VisIt, list of basic visualization tasks.

12.13.2022, Playlist for Voiced VisIt tutorials (in progress)
12.13.2022, VisIt Tutorial Playlist on YouTube

Open file
Open:mi_0011_1-720x180.nc

Open multiple files from a single run
Go to directory with Mark's fv3 one year run
Open:File Grouping=Smart:mi_*.nc database

Show plot
Add:Pseudocolor:temp

Show mesh
Add:Mesh:SigmaLayer_mesh

To see depth coordinate, scale the Z axis
Add:Pseudocolor:temp
temp:Operators:Transforms:Transform:Scale:Z=1000

Show a single layer
Add:Pseudocolor:temp
Control:Subset:Sigma_Layers:[Pick Layer]

Show a vertical slice
Add:Pseudocolor:temp
temp:Operators:Transforms:Transform:Scale:Z=1000
temp:Operators:Slicing:Slice:Orthogonal:X Axis:flip:Percent:[X]
temp:Operators:Slicing:Slice:Orthogonal:Y Axis:no-flip:Percent:[Y]
(Just use percent to start. I need to figure out how the other options work for this grid.)

Contours of specific temperatures for a single layer
I think it is confusing to do contour lines for more than 1 layer
Add:Contour:temp
Control:Subset:Sigma_Layers:[Pick Layer]
temp:Contour:Values:[space separated list of temperatures, e.g. 5 7 9 11 13]
To show lines, instead of areas, pick Wireframe
To show the edges of the domain,
Add:Subset:Bathymetry_Mesh:Single(black):Wireframe:Line width=2:No Legend

Isosurfaces of specific temperatures
Add:Contour:temp

Add:Pseudocolor:temp
temp:Operators:Transforms:Transform:Scale:Z=1000
temp:Operators:Slicing:Isosurface:Select by values:Space separated list of values:Apply

Extract 1D line
Add:Pseudocolor:temp
Controls:Query:Queries=Pick:Variables*:Choose Pick location**:Do Time Query:Query button
*The default variable is the variable of the most recent plot in the current window. You can use dropdown=scalars,temp, or type temp. You can plot multiple variables by entering a list of space separated values, e.g. "ZOO DIA GRE".
**For Choose Pick Location, I recommend: Pick using domain and element ID, then choose 'Node' from coordsn.csv.
For example, choose Node Id 4 which is (x, y, lon,lat, depth):
`674371.3125 5074093.5 -84.75620185 45.79836785 18.80982971`
After extracting a 'Pick', a second window pops up, and a 'Curve' is shown. Change 'Active window' to '2'. Click on Curve to modify properties, such as point symbol and size, line width and color.
The Time Query results are in **timesteps**, and are from 0 to [total number of timesteps - 1]. We need to convert that to real time for meaningful plots. Output is in **Cycles**.**
Warning: For a long time series, Choose a 'Stride', so it has less to calculate. If you don't do that, it might freeze your computer, or at least freeze VisIt.
***Look up later: Why are these different?*

Appearance of plots
Controls:Annotation

- General: remove Database, user information
- 2D/3D: show/hide axes, triad, bounding box, show/remove title, labels, tickmarks
- Colors: for fancy looking plots/animations
- Objects: can change position/appearance of legends, etc. Add a time slider or text, then can operate on with Python (change time units, etc.)

Save images
File:Set save options
File:Save window
Notes:
Set save options - you must do this, or else VisIt will save whatever is default.
Define filename or filename prefix
Check Family - numbers will be added to the prefix to avoid overwrite, e.g. third file written will be called [filename prefix]0002.png
Uncheck 'Output files to current directory" and choose the output directory.
File type: default PNG is good, use JPGs if you need smaller files.
I would leave the rest as-is unless you know why not.

Save the session (which plots are shown and how they look); to reopen a session, and to reopen a session with a different data set

File:Save session as
File:Restore session
File:Resource session with sources

To plot with experimental data, make .ult file containing

```
#header
x1 y1
x2 y2
…
xn yn
```

Then do
Open:Add:Curve

Make the session file for the demo

Open mi_*.nc database
Plot: pseudocolor temp, transform:scale:z=1000, Tilt axis
Annotations:
Remove: Database, User information, apply
Objects:
Add Time Slider, name it "Slider"...this is in the Python script, so it must be called "Slider".
Save session as: call it pseudocolor_with_slider
Close and reopen to test
File:Restore session:psudocolor_with_slider
Copy/paste mjd_callback.py to Commands window and hit Execute. Time will change upon first click to advance time.

To get a vector plot, make a new variable

# Plot a vector expression variable, example
DefineVectorExpression("myvec", "{u,v}")
AddPlot("Vector", "myvec")
DrawPlots()

Example of plotting observations against model output

Observation data:

Average Surface Water Temperature data for Lake Michigan, 2010:
Average GLSEA Surface Water Temperature Data (1995 - Present , csv data files in degree Celsius or degree Fahrenheit) (Download CSV for Celsius.)

Using Excel, open the CSV, and grab first column (days) and column from 2010 and copy to another spreadsheet.
Convert days to **Cycles**...Total Cycles = 1576800, 4320 cycles/day
Add a header of #Temperature.
Export as .prn (space separated columns), called ave_surf_temp_2010.prn. Rename to ave_surf_temp_2010.ult.

