# Assignment #2: 2D Scalar Field Visualization

**Due on September 23rd, before midnight**

## Goals and Requirements:

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

A skeleton code in Python will be provided to you to help you get started.

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

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

## Tasks:

### 1. Writing questions (40 points, 10 points for each question)

**1.1** Describe the generation of blue-white-red (**BWR**) color scheme using **HSV** color space using pseudo-code

**1.2** Describe the **heatmap** generation using **RGB** color space with pseudo-code

**1.3** Explain the limitation or issue of the Rainbow color scheme

**1.4** Provide a **complete** pseudo-code to describe Marching Squares algorithm, including the handling of different number of intersections for a quad.

### 2. Generate color plots (30 points, required)

Implement two different color-coding schemes, namely blue-white-red (BWR) and heatmap by using the algorithms in Task 1. The BWR will be turned on when the user presses key '1', and the heatmap is on when key '2' is pressed. The default color scheme is rainbow, which can be turned on by pressing key '0'. [Note that you can add other color scales, but the above heatmap and BWR color scales MUST be implemented! Otherwise, you will not receive these 30 points!]

Your program should be run in the following format

```
> python your_program.py
```

The program will ask for the input file name (including full directory) and load the data and return a message of successfully loaded or not. If loaded successfully, your program will get the data range from the data file (i.e., the minimum and maximum scalar values) and return this range to the user in the console. [Note that not returning the data range information will lead to the deduction of 5 points.]

**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
```

## 3. 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.

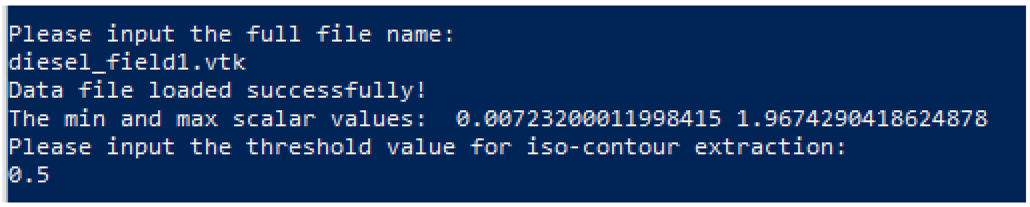
### 3.1 Compute one iso-contour with the user input iso-value

*Build on top of the program you complete in task 2*, complete the iso-contour computation and visualization given a specific iso-value.

Your program should be run in the following format

```
> python your_program.py
```

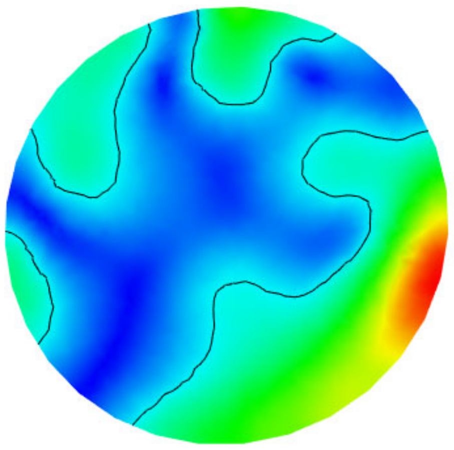
The following is an example session of the program that gives you an idea of how the interaction may look like. Note that you need not use the exact format as follows.



```
Please input the full file name:
diesel_field1.vtk
Data file loaded successfully!
The min and max scalar values:  0.00723200011998415 1.9674290418624878
Please input the threshold value for iso-contour extraction:
0.5
```

**Hints:**

You can use `vtkContourFilter` filter and its `SetValue(0, [scalar value of the contour])` 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.



### 3.2 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.

Your program should be run in the following format

```
> python your_program.py
```

In addition to the user interaction provided in the previous task, add an input so that the user can input an integer to specify the number of contours she/he wants to see

