Name: ___________________________________ AP CS | Unit 6 | Two-Dimensional Arrays

A Pixel Is Worth 24 Bits

Objective
You will understand how to traverse and manipulate two-dimensional arrays through the context of manipulating photos (filters, yay!). Additionally you will review representation of numbers in various bases and be introduced to inheritance and interfaces.

Collaboration
This assignment is to be completed in pairs with a Housemate and you are to employ the Pair Programming technique. You and your partner are to honor the Collaboration Policy throughout this assignment. For this project, you may choose your own partner. **You will need to join a group on Canvas**.

Project Structure
This entire project will be written throughout multiple classes. The headers of most of these methods have already been written for you. There are two folders that you need to be aware of. The first is the `classes` folder and it contains all of the Java class files, in addition to some Java docs for the classes in a folder called `doc`. The second folder, `images`, contains all of the images you may use in this project. If you wish to use your own photos, you should place them here. Aside from adding your own photos, do not modify the project structure in any way (i.e. do not move files and folders around). You will submit the entire project compressed into a zip folder.

Project Setup
Go to the Unit 6 page on Canvas and download the PhoToeShop.zip folder, unzip it, and place its contents inside the `AP CS/Unit 6 - 2D Arrays/PhoToeShop Project` directory.

Inside the project folder, select the `classes` folder, and run `package.bluej`.

You are now ready to explore the project. *However…*

It is imperative that you do not rearrange any of the files and folders once you've placed it in your `AP CS` directory. You may add your own pictures to the images folder, but leave everything else as is. The project structure is complex and contains many file dependencies, so moving things around can cause issues.

If you look at an advertisement for a digital camera, it will specify how many *megapixels* the camera can record. What is a megapixel? A digital camera has sensors that record color at millions of points arranged in rows and columns (Figure 1). Each point is a *pixel* or *picture (abbreviated pix) element*. A *megapixel* is one million pixels. A 16.2 megapixel camera can store the color at over 16 million pixels. That's a lot of pixels! Do you really need all of them? If you are sending a small version of your picture to a friend's phone, then just a few megapixels will be plenty. But, if you are printing a huge poster from a picture or you want to zoom in on part of the picture, then more pixels will give you more detail.

How is the color of a pixel recorded? One technique is the RGB (Red, Green, Blue) color model, which stores values for red, green, and blue, each ranging from 0 to 255. You can make yellow by combining red and green. That probably sounds strange, but combining pixels isn't the same as mixing paint to make a color. The computer uses light to display color, not paint. Tilt the bottom of a CD in white light and you will see lots of colors. The CD acts as a prism and lets you see all the colors in white light. The RGB color model sometimes also stores an alpha value as well as the red, green, and blue values. The alpha value indicates how transparent or opaque the color is. A color that is transparent will let you see some of the color beneath it.

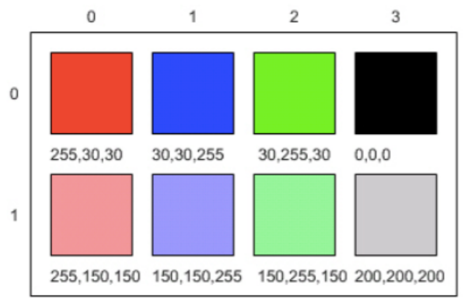


Figure 1 - RGB and Alpha Values Displayed in Rows and Columns

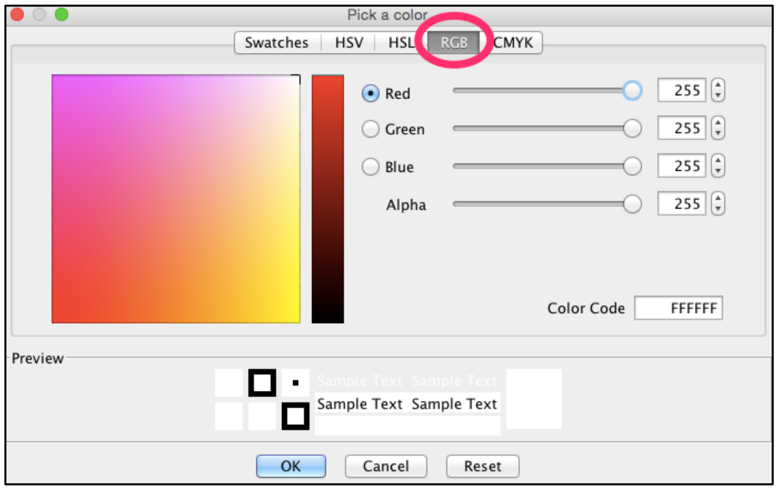
As a reminder, computers store all information as binary numbers. Binary numbers use the digits of 0 and 1 and powers of 2 to represent numbers as *bits*. A bit is simply a 1 or a 0 and a group of 8 bits is known as a *byte*.

1. How many bits are required to represent the numbers from 0 to 255?

2. How many bytes are required to represent one color using the RGB model?

3. How many pixels are in a picture that is 640 pixels wide and 480 pixels tall? How many bytes are required to fill such a picture with RGB colors?

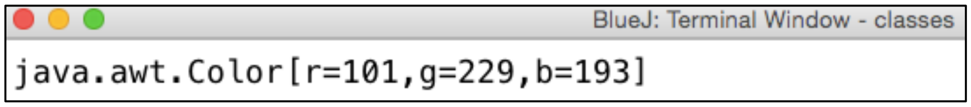
Run the `main` method in the `ColorChooser` class. This will display a window promting you to pick a color. Click on the RGB tab (see Figure 2) and move the sliders around to explore different color combinations.



