**Brent Barbachem & Sean Coogan**

**MSIM 741: Dynamic Color Options**

**December 2, 2015**

Applications such as Geographic Information Systems provide users with the ability to choose predetermined and/or custom color configurations to represent the data. The majority of users will be inclined to pick a color scheme similar to the one in Figure 1. The color scheme contains several different colors, but the color scheme does not enable users to successfully interpret data.



Figure [1]: Color interval including all colors

According to Dr. Ananya Mandal, anywhere from five to eight percent of men may be colorblind and anywhere from 0.5 to one percent of women may be colorblind (Dr. Mandal). There are three types of partial colorblindness, Tritanopia, Protanopia, and Deuteranopia. Tritanopia is the colorblindness that results from sensitivities in the short-wavelength cones; it is difficult to determine the difference between colors such as blue and yellow. Deuteranopia is the colorblindness that results from sensitivities in the medium-wavelength cones. Protanopia is the colorblindness that results from sensitivities in the long-wavelength cones. Interestingly, individuals that are sensitive to long or medium wavelength colors are unable to distinguish between green and red. Visualization programs often fail to offer explicit options for those that contain the colorblind genes. The goal of this project was to create software that would allow user(s) to dynamically alter the color patterns while adhering to, possible, colorblind restrictions entered by the user.

In the realm of computer visualization, there are numerous practical applications for the work completed for this project. Any software that takes advantage of computer visualization in order to display data can make use the concepts developed in this project. Software that utilizes color differences to visualize data should provide all users with an equal opportunity for interpretation. These applications include but are not limited to software such as Geographic Information Systems (GIS), Computer-Aided-Design (CAD), and 3D/2D Modeling Software. A more important achievement would be the incorporation of these concepts into the displays on computers. If colorblind options are created at the source of the display, the personal computer, then the software companies would not have to include these alterations.

We were able to identify, at least, five visualization guidelines from our course textbook. Below we address these five guidelines individually.

*Guideline 4.14 – To create a set of symbol colors that can be distinguished by the most colorblind individuals, ensure variation in the yellow-blue direction.*

We provide both command line options and keyboard interrupt options for the user(s), to ensure that colors can be distinguished by the majority of colorblind individuals. The user(s) can execute the program passing the argument “-cb colorblind\_option”. The colorblind\_option refers to either “A”, “D”, “P”, or “T” which represent the completely colorblind, medium-wavelength, long-wavelength, or short-wavelength sensitivities. The user(s) can also enter these letters during the execution in order to change the current colorblind arguments.

*Guideline 4.15 – Do not use more than ten colors for coding symbols if reliable identification is required, especially if the symbols are to be used against a variety of backgrounds.*

In order to successfully code this visualization guideline, we ensured that the default number of symbols would be eight. We chose eight, because it is less than the suggested maximum and we noticed that eight bands would provide each of the colors with enough contrast to easily distinguish the color changes. We did allow the user(s) to alter the number of color bands using a command line argument “-b number\_of\_bands”.

*Guideline 4.19 – Use a spectrum approximation psuedocolor sequence for applications where its use is deeply embedded in the culture of users. This kind of color sequence can be used where the most important requirement is reading map values using a key. If this sequence is used, the spacing of the colors should be carefully chosen to provide discriminable steps.*

We have used and elevation data map to illustrate the use of a psuedocolor sequence in which the colors are important for determining elevation bands. We have created an algorithm that will pick two saturated colors that are at least a certain distance apart in order to create bands that are easily distinguishable.

*Guideline 4.20 – If it is important to see highs, lows, and other patterns at a glance, use a psuedocolor sequence that monotonically increases or decreases in luminance. If reading values from a key is also important, cycle through a variety of hues while trending upwards or downward in luminance.*

We have used and elevation data map to illustrate the use of a psuedocolor sequence in which the colors are important for determining elevation bands. We have created an algorithm that will pick two saturated colors that are at least a certain distance apart in order to create bands that are easily distinguishable. We provide VTK with two distinct colors and a midpoint. VTK chooses the remaining colors for the lookup table based on the luminance values for the colors between each endpoint and the midpoint.

*Guideline 10.9 – In interfaces to view map data in 3D, the default controls should allow for tilt around a horizontal axis and rotation about a vertical axis, but not rotation around the line of sight.*

VTK allows the user(s) to perform these actions by default, but we manually added these features into our program. In order to add keyboard interrupts to our program we had to manually add in features such as keyboard interrupts, rotation, and tilt.

We decided that there were four main questions that needed to be answered in order to write the code that would produce the expected results. Is there a color generation technique that ensures that the generated color is highly saturated? VTK uses a color lookup table that includes the two end colors and a midpoint; how should we proceed with the calculation of a midpoint color? What threshold for distance is adequate for generating colors that are distinct? Most importantly, how do choose colors that meet the colorblind restrictions?

We decided that the colors at each end of the elevation interval should be as vivid as possible, so that the users would be able to easily see the colors. A higher saturation also ensures that there can be a larger number of distinguishable colors as opposed to two non-saturated colors. We were able to guarantee that both of our colors would be saturated by ensuring that one of the RGB fields was set to 1.0, one field was set to 0.0, and the last field was set to a random number.

