PyVista QUIC-Fire Plotting Guide

Overview:

Use: The script PlotQFpyVista.py uses a QUIC-Fire (QF) projects folder in conjunction with a simple plain-text input file to generate VTK generated movies (similar to ParaView). Python3 was used to develop the script, I'm unsure of compatibility with Python 2.7.

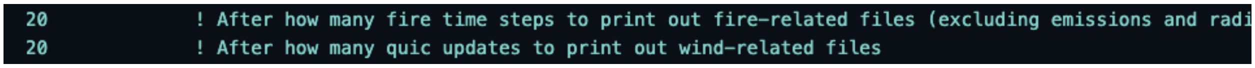
General Call: python3 PlotQFpyVista.py *path/to/scriptInputFile.inp*
Example Call: python3 PlotQFpyVista.py VistaAargs.inp

Dependencies: pyVista, numpy

PyVista Installation Instructions

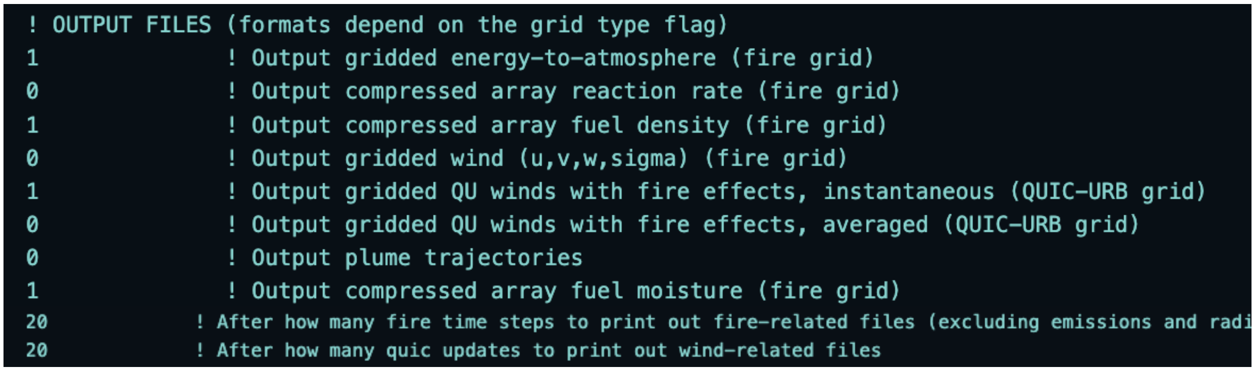
PyVista Camera/Plotting Controls

The script references the QF input files from a project to inform it on what to plot. This includes the time domain, time spacing of output, wind grid defined by qu_simparams.inp, and more. This means that you can have control over what the script will produce both by modifying the QF standard input files or the script specific input file. (Trick: If you want to preview what an animation will look like without drawing every timestep you can increase the QUIC_fire.inp 'timesteps to print out' line to a multiple of your original output period. (i.e. if you changed from 20->40 you would visualize half of your output) The total simulation time argument may also be reduced for visualizing less frames.)



```
20           ! After how many fire time steps to print out fire-related files (excluding emissions and radi
20           ! After how many quic updates to print out wind-related files
```

When the script reads QUIC_fire.inp it looks for data based on the non-zero flags of the input file. If you would like to exclude data from the visualization simply switch the flag to 0 in your input file and the script will exclude that data. Only the quantities shown in the below image with a '1' are supported for visualization as of now.



```
! OUTPUT FILES (formats depend on the grid type flag)
1             ! Output gridded energy-to-atmosphere (fire grid)
0             ! Output compressed array reaction rate (fire grid)
1             ! Output compressed array fuel density (fire grid)
0             ! Output gridded wind (u,v,w,sigma) (fire grid)
1             ! Output gridded QU winds with fire effects, instantaneous (QUIC-URB grid)
0             ! Output gridded QU winds with fire effects, averaged (QUIC-URB grid)
0             ! Output plume trajectories
1             ! Output compressed array fuel moisture (fire grid)
20           ! After how many fire time steps to print out fire-related files (excluding emissions and radi
20           ! After how many quic updates to print out wind-related files
```

PlotQFpyVista.py Input File:

The input file can be named anything as long as it's saved as plain-text and the correct name is given in the command line argument. Input variables and there values must be separated by an '=' without any spaces. The code that parses the input file is very simple and is separating based on the '=' so NO SPACES. All arguments are not needed for the input file, almost all arguments have a standard default, although I'm sure it's not completely water tight. All arguments, except paths and names, are brought to upper case letters so most are not case sensitive. (Note: To disable an input parameter from being registered without having to delete it from the input file just add a character to the beginning as all parameters are keyworded. i.e. *Plotting=sub* → *#Plotting=sub*)

Argument Descriptions:

-projectdir- REQUIRED Name of project folder or path to project folder to be visualized. (Note: '/' required at end)

Example: projectdir=Holtz1000s_noSORopt_slowTime/

-Plotting- **Options: SUB, ALL, IND (Defaults ALL)** Plotting style.

-SUB- Generates a single window with 4 sub-windows showing fuels, moisture, energy, and winds in their own quadrant. Great for showing all aspects of the fire.

-ALL- Generates a single window and combines all data imported into a single graphic. Makes more real-world looking fire but at the cost of obscuring a lot of information.

