Assignment #1: 2D Scalar Field Visualization

Due on September 18th, before midnight

Goals:

The goals of this assignment include (1) get familiar with VTK based visualization environment, (2) understand the principles and design of different color-coding schemes, and (3) implement color plots and iso-contouring using VTK for the visualization of a few 2D data sets.

A skeleton code in Python is provided to you.

Requirements of Submission:

You should submit your source code and report **in a single .zip file** via Canvas before the deadline.

Please name your submission as follows:
*Assign1_**FirstName_LastName**.zip* (or another compressed format)

Your report should include your answers to the writing questions and high-quality images captured from the individual visualization tasks listed below.

Important Notice on the Use of ChatGPT in Programming Assignments:

While the utilization of ChatGPT or similar large language models is permitted for programming assignments, it is not allowed during exams. We urge students to exhaust their understanding and attempt solutions independently before resorting to these tools. Remember: the true value of this course is derived from the challenges you overcome, and the knowledge you gain in the process.

Transparency is crucial: If you choose to use ChatGPT or any other external resources, it is mandatory to clearly indicate so in your assignment report. You will not be penalized if you use ChatGPT. However, if you failed to report it and was caught, it would be considered as plagiarism and your score will be zero. Note that it was prohibited to borrow codes from students who already took this course.

Tasks:

1. Writing questions (30 points, 10 points for each question)

1.1 What issues need to be considered when choosing a proper color-coding scheme to generate effective color plots?

1.2 Given the HSV color space, which channel(s) is suitable to encode quantitative information (e.g., scalar fields)? Why?

1.3 Provide a **complete** pseudo-code to describe Marching Squares algorithm, including the handling of quads with different numbers of intersections.

2. Generate smooth color plots (30 points)

Implement three color-coding schemes: (1) blue-white-red (BWR), (2) heatmap, and (3) gray scale. Please refer to the lecture slides for the design of the respective color transfer functions for these color-coding schemes.

Suggestion: You can map scalar values into RGBA colors by using a vtk lookup table. The table is built from a discrete linear ramp of each value. The first color value corresponds to the minimum scalar value while the last color value is mapped to the maximum scalar value. Assume that you can read the input file via one of the vtk reader classes (e.g., `vtkDataSetReader`, `vtkPolyDataReader`) and obtain a geometry, e.g., a `vtkPolyData` object. In the next step, you will need to create a `vtkPolyDataMapper` object and set the `vtkPolyData` object as its input data. The vtk mapper is responsible for pushing the geometry into the graphics library. It may also do color mapping if scalar attributes are provided.

Here is the sample code to create a vtk color lookup with 256 values.

```
lut = vtk.vtkLookupTable() # Initialize the vtk lookup table
nc = 256 #the size of the table - increase this number to obtain more precise color map
lut.SetNumberOfTableValues(nc)
lut.Build()
```

To set a color value in the look-up table, you can use ***`lut.SetTableValue(index,rgba_value)`***

Here is the sample code for rainbow color mapping:

```
# The min scalar value is mapped to 0, and the max value is mapped to 1
sMin = 0.
sMax = 1.

hsv = [0.0,1.0,1.0]

for i in range(0, nc):
        s = float(i) / nc #uniformlly interpolate the scalar values
        hsv[0] = 240. - 240. *(s-sMin)/(sMax-sMin) #rainbow color calculation
        rgba = hsvRgb(hsv) # convert hsv to rgb
        rgba.append(1.0)  # set alpha (or opacity) channel
        lut.SetTableValue(i, *rgba)
      # you can use lut.SetTableValue(i, rgba[0], rgba[1], rgba[2], 1.0)
```

Finally, you can assign the color look-up table to the `vtkPolyDataMapper` as follows:

```
vtk_poly_mapper.SetScalarModeToUsePointData()
vtk_poly_mapper.SetLookupTable(lut)
vtk_poly_mapper.SetScalarRange(min_scalar, max_scalar) #Specify range in terms of scalar minimum and maximum (smin,smax). These values are used to map scalars into lookup table
vtk_poly_mapper.SelectColorArray(scalar_field) #set the name of scalar field
```