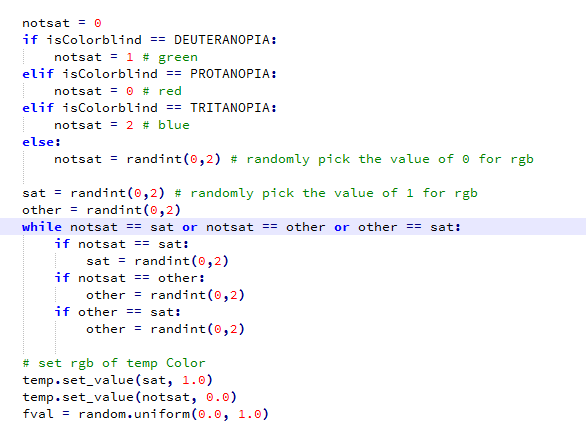


Figure [2]: Saturation Technique

The program contains two different midpoint determination algorithms. The first algorithm uses a simple grey color as the midpoint for the elevation interval. Interestingly, this simple method create a more aesthetically pleasing interval, regardless of its simplicity. The second algorithm calculates a distance between the two colors. The colors are treated as 3D points where RGB can also be interpreted as xyz. The distance between the two colors is calculated using the following formula:

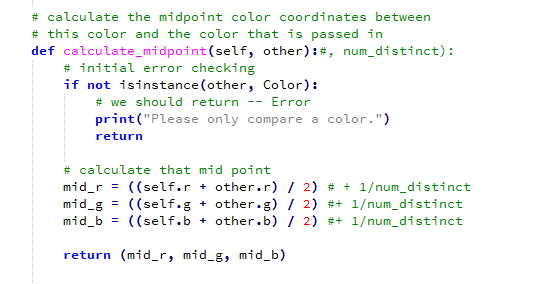


Figure [3]: Midpoint calculation function

The function returns the RGB values for the midpoint color, and this color is passed onto the creation of the color lookup table. The code includes a switch that can be selected, only from within the code, to change which of these options will be used for selecting the midpoint.

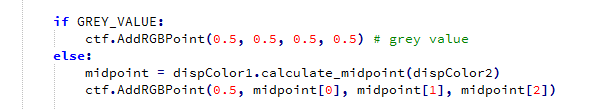


Figure [4]: Midpoint selection

We wanted to pick two colors that would differ on the spectrum by a distance that was sufficient enough for the user to easily determine the correct color. We used the Euclidean distance calculation to calculate the distance between the colors, shown in the formula below.

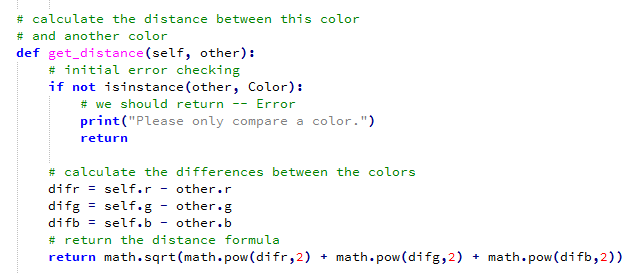


Figure [5]: Distance between colors

The number of colors become constrained as the user enters possible colorblind options, so we needed to ensure that we could still pick two colors. Our algorithm will randomly generate a color that is not restricted by the colorblind options. After one color has been generated, we will attempt to generate a second random color. If the color is not a certain distance away from the other color, or we have tried ten times, then we will start the entire process over again. The **red** arrow in the image below indicates the required distance between the two colors.

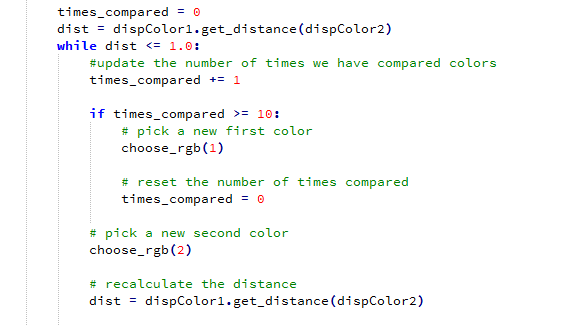


Figure [6]: Choosing two colors

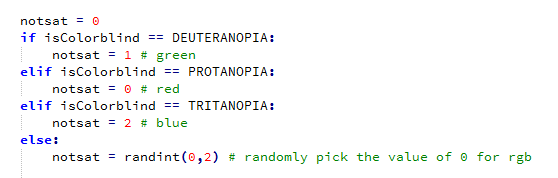


Figure [7]: Setting one RGB value to 0.0

It is important to discuss what was accomplished in regards to the colorblind options, because the entire project was built upon this subject matter. There is a slight difference between the accepted definition of colorblind in the long- and medium-wavelengths and how we decided to display the distinction. We could have coded both of these choices the exactly the same, because Protanopia and Deuteranopia are each characterized by the inability to differentiate **red** and **green** shades. In order to display a visual difference between these two options, we chose to eliminate the **red** colors when the user selects Protanopia, and we chose to eliminate the **green** colors when the user selects Deuteranopia. The figure above displays how we were able to eliminate the appropriate **red** or **green** fields; depending on the user input options, the **red** or **green** field will be set to 0.0. The user can also select the Tritanopia colorblind option, and the program will eliminate **blue** and **yellow** colors. The elimination of two colors occurs, because Tritanopia is defined as the inability to differentiate between **blue** and **yellow** colors. We were able to disregard **blue** colors through the same process as the elimination of green and red previously discussed. We were able to disregard **yellow** colors through the process invoked in the image below. The image below is indicating that the **green** component of the color should contain a value greater than 0.7 to ensure that no traces of **yellow** appear.

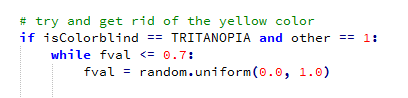


Figure [8]: Eliminating yellow

***Results:***

In this section we will discuss the results of the project and what we accomplished.

***Conclusion:***

In this section we will go back through and note all of the stuff that was discussed.

***Acknowledgements/Resources:***

<http://www.news-medical.net/health/Color-Blindness-Prevalence.aspx>

Course Textbook.

<https://tug.org/pracjourn/2007-4/walden/color.pdf>