-IND- Generates a video for each quantity included and titles them with the quantity. If all four flags are '1' then 4 videos will be generated.

Example: Plotting=SUB

-Interactive- **Options: 0 or 1 (Defaults 1)**

0: Plot window is hidden from user while graphics are generated. This means the user will have no control over the camera other than what is set from the input files.

1: Plot window is completely interactive. Window controls fully engaged (see pyVista controls).

-legacyVersion- **Options: 0 or 1 (Defaults 0)**

0: Uses current file structure standard where QF outputs are located in 'projectFolder/Output/'

1: Uses legacy file structure where QF outputs are located in 'projectFolder/'

-Framerate- **Options: Any Integer (Defaults 1)** Sets the framerate of the video generated. Framerate=1 equates to 1 frame per second.

-Title- Name of title of video. Extension controls video type made. '.mp4' is suggested as they play on most OS without much effort and are also quite small storage-wise.

Example: title=TestingVideo.mp4

-Outdir- Relative path to where the video should be saved. Should generate the folder if it doesn't exist. (Note: '/' required at end)

-Quivers- **Options: 0 or 1 (Defaults 0)** Adds vectors of surface winds.

-quiverScale- **Options: Real [0,inf] (Defaults 1.0)** Value to scale vectors by.

-quiverDensity- **Options: Real [0,1] (Defaults 0.5)** Sets what percentage of surface wind vectors to use. *quiverDensity=1.0* means that all winds will be shown as a vector. *quiverDensity=0.1* means every 10th wind value at the surface will be shown.

-windSlices- **Options: 0 or 1 (Defaults 0)** Adds cross-section slices through wind field. Number of slices, orientatizon, winds sampled, and opacity are all controlled by their own input arguments.

-sliceAxis- **Options: X, Y, or Z (Defaults Y)** Axis to distribute slices along. Slices will be perpendicular to axis chosen.

-sliceWind- **Options: U, V, or W (Defaults W)** Desired wind to be sampled.

-nSlices- **Any Integer (Defaults 5)** Number of slices along axis to be visualized. When 1 is chosen it unfortunately sets it at the edge. Future implementation will allow for direct placement of slices. This is what we have for now.

-sliceOpacity- **Real [0,1] (Defaults 0.15)** Sets the opacity for the slices.

-windBarNormalized- **Options: Auto, Custom, Free, None (Defaults Free)** Adjusts the colorbar range for wind plots.

Auto: Max and min wind speeds of complete time series are determined and colorbar range is set to these values.

Custom: Uses user defined *windbarBounds* parameter to set colorbar range.

Free: Colorbar range will adjust for each slice dependent on the min and max wind speeds of the current frame.

None: Removes the colorbar from the wind plot.

-windBarBounds- **Range: [lowerBound,upperBound]** If *windBarNormalized=Custom* then *windBarBounds* value is grabbed and colorbar range is set to these values.

Example: windBarBounds=[0.0,13.5]

-xyView- **Options: 0 or 1 (Defaults 0)** Fixes the camera to an tops-down vertical projection view. This removes all 3D parallax effects. Useful for showing spatially accurate plots of behavior. (Note: Cannot pitch camera while XY-view is active)

-fireTracking- **Options: 0 or 1 (Defaults 0)** Automatically controls camera view to focus on approximately the center of mass (CoM) of most recently burned fuels. Underlying method keeps track of a history of CoM points and averages their positions (helps remove some of the jitteriness of the camera). With *interaction=1* the user can still move the camera but before a frame is written to the movie the camera will adjust to the same offset position (i.e. your camera view will not be preserved, it will always produce the same movie).

> **-trackingOffset-** **Vector: <xOffset,yOffset,zOffset> (Defaults <200.0,200.0,300.0>)** Camera position offset from *fireTracking* focal point. Camera always points at focus, this vector sets how far away and from what side the camera is looking.
>
> *Example: trackingOffset=<200.0,200.0,300.0>*
>
> **-trackHistory-** **Any Integer > 0 (Defaults 10)** Number of previous CoM points to keep and average over. Value of 1 results in camera focusing on most recently consumed fuel. The longer the history the smoother the camera movements in theory.

Explicit Camera Controls – These explicit camera controls set the camera in a static position. Useful for creating movies with *interaction=0*. Helps with larger domains where visualizing each frame to the screen takes longer and removes need for user to hit ‘q’ key to advance frames and position camera. (Note: *fireTracking* also helps in the same way when *interaction=0*)
-camPosition- **Vector: <xPos,yPos,zPos> (Defaults <200.0,200.0,300.0>)** Camera position in space. (Note: this does not point the camera, simply where the camera is located)
-camFocus- **Vector: <xFoc,yFoc,zFoc> (Defaults <200.0,200.0,300.0>)** Camera focus location. This is the position the camera will point to.
-camNormal- **Vector: <xNorm,yNorm,zNorm> (Defaults <0.0, 0.0,1.0>)** Sets cameras ‘upward’ direction. This allows for rotation of the camera.

-camOrbit- **Options: 0 or 1 (Defaults 0)** Produces an opening and ending animation to video where camera revolves around domain. Useful for showing beginning and end state of domain when some features/impacts of fire may be obscured. Adds significant size to videos as it takes 36 frames to complete each orbit animation. (Note: Unfortunately this functionality can only work if *interaction=1* meaning that if a video with orbit is wanted the user must hit ‘q’ to advance all frames to complete the movie.)