Visualization HW1
Seren Lowy – 2023/9/18

1.1 Issues in choosing color coding schemes for plots

One essential issue in color coding is data visibility. We want the changes in data to be easy for viewers to notice, so we need to consider what color schemes will not hide essential features of the data. For example, in quantitative data, extreme values should use colors which are clearly different from opposite extremes and mediocre values.

Another issue is color semantics. When coding for data using colors, it is important not to encourage misinterpretation. We should respect the traditional meanings of colors and avoid representing a specific meaning in our chart using a color which has a conflicting traditional meaning. For example, since a red hue has traditional meanings like “profit”, “heat”, or “danger”, it should not be used to represent “cold” on a temperature scale or “low incidence” on a map of epidemic rate. Along with hue, degrees of saturation or luminance can have traditional meanings. Colors with the lowest saturation should represent data values with the least significance, typically those closest to zero, because high saturation traditionally tells viewers to pay attention while low saturation means that there is nothing special. Placing high saturation in the middle of a divergence scale or at the low end of a sequential scale would be confusing and tiring because it would appear too often on most charts and draw attention to mediocre values, while making hues less distinct and obscuring visibility for notable values.

An issue of color schemes as a whole is practicality. We should avoid making the color scheme noisy or fastidious. To prevent noisiness, scales for ordinal or quantitative data should use hues that vary in order around the color wheel. Saturation and luminance should typically increase/decrease monotonically, or less often parabolically, with a maximum of one peak or trough which is placed as near as possible to the middle of the scale. Preventing fastidiousness means that if hues are used to encode meaning at all, they should be clearly distinguishable from each other. This is especially critical for categorical data, since viewers will likely judge wrong when having to interpret an element on the chart based on which of several minutely similar color categories it belongs to. As a corollary, in divergence scales for ordinal or quantitative data, selecting closely similar colors for the extremes will cause confusion and disorientation. However, according to the second Gestalt principle (similarity), colors used to encode related categories may and should be somewhat more similar to each other, while still different enough to be identified where they appear on the chart. [1]

Sources: [1] lecture notes

1.2 Quantitative color encoding with HSV

Saturation and value (luminance) are the best choices for representing the smaller gradations or continuous variations present in quantitative data, because our human eyes are receptive to even subtle changes in these color space dimensions. Since our eyes have difficulty perceiving smaller changes in hue, especially within certain segments of the hue range such as dark blue and blue-green hues, it is better to avoid using hue to capture continuous variations in data. [1]

Sources: [1] lecture notes

1.3 Marching Squares algorithm

The marching squares algorithm starts from a discretely sampled version of data in a grid format. It outputs a vectorized geometric map of isocontours, where each isocontour is a set of one or more

closed polygons which do not intersect with each other or with any polygons from the other isocontours. Each isocontour is generated from one thresholded binarization of the source data. [2]

To generate one isocontour[1, 2]:

