CS100J, Spring 2008. Assignment A4. Color models. Due: Monday, 03 March.

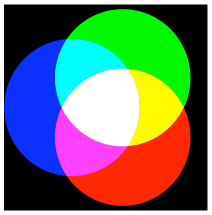
This assignment: (1) introduces three color models that are used in computing and graphics and (2) gives you practice in writing functions. Start early on this assignment and keep track of your time. Just before you submit, tell us how many hours you took in a comment at the top of class `A4Methods`.

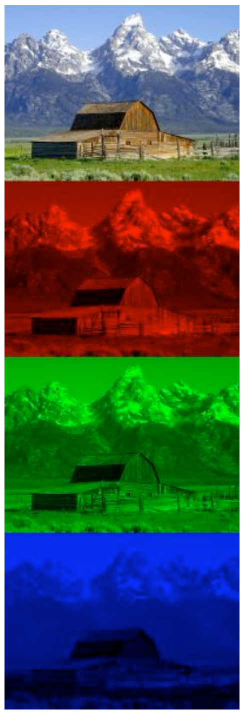
This assignment was given a few semesters ago in CS100J. We give it again because it is such a good one, although we have made some changes. We ask you not to cheat by obtaining a copy of the earlier solution or a version handed in by another student. Such cheating helps no one, especially you, and it makes more unnecessary work for us. We are working hard to make this a good course, and in return we expect you to be honest. We will not take kindly to cheating.

You may work with one partner. Form your group on the CMS several days before the assignment is due; do not wait until the last minute. Remember: it is dishonest for partners to split the work in half and work independently. Program and test and debug together, alternating writing the methods (with the other person watching and helping).

You must also produce a class `A4Tester` that contains test cases. Include enough test cases to be sure that your program is correct. When you start on a method, think of the test cases you will need to test the method and write the `assertEquals` statements before writing the method body. Add more test cases as you write the method body to ensure that there is complete test coverage —so that each statement in the method body is exercised during at least one test case. We give you some test cases.

In writing this assignment, we made heavy use of articles on the various color models on Wikipedia.





Color model RGB

The RGB model is an additive model in which **red**, **green**, and **blue** are mixed to produce other colors, as shown in the image to the left. Black is the absence of color; white is the maximum presence of all three. RGB has its roots in the 1953 RCA color-TV standards and in the Polaroid camera. RGB is used in your TV screen and on web pages.

The amount of each color is represented by 8 bits, which gives a number in the range 0..255. Black is[0, 0, 0] (the first integer is the red, R; the second, G, green; the third, B, blue), and white is [255, 255, 255]. There are 16,777,216 different colors.

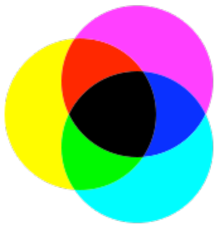
Class java.awt.Color has some constants that you can use for some of the possible colors. For example, Color.magenta is [255, 0 , 255] and Color.orange is [255, 200, 0]. Web page http://en.wikipedia.org/wiki/List_of_colors, gives names to many colors in the RGB model.

In the upper right is a colored image. Below it is its separation into red, green, and blue. In the three separation panels, the closer to black a point is the less of that color it has. The white snow is made up of a large amount of all three colors and the brown barn is made up of red and green with very little blue. Here is a high resolution version of these four images.

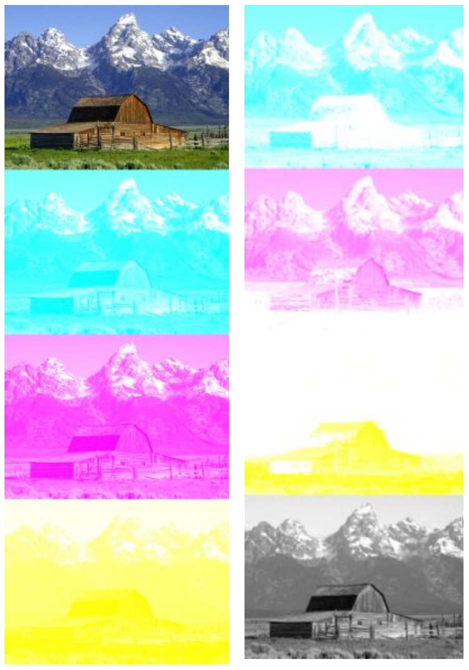
In some graphics systems on the web, RGB is used with Java **double** numbers in the range 0..1.0, so that 255 is actually given by 1.0. Later, when we do calculations, we convert each number in the integer range

0..255 to a **double** number in the range 0..1.0, calculate, and then convert back to 0..255.

The complementary color of RGB color [r, g, b] is the color [255-r, 255-g, 255-b]. It is like a color negative.



