Prof. Dr. Hans Hagen
hagen@cs.uni-kl.de

Dipl.-Inf. Cornelius Müller
cmueller@cs.uni-kl.de

Processing (`http://processing.org`) is an open source Java-based programming language and environment that allows easy creation of images, animations, and interactions.

This assignment is aimed at understanding color mapping of 2D scientific data. To this purpose, the overall task consists of loading and displaying a data file using color mapping techniques.

Two datasets are the basis for this assignment. The first – *temperature* – results from a climate simulation and represents a projection of world-wide temperatures according to a specific computational model. The second dataset – *vorticity* – is taken from a Computational Fluid Dynamics (CFD) simulation. It consists of (scalar) values of the vorticity variable that measures the strength of locally rotating flow behavior. In other words, it provides a simple indicator of vortical motion in the flow, which is often of interest in analyzing technical flows (see also `http://en.wikipedia.org/wiki/Vortex` for a more detailed explanation).

You will get one point for each completed steps of the three steps below.

To get a point for a step, implement the tasks below. You should be able to demonstrate a working program and explain your implementation. There are three steps in every exercise. You may use either your own computer or one of the computers in the lab in 36/223. This exercise is due by **June 13, 2014** (as well as Assignment 2), but you are encouraged to present it earlier, maybe even before you start assignment 2, and preferably at one of the Question Times at Thursday, 11:45 am in 36/223.

Prestep: Hello, World!

Download, install, and start Processing. On the lab computers, Processing is installed under SuSE Linux at `/opt/processing-2.1.2/processing`. Familiarize yourself with the Processing system, using tutorials (`http://processing.org/tutorials/`), reference documentation (`http://processing.org/reference/`), and the Processing wiki (`http://wiki.processing.org`).

Essentially, Processing works in an event-driven manner, i. e. functions in your program are called to respond to external events. Two basic functions are called by the environment to initialize your program (function `setup()`) and to redraw the window (function `draw()`). Other interaction, e. g. mouse clicks and keyboard events, are handled similarly (functions `mouseClicked()` and `keyPressed()`).

Download the example sketch `Example1.zip` from the *Exercises* section of the OLAT course page (a *sketch* is Processing's notion of a project). Unpack it to a location of your choice, and run it within Processing. The sketch demonstrates the basic functionality you need to build on for the remaining steps, and provides a simple Buttons class for minimal GUI functionality.

Step 1: Data? What data?

Write a function that reads a rectilinear, scalar dataset (`.asc` files) from the filesystem into your program. The file format consists of a short description of the rectilinear grid (size, spacing), followed by a textual representation of the data, e. g.

```
Nx Ny Sx Sy v00 v10 v20 … v01 v11 v21 …
```

with numbers separated by whitespace. Here, `Nx` and `Ny` represent the number of samples on the horizontal and vertical axes of the regular grid, `Sx` and `Sy` describe the cell size, and `v00`, `v10`, …are the `Nx` $\times$ `Ny` data values along x-axis (fastest) and y-axis (slowest). To clarify, the overall size of the domain is then $(\texttt{Nx} - 1) * \texttt{Sx}$, $(\texttt{Ny} - 1) * \texttt{Sy}$, and you may assume the grid to be given at the origin.

There are two data files to test your reader, `temperature.asc` and `vorticity.asc`, located in the `data/` directory of the provided sketch.

Note: Processing does not provide sophisticated file parsing routines. A simple way to read the files into a list of numbers is to use the statement

```
String tokens[] = splitTokens( join( loadStrings(filename), " " ), " " );
```

Summary of Tasks: Write a function to load the two datasets.

Step 2: (More than) 50 Shades of Grey

Convert the rectilinear data read in Step 1 into an image (using Processing's `PImage` class) by assigning a greyscale value to every datum in the dataset. You can achieve this by first applying a linear map to normalize the value range of the data to $[0, 1]$ and then assigning R, G, B components from the normalized values.

Display the resulting image on the screen, possibly using magnification (see Processing's `scale()` function). Again, test with both datasets.

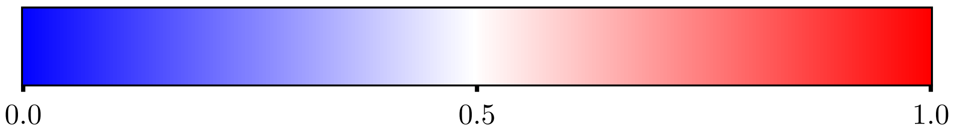
Summary of Tasks: Convert the data into an greyscale image and display it on the screen.

Step 3: So colorful!

Several custom color maps are provided in the `data/colormaps/` directory of the given sketch. They describe how each value in $[0, 1]$ is mapped to a color using a piecewise linear map resulting from interpolation of a fixed number of colors. If x_i are the parameters and c_i the corresponding colors, then x is mapped to

$$c(x) \;=\; (1-r) * c_i + r * c_{i+1}, \;\text{ where }\; r = (x - x_i)/(x_{i+1} - x_i), \;\text{ for }\; x \in [x_i, x_{i+1}]$$

For example, the `coolwarm` colormap is given by three colors blue, white, and red at parameters 0.0, 0.5, and 1.0, respectively, as depicted below.



The file format is again simple: each line contains one color, given by first its parameter, and followed by the three components red, green and blue.

Implement a `ColorMap` class that allows reading a color map into your program from one of the provided files, and write a method `interpolate()` that returns the interpolated color for a value in $[0, 1]$. (Hint: use binary search to identify the interval containing the interpolation point.)

Extend your program from Step 2 such that the mapping is performed using the `ColorMap` class and one of the provided color maps.

Augment your program with a minimal GUI using the `Buttons` class provided in the sketch. There should be a button for each available color map, such that when clicked the corresponding color map is applied to the image.

Use these facilities to compare color maps on the provided data sets. Consider the advantages and disadvantages of individual color maps.

Summary of Tasks: Extend your program such that the color mapping is performed using the provided color map files. Implement a minimal GUI to switch between the color maps. Compare them.