1. Accept a real number known as isovalue. This will be used to threshold the input data.
2. Sample the input data values over a discrete grid.
3. Compute an isovalue threshold from the sample grid.
    a. Sample value < isovalue → output bit 0
    b. Sample value < isovalue → output bit 1
4. For every sample in the grid, use the threshold to classify the configuration of its neighborhood.
    a. The neighborhood we want to consider around each cell is the cell itself plus the cells below by 1, right by 1, and diagonally below right by 1x1. If a sample does not have some of these neighbors, assume zeros for the missing neighbors.
    b. Compute the configuration id for a neighborhood by concatenating the bits from the four cells in the neighborhood and mapping to a half-byte integer. Follow a consistent order of traversing each neighborhood, such as clockwise. For example, if all cells are 0, the configuration is $0000_{two} = 0_{ten}$. If the upper left and upper right cells are 1, the configuration is $1100_{two} = 12_{ten}$.
5. Map the configuration ids to abstract drawings, topologically defined over a square area.
    a. ID# (0, 15): a drawing with no edges and a uniform opacity level.
        (0): transparent
        (15): opaque
    b. ID# (1, 2, 4, 8): a single edge spanning two adjacent sides of the bounding square.
        (1): opaque in lower left corner, transparent elsewhere
        (2): opaque in lower right corner, transparent elsewhere
        (4): opaque in upper right corner, transparent elsewhere
        (8): opaque in upper left corner, transparent elsewhere
    c. ID# (7, 11, 13, 14): same edges, opposite opacities as group b. in corresponding order.
    d. ID# (3, 6, 9, 12): a single edge bisecting the bounding square thru two opposite sides.
        (3): horizontal bisector, opaque below, transparent above
        (6): vertical bisector, opaque right, transparent left
        (9): vertical bisector, alpha opposite of (6)
        (12): horizontal bisector, alpha opposite of (3)
    e. ID# (5, 10): a drawing with two edges spanning opposite pairs of adjacent edges.
        * Ambiguity: the opaque area could be a “straits” [S] or “isthmus” [I] pattern.
        (5) [S] opaque in upper left and lower right corners, transparent elsewhere
        (5) [I] transparent in upper right and lower left corners, opaque elsewhere
        (10) [S] opaque in upper right and lower left corners, transparent elsewhere
        (10) [I] transparent in upper left and lower right corners, opaque elsewhere
6. Assign the geometric drawings to places on the output canvas. Each drawing is situated in a bounding square spanning between the original sample locations of the upper left and lower right cells of its corresponding neighborhood.
7. Compute normalized isovalue interpolations for endpoints of edges on the boundaries of cells.
    a. Compute two endpoints for each cell, if needed.

   i. Along the top edge, toward the next cell to the right, if there is one
   ii. Along the left edge, toward the next cell below, if there is one
b. Each value is a linear interpolation.
   i. The source range for interpolating is the positions of sample points. The range spans from the current sample to the next sample in the given direction (rightward or downward).
   ii. The destination range is scalars from 0 to 1.
   iii. The interpolation key is the isovalue.
   iv. If the isovalue is not in the source range, do not interpolate.

8. Fix the positions of the endpoints of edges in the drawings based on the interpolations.
   a. Endpoint along the upper edge of the square:
      $\mathbf{(x_{current} + i_{upper\_current} * R, y_{current})}$
      - $x_{current}$ is the x coordinate of the origin of the cell (its top-left corner)
      - $i_{upper\_current}$ is the isovalue interpolation along the upper edge
      - R is the grid resolution
      - $y_{current}$ is the y coordinate of the origin of the cell
   b. Endpoint along the left edge of the square:
      $\mathbf{(x_{current}, y_{current} + i_{left\_current} * R)}$
      - $i_{left\_current}$ is the isovalue interpolation along the left edge
   c. Endpoint along the right edge of the square (if there is a right edge):
      $\mathbf{(x_{next}, y_{current} + i_{left\_next} * R)}$
      - x_{next} is the x coordinate of the origin of the next cell to the right.
      - i_{left_next} is the isovalue interpolation along the left edge of the next cell to the right
   d. Endpoint along the lower edge of the square (if there is a lower edge):
      $\mathbf{(x_{current} + i_{upper\_next} * R, y_{next})}$
      - y_{next} is the y coordinate of the origin of the next cell below.
      - i_{upper_next} is the isovalue interpolation along the upper edge of the next cell to the right
9. Resolve straits/isthmus cell configurations (see step 5.e.)
   a. Compute the two-dimensional linear average of the samples at the four corners.
   b. Threshold the average by the isovalue. If the average is less, resolve the cell as a straits cell. If the average is greater or equal, resolve it as an isthmus cell.
10. Draw region boundary lines inside each cell using the fixed endpoints, and fill with color opacity according to the fill pattern for the final configuration of the cell.
11. Collate all the cells together into a global drawing layer.
12. Add a boundary line along all boundaries between opaque and transparent areas.