Color model CMYK

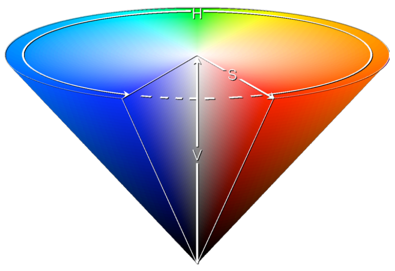


On your ink-jet printer, you buy expensive ink cartridges in the colors cyan (RGB color [0, 255, 255]), magenta [255, 0, 255], yellow [255, 255, 0], and black [0, 0, 0]. Black is referred to using K, for "Key plate". Theoretically, you need only CMY, but the K (the black) is there because without it, black doesn't look black enough. Also, you have to use a lot of CMY colors to get real black, and the paper gets too wet. Finally, text is usually black, and you save a lot of ink by having the additional black cartridge.

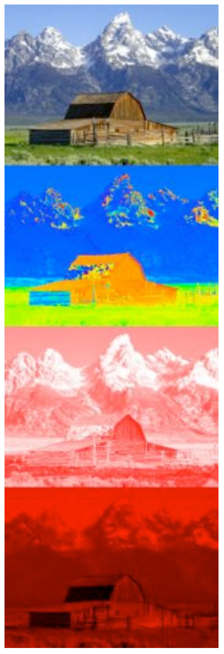
In the upper right, you see an image; below it is its separation into its cyan, magenta, and yellow. To the right of that, you see the same image separated into four components; C, M, Y, K. Much less of the CMY colors are needed to make the image when black is also used.

Printers use CMYK. We usually print on white paper, and the absence of C, M, Y, and K leaves it white. As more of each color is added, color is added to the page. Add more of each of the three basic colors and the image gets blacker. This works well in printing on white paper. The RGB system starts with a black background and adds more and more of each color to get white. This is called an "additive system"; CMYK is a "subtractive" system. Here is an enlarged version of the CMY image and an enlarged CMYK image.

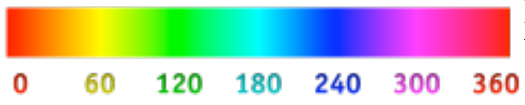
In our CMYK system, each of the four components is represented by a **double** value in the range 0..1.0.



Color Model HSV (or HSB)



The HSV model, used heavily in graphics applications, was created in 1978 by Alvy Ray Smith. Artists prefer the HSV model over others because of its similarities with the way humans perceive color. HSV can be explained in terms of the cone that appears to the left.



H, the *Hue*, defines the basic color. It is an angle in the range 0 ≤ H< 360, if one views the top of the cone as a disk. Red is at angle 0. As the angle increases, the hue changes to orange, yellow, green, light blue, dark blue, violet, and back to red. The image in this paragraph shows the angles for some colors.

S, in the range 0 ≤ S ≤ 1, is the *Saturation*. It indicates the distance from the center of the desk. The lower the S value, the more faded and grayer the color. The higher the S value, the stronger and more vibrant the color.

V, the *Value*, also called the *Brightness*, is in the range 0 ≤ V ≤ 1. It indicates the distance from the point of the cone to the wheel. If V is 0, the color is black; if 1, the color is as bright as possible. The saturation indicates how faded or "colorless" the color is.

To the right at the top is a picture. Below it we see its hue, saturation, and brightness components. The hue component shows color. The snow has color, but its saturation is low, making it almost grayish. Look at the various components of the image —the sky, the green grass, the snow, the dark side of the barn, etc.— to see how each component H, S, and V contributes. See more detail in this high-resolution version.

A GUI (Graphical User Interface) that provides understanding, and your assignment

Click on the link a4.jar to download a java application that is the result of this assignment (or download it from the course webpage). After downloading it, double click its icon, and a GUI will appear on your monitor with two color panes and sliders for all the RGB, CMYK, and HSV components. The initial color of the left pane is red, and the pane to its right contains the complementary color. In the title, you see the RGB values for red. In addition, the color pane and complementary color pane contain information on the RGB, CMYK, and HSV values for the current colors in the panes.

When you move one slider, the colors change accordingly —and other sliders change too, so that all three models (RGC, CMYK, and HSV) register the same color.

As a first move, very slowly move the Hue slider H up and watch how the colors (and the other sliders) change. Then change the saturation S and value V sliders to see their effect. Play around like this, while reading about the color models, to get an understanding about they relate to each other. It's neat!

The assignment

Here are five files for classes A4, HSV, CMYK, A4Methods, and A4Tester. Download the .zip file, either from here or from the course webpage, store the files in a new directory, compile them in DrJava, and execute `A4.main(null);`. You will see the GUI as discussed above, but the text will not display properly. Also, the RGB sliders work, but the other sliders have no effect. Your job is to write and check out, *one by one*, the methods in class `A4Methods`. As you do, more and more of the GUI will work properly. As you proceed, you must use test cases in class `A4MethodsTester`. We have provided a number of test cases. Below, we show you the order in which to write the functions. First, a word about objects to contain RGB, CMYK, and HSV values.

(a) Class `java.awt.Color` is used to maintain RGB colors. (You have to use `import java.awt.*;`). Create a new RGB color object using **`new`** `Color(r, g, b)`, where `r`, `g`, and `b` are in the range 0..255. Given an RGB object `rgb`, obtain its three components using `rgb.getRed()`, `rgb.getGreen()`, and `rgb.getBlue()`.