Model data, surface temperature:
Add:Pseudocolor:temp
Control:Subset:Sigma_Layers:[0]
Controls:Query:Queries=Pick variable temp
Choosing the node:
In VisIt...for surface temperature, I picked a lat/lon in the middle of LM, and the first layer - node 3725:

3725	447771.3125	4880996.5	-87.65232057	44.08028213	4.223293781

Time queries are by **timesteps**. There are 8760 timesteps. Do a time query with stride 8760/365=24 to get daily points. Do not try to extract a query for each timestep, I did that and had to force-quit visit. Extracting 365 took a while. The results are plotted to Window 2.

Extraction is using timesteps, but the results are in **Cycles**.

Plot both model and observations:
Make Window 2 the active window. That is where the extracted data from VisIt is plotted. Add the observation by adding a curve plot of the ULT file.
File:Open file:ave_surf_temp_2010.ult
Add:Curve:Temperature:Draw

Scripts

Convert Time from MJD

Add time conversion from MJD:
Control:Commands:Paste this in the window: (mjd_callback.py, below and in VISIT/SCRIPTS)
There are 8 tabs in the Commands window. They are saved to ~/.visit as script[1-8].py, e.g. ~/.visit/script1.py
Instead of copy/paste mjd_callback.py to the Commands window, you can do
cp mjd_callback.py ~/.visit/script1.py
Then that will be in the Command window, and you can hit 'Execute'

mjd_callback.py (better to download to avoid cut/paste errors, in directory VISIT/SCRIPTS)

```
# Callback to update timestamp for every new timestep
# Place this python code in ~/.visit/visitrc before starting VisIt.
# Four separate callbacks are registered for forward step, backward step and slider set
# of the GUI's VCR controls. There is a bit of a problem with the 'play' button mode.
# There is NO callback action associated with each step of the 'play' mode. So, handling
# that requires using the WidnowInformation callback. VisIt will wind up queueing up all
# the callbacks until playing is stopped.

# line 42: start time of simulation needs to be changed accordingly.

import datetime
import calendar

def UpdateTimestamp(arg):
   global lastState
   global t_start
   currentState = arg.timeSliderCurrentStates[0]
   if currentState != lastState:
      try:
         m = GetMetaData(arg.activeSource)
         # time:format = "modified julian day (MJD)", so multiply by seconds in a day
         tcur = m.times[currentState]*86400.  + t_start
         ts = datetime.datetime.utcfromtimestamp(tcur).strftime('%Y-%m-%d %H:%M:%S')
         timestamp = "Time: " + ts + " GMT"
         annot_obj = GetAnnotationObject("Slider") # find out names by GetAnnotationObjectNames()
         annot_obj.text =  timestamp
         annot_obj.position = (0.03, 0.94)
         annot_obj.height = 0.05
         annot_obj.fontBold = 1
         lastState = currentState
      except:
         return

def onSetTimeSliderState0():
   UpdateTimestamp(GetWindowInformation())

def onSetTimeSliderState1(timeState):
   UpdateTimestamp(GetWindowInformation())

def onWindowInformation(arg):
```

```
    UpdateTimestamp(arg)

#time:units = "days since 1858-11-17 00:00:00" ;
#               time:format = "modified julian day (MJD)" ;
t_start = calendar.timegm(datetime.datetime(1858, 11, 17, 0, 0, 0).timetuple())
lastState = -1
RegisterCallback("SetTimeSliderStateRPC", onSetTimeSliderState1)
RegisterCallback("TimeSliderNextStateRPC", onSetTimeSliderState0)
RegisterCallback("TimeSliderPreviousStateRPC", onSetTimeSliderState0)
RegisterCallback("WindowInformation", onWindowInformation)
```

----Below this line are my scratchy notes, please ignore----------

Directory with Mark's 1 year of output:
/Users/lisalowe/ORD/Everything/mi_gem_archive/041820163/mi_*.nc

Show Commands/CLI

temp -- Min = 3.98727 (node 115030 at coord <490912, 4.713e+06, -147.606>)
temp -- Max = 16.0773 (node 113526 at coord <466587, 4.98615e+06, -32.4144>)

PickByNode(curve_plot_type=0, do_time=1, domain=0, element=1000, preserve_coord=0, vars=("temp"))

```
>>> PickByNode(curve_plot_type=0, do_time=0, domain=0, element=1000, preserve_coor
{'pick_letter': 'C', 'incident_zones': (1819, 1820, 1902, 1903, 1904, 1905), 'temp': 12.84948444366455, 'point': (622070.1875,
5016153.5, -2.0225820541381836), 'timestep': 0, 'filename': 'mi_0011_1-720x180.nc', 'node_id': 1000}
>>> GetPickOutput()
'\nC:  mi_0011_1-720x180.nc  timestep 0\nSigmaLayer_Mesh \nPoint: <622070, 5.01615e+06, -2.02258>\nNode:
1000\nIncident Zones:  1819  1820  1902  1903  1904  1905  \ntemp:  <nodal> = 12.8495\n\n'
```

Lineout mode:
https://visit-sphinx-github-user-manual.readthedocs.io/en/latest/Quantitative/Lineout.html

The actual number of nodes is 115,900.

6,000 nodes

Lineout
548466.63 4793653 -0
548466.63 4793653 -300

548466.63 4793653 -10
561848.5 4756940.5 -10