Sources:
[1] lecture notes
[2] Chunawala, Q. (2023) “Drawing shapes with marching squares.” Baeldung CS. https://www.baeldung.com/cs/marching-squares

Visualization HW1
Seren Lowy – 2023/9/18

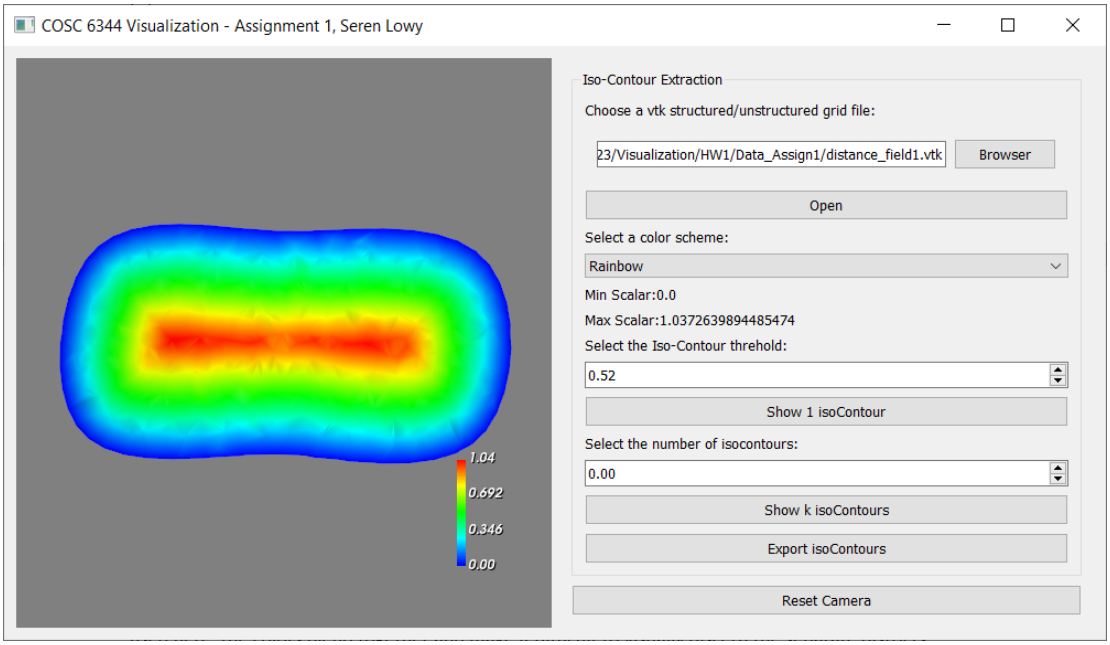
2. Color-coding schemes

Implementation

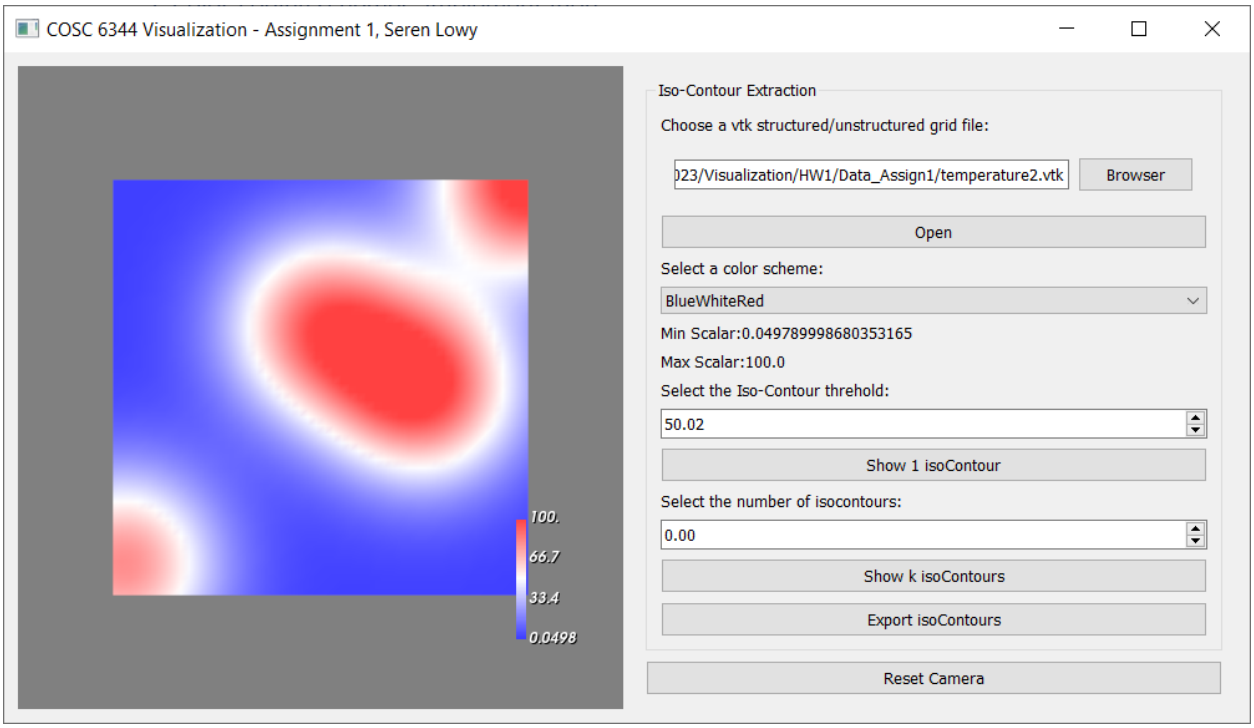
Please see the code file `hw1_2.py` in this folder.