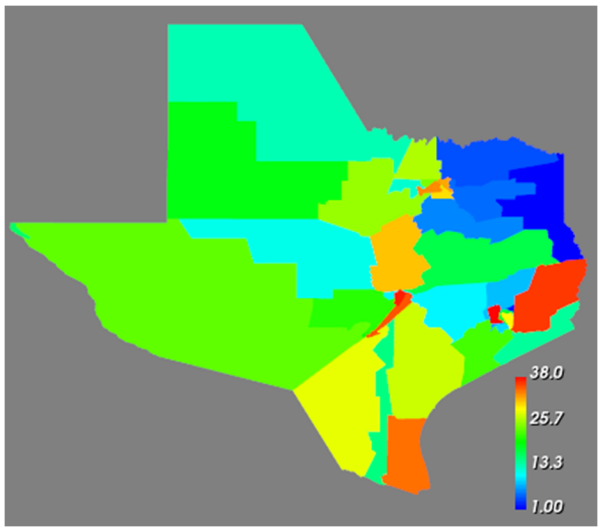
Additional suggestions: please consider adding an interface that allows the users to adjust `smin` and `smax` values to counter the dynamic range issue for some data sets (e.g., torus.vtk).

3. Discrete Color Mapping of Texas Districts (10 points)

When representing scientific data using VTK, color mapping schemes play a critical role in visual clarity. In this assignment, not only will you be dealing with continuous color mapping schemes but also with categorical data. Specifically, you will be visualizing the gerrymandered districts of Texas ("`texas.vtk`").

Your primary challenge lies in **designing a discrete color scheme**. This color scheme must ensure that no two districts are colored similarly to facilitate clear distinction. To provide context, an example screenshot is attached below which employs the default rainbow color scheme. As you will observe, this scheme results in multiple districts having almost indistinguishable colors, making it inappropriate for our purpose. Note that this color scheme is only applicable to the Texas district data set, don't worry about applying it to the other data sets.

1. Analyze the provided screenshot and discern the limitations of the rainbow color scheme in visualizing Texas's districts.
2. Design and implement a new discrete color scheme that avoids these pitfalls so that **no two adjacent districts/counties have similar colors.**
3. **Capture and include a screenshot** of your visualization using this new color scheme in the assignment report.



4. Compute and visualize iso-contour (30 points, 15 points for each sub-task)

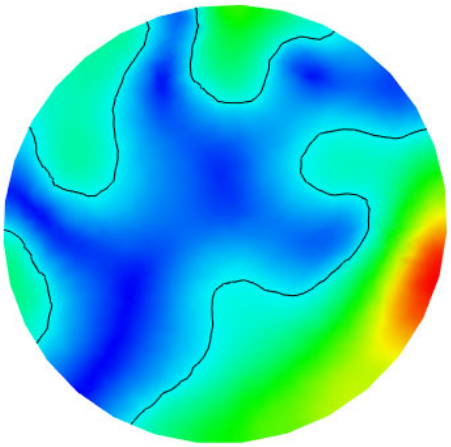
This task includes the following two sub-tasks. Getting the first sub-task done correctly will help you complete the second sub-task.

4.1 Compute one iso-contour with the user input iso-value

Build on the program you have completed so far, complete the iso-contour computation and visualization given a specific iso-value.

Hints:

You can use `vtkContourFilter` filter and its `SetValue(0, [scalar value of the contour])` function for this task. You need to call this filter before adding the mapper following the VTK pipeline. A sample visualization can be found to the right.



4.2 Compute K iso-contours with the user input integer K (>=1)

Now you know how to compute and visualize one iso-contour with a specific iso-value, we can move to the implementation of K iso-contours. One simple way to achieve that is after the user input the number of contours (K) they want to see; you can evenly subdivide the data range and get K iso-values within the range.

Hints:

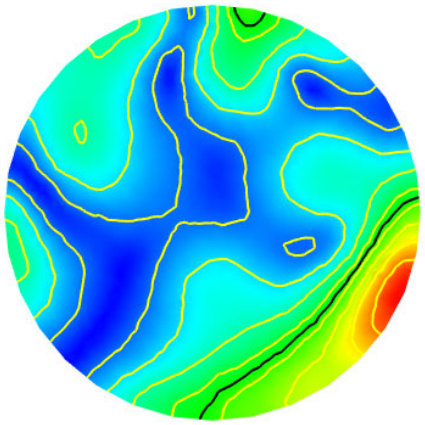
You can use the `vtkContourFilter` and its member function `GenerateValues([number of contours], [scalar data range])`. The scalar data range can be obtained using:

```
scalar_range = vtk_reader.GetOutput().GetScalarRange()
```

or `scalar_range = [min_scalar, max_scalar]`

`min_scalar, max_scalar` were already computed previously in the skeleton code.

