like to preserve that color information, so I will focus on that method.

- The variable that will determine how fast Ray Casting runs will be the Samples per Ray. If you want a relatively fast image that will give you an idea of how the plot will look, you can set this to 100 (or even 10 if you want to barely be able to see it). If you want a high quality image that can be put into a paper, you can set this to 1000. Anything above that will take literally forever to render, that is, on the order of many hours.

- Lighting is important for a high quality Volume plot, make sure that the box is checked that turns it on. There is also a section that is labeled "Low grad lighting", which adds contrast to the image. This sharpens smaller details and creates a better picture overall. I think that you have the best image when the reduction factor is set to high and the reduction max value is set to 0.1, but you could play around with this.

- Now here comes the hard part: opacity. If you jump over to the tab that is labeled 1D transfer function, you'll see lots of colors and numbers. You can set the colors, move where they are on the label, set the maximum and minimum values, as well as set if it's a linear or logarithmic scale.

- In the box below, you'll notice that there are lots of buttons, and if you drag your mouse across the colorful box, the amount of semi-transparent white changes in the box. The more white there is in a certain area of the box, the more opaque that value will be in your Volume plot. So if you want to more prominently show the higher valued material, then there should be a lot of white on the right side of the box, and very little on the left.

- Creating a nice looking color map (that's colors and opacities) can be one of the most difficult (and frustrating) parts of this REU group, since every time you make a little change, you have to wait a few minutes for VisIt to draw the image, and you might make hundreds of these little changes. But with time and practice, you'll get the hang of it.

- There are a couple of buttons to help you with the opacities. First, I would suggest maximizing the Volume Attributes screen so that you can make

finer changes to the opacites. If you want to draw it by hand (which I'd recommend), you want to make sure that the "Freeform" button is clicked. But if you want a bit of help, you can click the "Gaussian" button, which provides you with a bunch of Gaussians, which you can use to change the position, width, and height of each Gaussian. The smooth button will smooth out the jagged edges that you may have drawn in, which can help make the transition between one color and another more natural. The "Attenuation" button will raise or lower all of the opacities at once, without changing the hills and mountains that you have pain-stakingly drawn.

- Currently, we use the Volume plot for stars and disks of matter, which means the higher density material is deeper inside the star/disk. But choose your colors carefully, because if you have low opacity red over higher opacity white, it is going to look orange. In order to minimize this effect, I recommend putting darker colors deeper inside the star/disk, since you want to aim for stark colors at each level. Is that possible? Probably not, but its a good goal. But even putting darker colors inside, you will have a bit of this washing out effect, so you will still have to play with opacities to minimize it more.

- As with the other plot, use "VolumeAttributes()" in the CLI to see the full range of options that you can set for this plot.

5 Operators

Here are some operators that you can use to make your pictures look nicer. VisIt has a lot of them, and I encourage you to explore them, but the ones that I have found most useful are the Delauney, Box, and Reflect operators.

5.1 Delaunay

The Delaunay operator will create smooth surfaces for you. So if you have a sphere of points, you can apply a Delaunay operator to it to have a smooth sphere of points. I have always left the default setting for this operator, which you could change with "DelaunayAttributes()" in the CLI. It will work for other shapes too, but typically we use this operator to draw the surface of black holes.

5.2 Box

The Box operator is an incredibly useful operator. With it, you can create a box of any size (in the options you set the extents of the box), and then your plot will only be drawn inside that box. Alternatively, you can choose "Invert", and your plot will only be drawn outside that box. This is very useful in many cases, such as the Vector plot, in which you can crowd more arrows in a smaller area. If you wish to change the extents of the Box in the CLI, just use "BoxAttributes()".

5.3 Reflect

Typically, the data sets that we recieve assume symmetry about the $z = 0$ plane. That is, we just get data for $z > 0$, since it is the same for $z < 0$. So that means that we have to reflect the data about the z-axis. To do that, you just use a Reflect operator. If you open the Reflect operator options, under 3D, you will see a pink ball in one of the octants. That represents your original data. Since you want to reflect it across the z-axis, then you click on the space just to the right of it, which should turn into a green square. Alternatively, you can access the options when you type in "ReflectAttributes()" into the CLI, and if you type in "reflections = (1, 0, 0, 0, 1, 0, 0, 0)" and then type in "SetReflectAttributes()", then the CLI will reflect it for you. This will work for reflecting in any directions, you just will have to highlight a different octant.

6 Closing Remarks

There you have it! A comprehensive guide on all things VisIt and imaging. Like I said before, this is not everything that VisIt has to offer, but it is about everything that I know about VisIt. However, if you know more, or things change in this REU group, please feel free to add to this document! And if no one in the group can answer your question, feel free to send me an email at sconnel42@gmail.com, and I'll see if I can answer it.

Have fun!