Screenshots

Rainbow

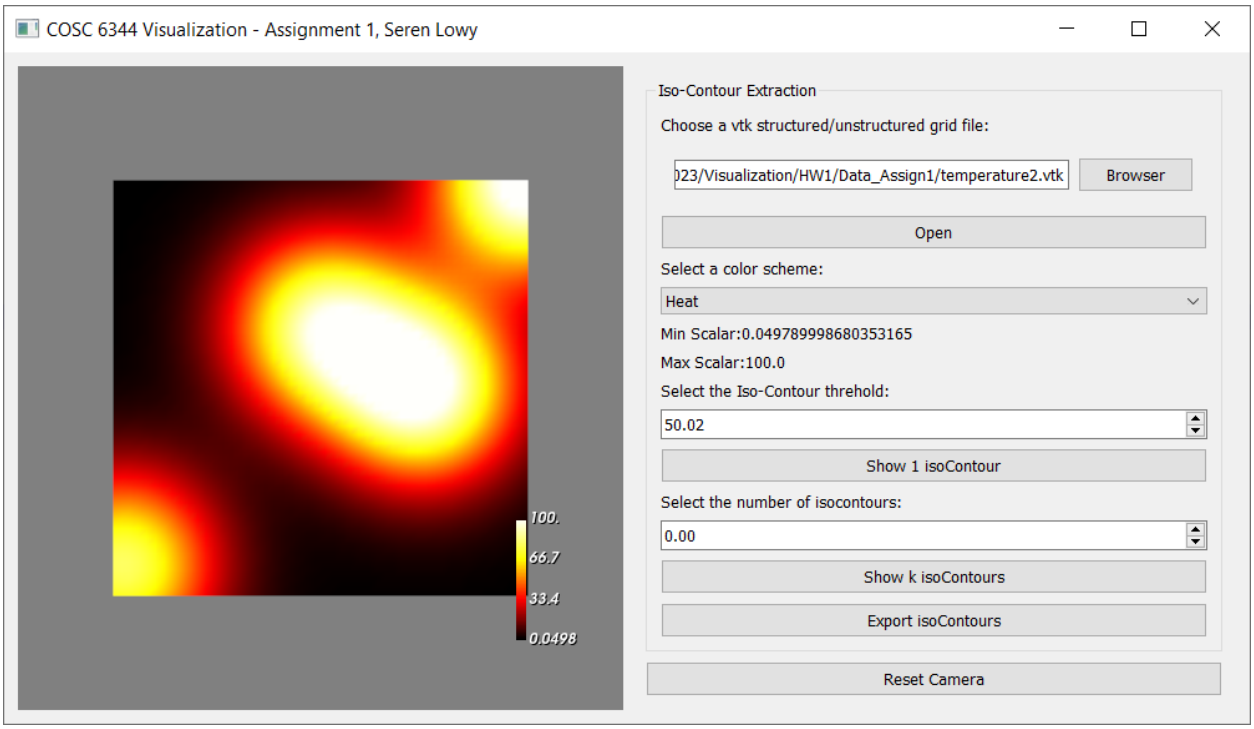


Blue, White, Red

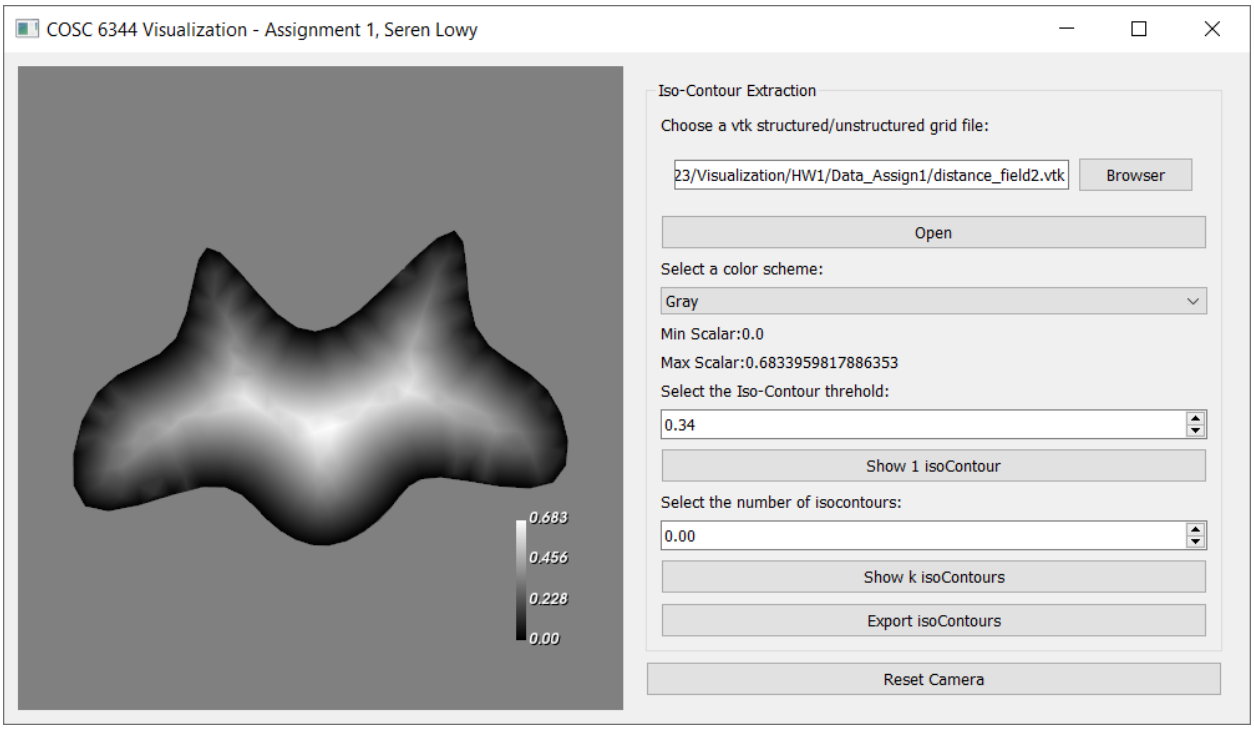


Visualization HW1
Seren Lowy – 2023/9/18

Heat



Grayscale



Visualization HW1
Seren Lowy – 2023/9/18

3.1 Texas voting districts: sequential color scheme

The example image attempts to show the Texas voting districts, a categorical dataset, using a sequential color scale. This causes districts with similar id numbers to appear with very similar colors. If districts with adjacent id numbers are geographically adjacent, which happens frequently on the map of Texas used here, the colors blend together and make it difficult to visually discern the separate districts.

3.2 Texas voting districts: improvements to color scheme

Colors that are widely spaced apart instead of continuous work better for representing categorical data, which is discrete and has a finite range, typically a very small range. The Texas dataset contains only 38 categories. This many categories is still a tight squeeze on a sequential color scale such as the rainbow (hue) scale, but the rainbow scale only occupies one linear path in the HSL color space (i.e. the line at constant, maximum saturation and middle luminance where only hue varies). It should be possible to differentiate colors better using other dimensions which the rainbow scale does not explore (saturation and/or luminance).