Red 255
Green 255
Blue 255
Alpha 255
Color Code FFFFFF

Figure 2 - The Color Chooser

When you click the `OK` button, the red, green, and blue values for the color you picked will be displayed as shown in Figure 3. The `Color` class has a `toString` method that displays the class name followed by the RGB values. The `toString` method is automatically called when you print an object. Unless a developer changes the behavior of this method, it will print the RAM address of the object's reference variable. Typically we like to write our own version of `toString` so that we can print the state of an object.



```
java.awt.Color[r=101,g=229,b=193]
```

Figure 3 - toString output from the ColorChooser Class

Java represents color using the `java.awt.Color` class. This is the *full name* for the `Color` class, which includes the *package* (a group of classes) name of `java.awt` followed by a period and then the class name `Color`. Java groups related classes into *packages*. The `awt` stands for Abstract Windowing Toolkit, which is the package that contains the original Graphical User Interface (GUI) classes developed for Java. You can use just the short name for a class, like `Color`, as long as you include an import statement at the beginning of a class source file, as shown below. The `Picture` class contains the following import statement.

```
import java.awt.Color;
```

Use the `ColorChooser` class (run the `main` method) to identify the RGB values of the colors below.

1. Pink
2. Yellow
3. Purple
4. White
5. Green
6. Black

Run the `main` method in the `PictureExplorer` class. This will load a picture of a beach from a file located in the `images` folder, make a copy of that picture in memory, and show it in the explorer tool (Figure 4). It makes a copy of the picture to make it easier to explore a picture both before and after any changes. You can use the explorer tool to explore the pixels in a picture. Click any location (pixel) in the picture and it will display the row index, column index, and red, green, and blue values for that location. The location will be highlighted with yellow crosshairs. You can click on the arrow keys or even type in values and hit the enter button to update the display. You can also use the menu to change the zoom level.

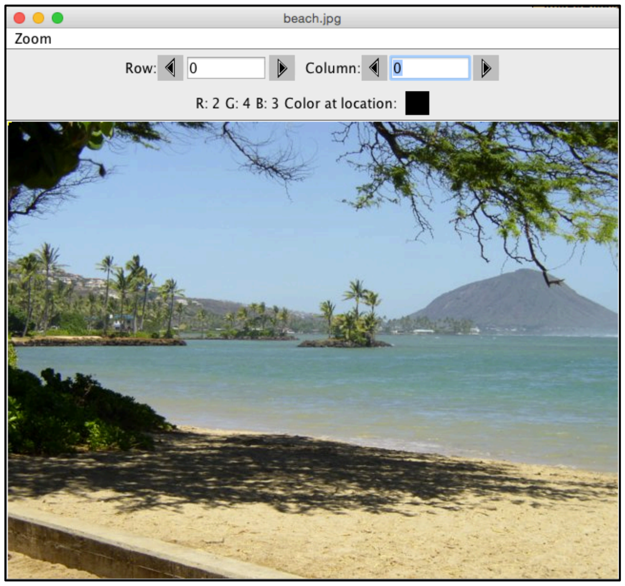


Figure 4 – The Picture Explorer

1) What is the row index for the top left corner of the picture?

2) What is the column index for the top left corner of the picture?

3) The width of this picture is 640. What is the right most column index?

4) The height of this picture is 480. What is the bottom most row index?

5) Does the row index increase from left to right or top to bottom?

6) Does the column index increase from left to right or top to bottom?

Set the zoom to 500%. Can you see squares of color? This is called *pixelation*. Pixelation means displaying a picture so magnified that the individual pixels look like small squares.

Creating and Exploring Other Pictures

Here is the `main` method in the class `PictureExplorer`. Most classes in Java can have a `main` method, and it is where execution starts when you execute the command `java` *`ClassName`* in a command line interface like Terminal (like when I randomly sort you into groups). Think of `main` as the method that executes a class.

```
public static void main( String args[])
{
  Pic pix = new Pic("beach.jpg");
  pix.explore();
}
```

The body of the `main` method declares a reference to a `Pic` object named `pix` and sets that variable to refer to a `Pic` object created from the data stored in a JPEG file named "`beach.jpg`" in the `images` folder. A JPEG file is one that follows an international standard for storing picture data using *lossy compression. Lossy compression* means that the amount of data that is stored is much smaller than the available data, but the part that is not stored is data we won't miss. Lossy compression techniques are often used with streaming media services such as Netflix and Pandora.

7) Modify the `main` method in the `PictureExplorer` class (it can be found at the very bottom of the class) to create and explore a different picture in the images folder.

8) Add your own picture to the `images` folder and then create and explore that picture in the `main` method. If the picture is very large (for instance, one from a digital camera), you can scale it using the `scale` method in the `Pic` class.

   For example, you can make a new picture ("`myPictureSmall.jpg`" in the `images` folder) one-fourth the size of the original ("`myPicture.jpg`"), save it to your `images` folder, and then explore it using:

```
  Pic original = new Pic("myPicture.jpg");
  Pic editedPic = original.scale(0.25,0.25);
  editedPic.write("myPictureSmall.jpg");
  editedPic.explore();
```