(b) Use the given classes `CMYK`, and `HSV` for objects that contain CMYK and HSV. Take a look at the classes; their use should be obvious. All fields are public, to make it easy to use them. There are no methods except for the constructors.

0. Function complementRGB(Color). Write this function first. We gave you some code; you have to fix it. We provide a test case for it in class A4Tester, which won't work until you rewrite the function.

1. Function truncateTo5(String). There is a return statement in it, so the program compiles. Write this function. We provided test cases for it in class A4Tester.

2. Function toString(Color). Write this function. When this is written, RGB values will appear in the title of the GUI and in the two color panes. We provided one test case for this function in class `A4tester`; that is all that is needed.

3. Function toString(CMYK). Write this function. When written, CMYK values will appear in the two color panes. This function should call function `truncateTo5` to truncate the CMYK values to 5 characters. We do NOT provide test cases for this function. You write at least two of them.

4. Function toString(HSV). Write this function. When written, HSV values will appear in the two color panes. This function should call function `truncateTo5` to truncate the HSV values to 5 characters. We do NOT provide test cases for this function. You write at least two of them.

5. Function RGBtoCMYK. This function converts an RGB value to a CMYK value. When you get it working, moving the RGB sliders will cause the CMYK sliders to move as well. There are several different ways to convert, depending on how much black is used in the CMYK model. Our conversion uses as much black as possible. Let R, G, and B be the color components of the RGB in the range 0..1.0 (not 0..255!!). Then the conversion is as follows:

1. Compute $C' = 1 - R$, $M' = 1 - G$, and $Y' = 1 - B$.

2. If C', M', and Y' are all 1, then use the CMYK values (0, 0, 0 , 1).
   If not, then compute and use:
   K = minimum of C', M', and Y',
   $C = (C' - K) / (1 - K)$, $M = (M' - K) / (1 - K)$, $Y = (Y' - K) / (1 - K)$.

That's it! That's not too bad, is it? Providing test cases is a bit problematic because **`double`** values are only approximations to the real values, and slightly different ways of computing might produce different results. To show you how to do this, we provide in `A4Tester` the three test cases, testing only the truncation of the values to 5 characters, since that is what appears in the color panes of the GUI.

6. Function CMYKtoRGB. This function converts a CMYK value to an RGB value. When you get it working, moving the CMYK sliders will cause the RGB sliders to move as well. Let C, M, Y, and K be the color components of the CMYK value, all in the range 0..1.0. Then the conversion is as follows:

$R = (1 - C)(1 - K)$, $G = (1 - M)(1 - K)$, and $B = (1 - Y)(1 - K)$.

This produces RGB values in the range 0..1.0, and they must be converted to the range 0..255. Be sure you do this.

Put at least two test cases for this function in class `A4Tester`. To figure out what test cases to use and the values they produce, you can use the GUI that we provided (execute a4.jar).

7. Function RGBtoHSV. This function converts an RGB value to an HSV value. When you get it working, moving the RGB sliders will cause the HSV sliders to move as well. Here's how the conversion works.

Let `H` satisfy `0 ≤ H < 360` and `S, V, R, G,` and `B` be in the range `0..1`. Let `MAX` be the maximum and `MIN` be the minimum of the (`R`, `G`, `B`) values.

H is given by 5 different cases:

(a) MAX = MIN: $H = 0$.
(b) MAX = R and G ≥ B: $H = 60.0 * (G - B) / (MAX - MIN)$.
(c) MAX = R and G < B: $H = 60.0 * (G - B) / (MAX - MIN) + 360.0$

(d) MAX = G: H = 60.0 * (B - R) / (MAX - MIN) + 120.0
(e) MAX = B: H = 60.0 * (R - G) / (MAX - MIN) + 240.0

S is given by: if MAX = 0 then 0, else 1 – Min/Max.

V = MAX.

You have to provide at least 5 test cases in class A4Tester, so that each expression in the cases for H are evaluated in at least one test case.

8. Function HSVtoRGB. This function converts an HSV value to an RGB value. When you get it working, everything in the GUI should work.

Let `Hi = floor(H/60) % 6, f = H/60 – Hi, p = V(1-S), q = V(1-fS), t = V(1-(1-f)S).`

Then `R`, `G`, and `B` depend on the value `Hi` as follows:

If `Hi = 0`, then `R = V, G = t, B = p`
If `Hi = 1`, then `R = q, G = V, B = p`
If `Hi = 2`, then `R = p, G = V, B = t`
If `Hi = 3`, then `R = p, G = q, B = V`
If `Hi = 4`, then `R = t, G = p, B = V`
If `Hi = 5`, then `R = V, G = p, B = q`

This produces RGB values in the range 0..1.0, and they must be converted to the range 0..255. Be sure you do this.

Provide at least 6 test cases for this function because of the 6 possible values of `Hi`.

Submission of the assignment

Place a comment at the top of class A4Methods that indicates how much time you spent on this assignment. Make sure that class A4Methods is indented properly. Make sure that class A4Tester has the appropriate test cases. Submit files A4Methods.java and A4Tester.java on the CMS by the due date. We hope you have enjoyed this assignment and found it instructive.