The HSL color space is shaped in a way that limits the range of saturation when luminance is too low or high. At any luminance, colors at low saturation are closer to each other overall, even when their hues differ widely. To improve the distinguishability of the color scheme, we need to choose colors that are not close to any other colors in the scheme. So it would be better to include fewer colors in our scheme from the high and low extremes of luminance or the low extreme of saturation.

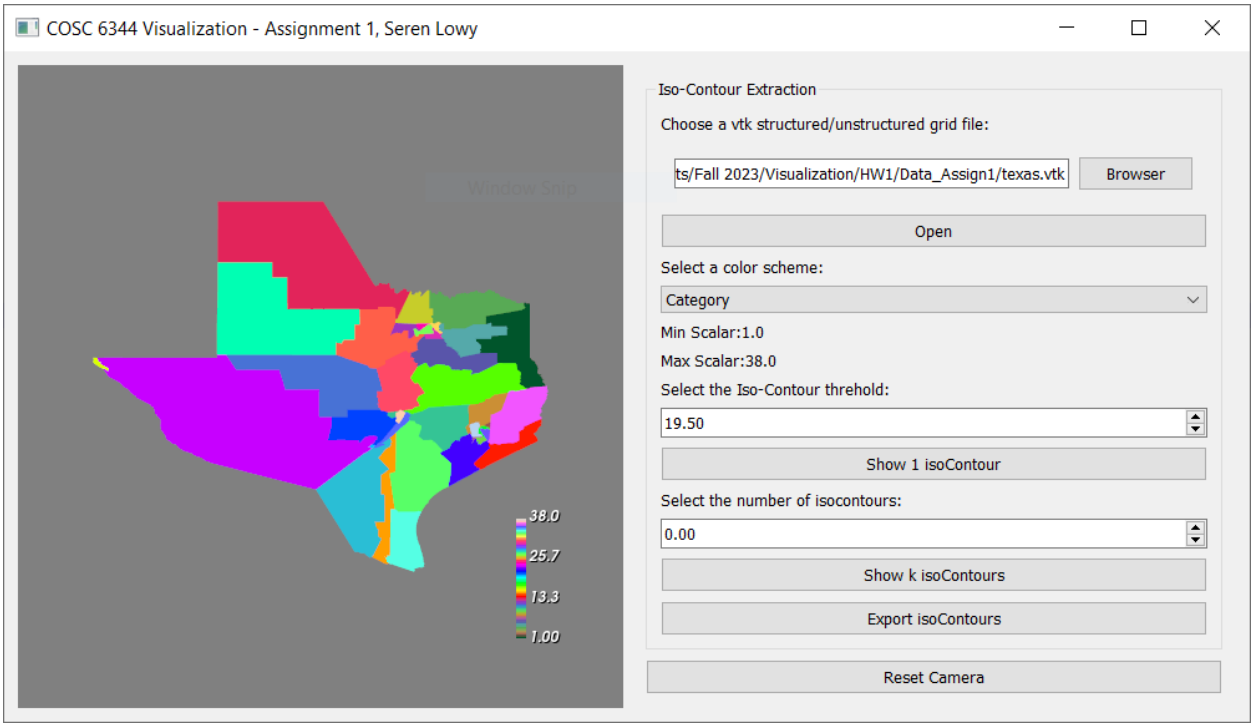
To ensure that colors are distinguishable, we can start from a lower bound of luminance and take large steps around the hue wheel. After a complete circuit, we can increase luminance by a large step and begin another circuit of hue. This time, we can vary saturation by a large step before increasing luminance again. If needed, we can add another cycle of hue at the upper bound of luminance. To help distinguish the colors even better, the discrete hues and saturations chosen at adjacent discrete levels of luminance should be out of alignment, and there should be fewer hues at the extremes. This should allow enough colors to paint the number of regions that we need to visualize on the map of Texas.