```
> Please input the numbers of contours to extract:
> 10
```

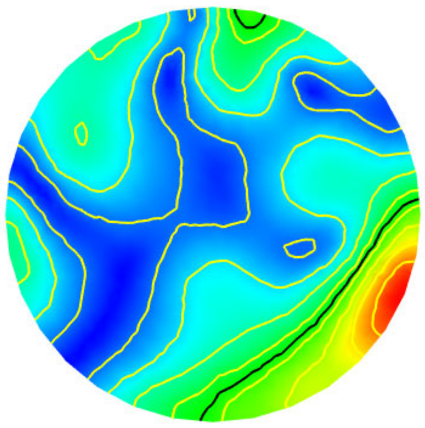
**Hints:**

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 3.2, and the yellow contours were computed the output of Task 3.3. 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
2, YOUR NAME')
```

## 4. Adding a GUI interface (bonus/optional, 5 points)

Install PyQt and use 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-value, then the program will update the iso-contour and re-renders 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!)

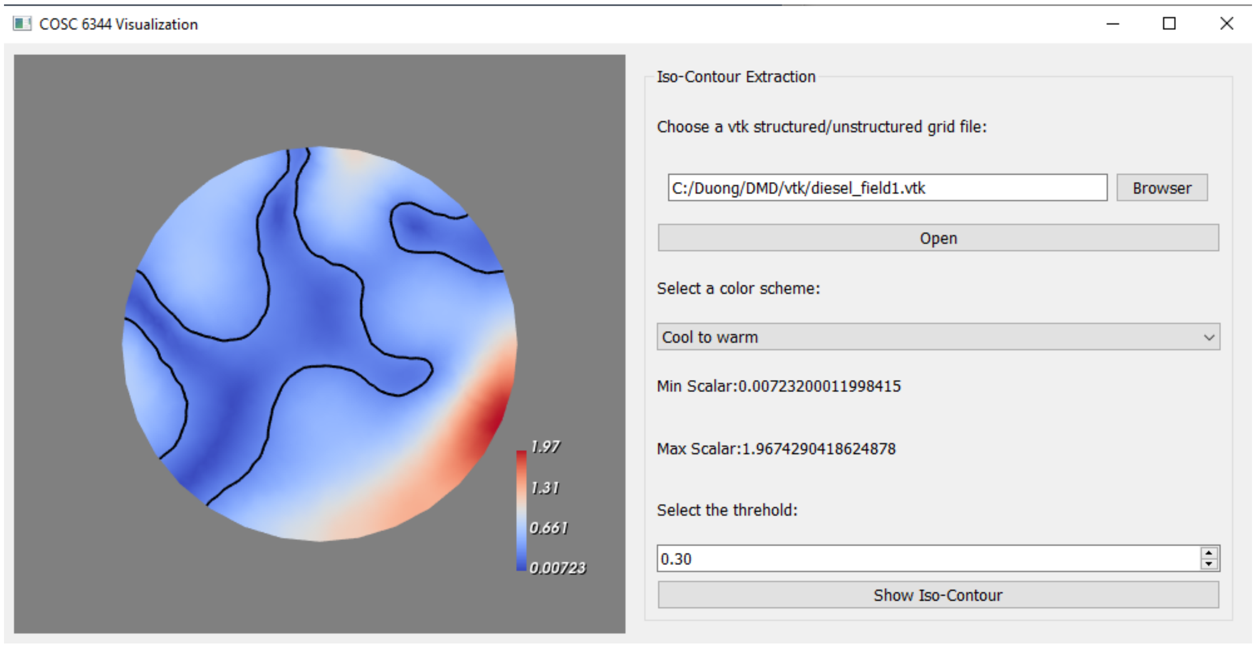
You can find a list of official PyQT widgets in the following link

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:



Choose a vtk structured/unstructured grid file:

C:/Duong/DMD/vtk/diesel_field1.vtk

Select a color scheme:

Cool to warm

Min Scalar:0.00723200011998415

Max Scalar:1.9674290418624878

Select the threhold:

0.30

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

## Grading rubric:

| *Tasks* | *Total points* |
|---:|---:|
| 1 | 40 |
| 2 | 30 |
| 3 | 30 |
| *4 (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!