An example output for the diesel_field1.vtk is shown to the right. The black iso-contour is computed by Task 4.1, and the yellow contours were computed the output of Task 4.2. They should be clearly distinguishable!



NOTE THAT you need to **replace** "YOUR NAME" in the following line near the end of the main function **with your actual name**. **Without that, your submission will NOT be accepted!**

```
vtk_render_window.SetWindowName('COSC 6344 Visualization - Assignment 1, YOUR NAME')
```

Adding a GUI interface for the above tasks

Utilize PyQt to develop a complete visualization program with GUI integrating the above tasks. The program should allow the user to interactively select a .vtk file (with 2D scalar field) to load and visualize. After the program successfully loads the selected .vtk file, the color plot of the 2D scalar field will be shown on the vtk view, while the user can choose from different color-coding schemes provided. For the iso-contouring, the GUI should provide a widget for the user to input different iso-values, then the program will update the iso-contour and re-render it in the vtk view. The GUI should also provide a widget that enables the user to select the number (K) of iso-contours to compute and visualize. Additional widgets may be added to facilitate other user interactions (Be creative and thoughtful!)

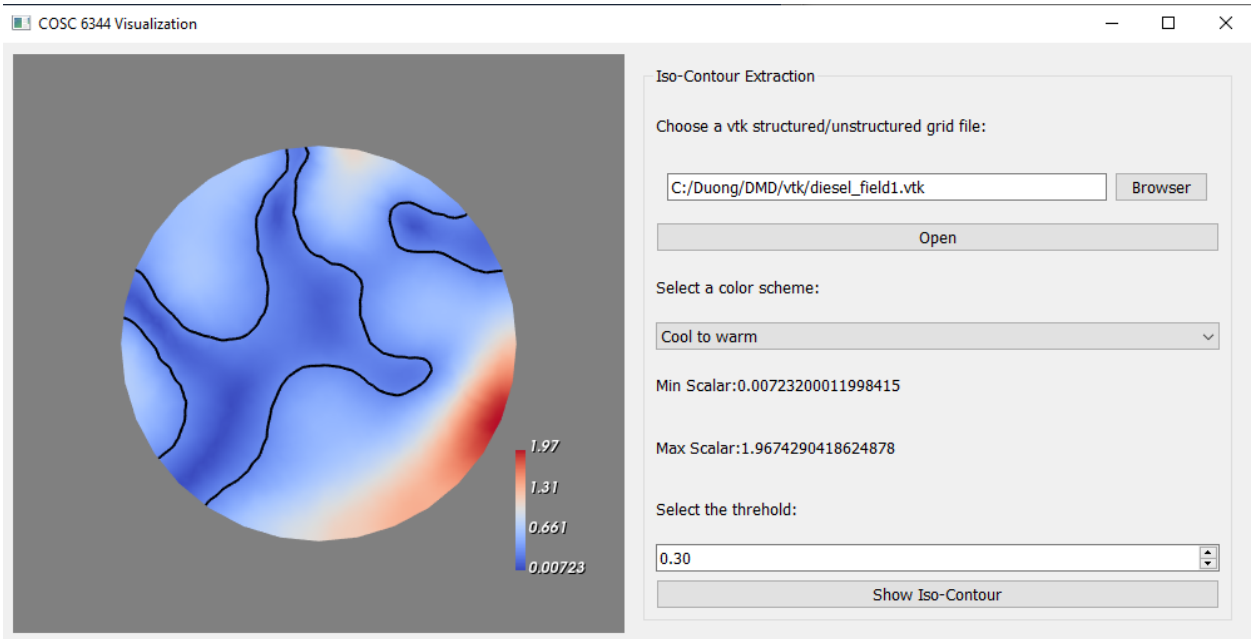
You can find a list of official PyQT widgets in the following link

https://doc.qt.io/qtforpython/PySide2/QtWidgets/index.html#module-PySide2.QtWidgets

The following tutorial is a good starting point for you to get familiar with PyQt design

https://www.tutorialspoint.com/pyqt/pyqt_qcombobox_widget.htm

The following pictures shows a sample GUI interface created by using QT. Note it does not have the interface for specifying different number of contours.