Example:

Black pink (220,240,40)	Black sea green (100,240,40)				
Dark red (0,120,80)	Dark gold (40,120,80)	Dark green (80,120,80)	Dark cyan (120,120,80)	Dark blue (160,120,80)	Dark violet (200,120,80)
Deep orange (20,180,100)	Deep lime (60,180,100)	Deep sea green (100,180,100)	Deep lapis lazuli (140,180,100)	Deep indigo (180,180,100)	Deep pink (220,180,100)
Vivid red (0,240,120)	Vivid orange (20,240,120)	Vivid gold (40,240,120)	Vivid lime (60,240,120)	Vivid green (80,240,120)	Vivid sea green (100,240,120)
Vivid cyan (120,240,120)	Vivid lapis lazuli (140,240,120)	Vivid blue (160,240,120)	Vivid indigo (180,240,120)	Vivid purple (200,240,120)	Vivid pink (220,240,120)
Light orange (20,180,140)	Light lime (60,180,140)	Light sea green (100,180,140)	Light lapis lazuli (140,180,140)	Light indigo (180,180,140)	Light pink (220,180,140)
Pale red (0,120,160)	Pale gold (40,120,160)	Pale green (80,120,160)	Pale cyan (120,120,160)	Pale blue (160,120,160)	Pale violet (200,120,160)
				White orange (20,240,180)	White lapis lazuli (140,240,180)

Visualization HW1
Seren Lowy – 2023/9/18

3.3 Texas voting districts: improved color scheme screenshot

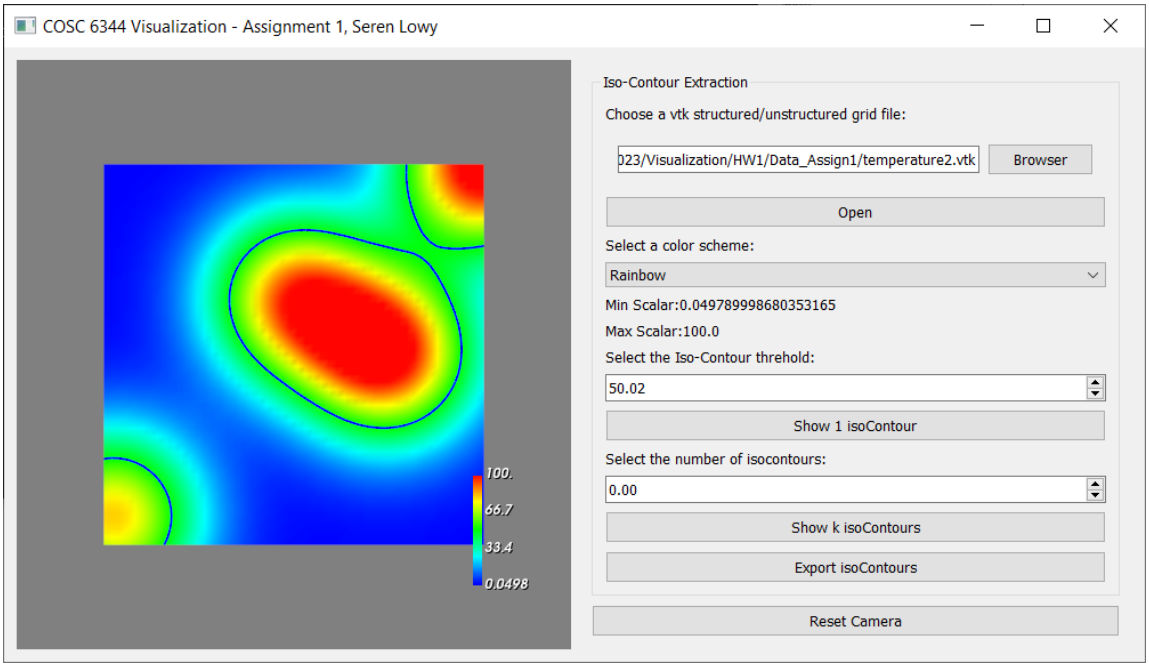


4. Contour visualization

Please review the code file `hw1_main.py` which is mostly your skeleton code with modifications and functions added where asked.

Screenshots

1 isocontour



Visualization HW1
Seren Lowy – 2023/9/18

K isocontours