The sample PyQT code includes additional widgets for you to play with.

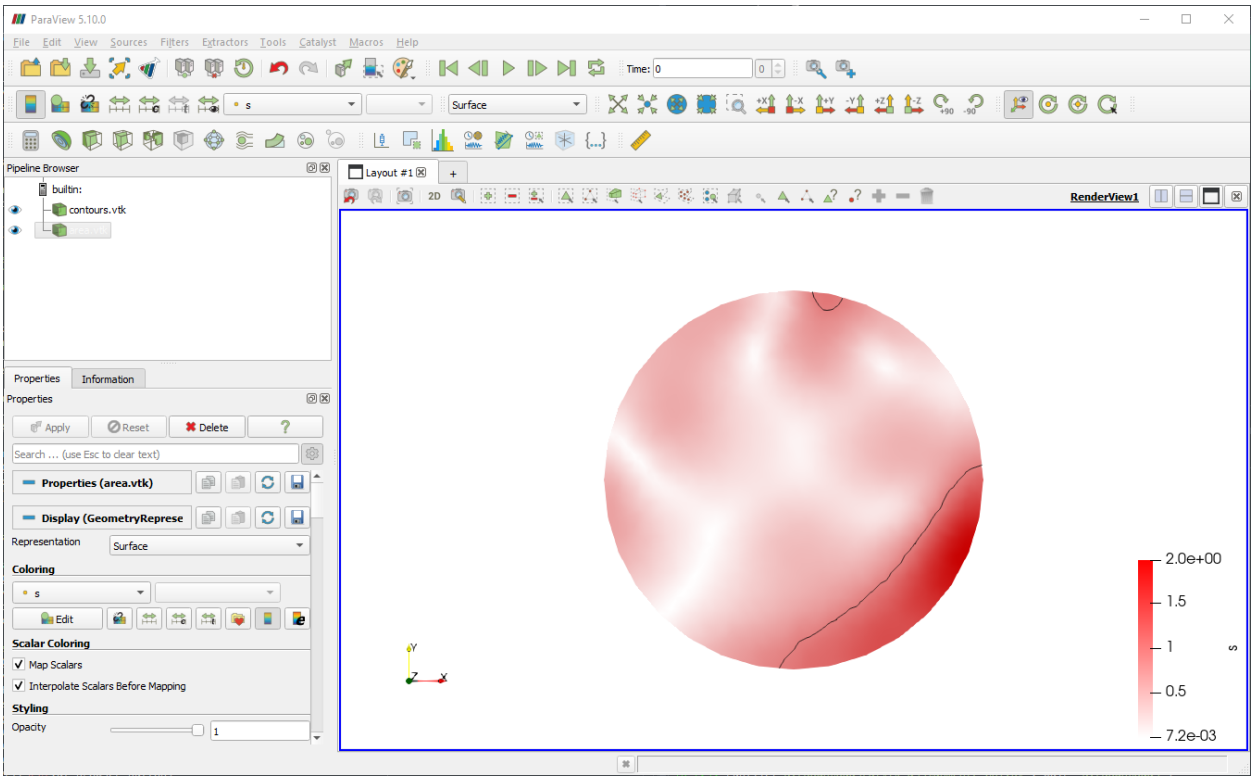
5. Export model file(s) (bonus 5 points)

It's important that you know how to export your visualization data to be viewable on other visualization software. This task will ask you to **export the computed iso contour lines** into a model file that can be

viewed in another visualization program. While you can use whatever file type and program of choice, we recommend exporting using the vtkPolyDataWriter class and reading the file using Paraview.

The specific steps for exporting the iso contour lines using vtkPolyDataWriter are listed as comments in the provided skeleton code, you first create a vtkPolyDataWriter object, set it's input as the output of the iso contour mapper, set the file name, and then write().

Using Paraview, you can read in the original data set file along with the exported iso contour vtk file and visualize them together as shown here:



To complete this task**, submit the screenshot showing the iso contour lines drawn on top of the original dataset, in any other visualization program such as Paraview.**

Grading rubric:

Tasks	*Total points*
1	30
2	30
3	10
4	30
5 (bonus)	*5*

Suggestions:

Please refer to the VTK tutorial (https://vtk.org/Wiki/VTK/Tutorials) and Python examples (https://lorensen.github.io/VTKExamples/site/Python/) for more information and examples.

Have fun!