2D Arrays & Images

Introduction

- Images are 2D arrays of Pixels
- Pixels are integers values between `0` and 255

Loading an Image into a 2D array

```
int[][] img = ImageData.load("myImage.jpeg");
```

- `ImageData` is a library for reading an image into a 2D array of `int` values.
- `load()` takes the name of a file containing an image and returns an `int[][]`
  - Can you guess how many *rows* are in the output array? How many *columns*?

Contents of the 2D array

```
int[][] img = ImageData.load("myImage.jpeg");
```

- Returns a `int[][]` with dimensions that match the input image:
  - `img.length` is the number of rows & the vertical height of the image
  - `img[0].length` is the number of columns & the horizontal width of the image

Each pixel is represented as an `int` in `img`. The `int` values represent the alpha, red, green, and blue values of each color.

When expanded into a *binary string*, the int (a whole 32-bit number) looks like this:

```
11100001011000011001100111111101101
```

Contents of the 2D array

```
int[][] img = ImageData.load('myImage.jpeg');
```

Each pixel is represented as an `int` in `img`. The `int` values represent the alpha, red, green, and blue values of each color.

If we break the 32-bit number into 4 groups of 8 bits and convert the values to decimal, we can see the values (0-255) for each color:

	alpha	red	green	blue
binary	11100010	11000011	00110011	11101101
decimal	226	195	51	237

This text is written in the color represented by the above int.

Contents of the 2D array

You don't have to worry about the math, but given an int representing a pixel, then:

```
int red = ImageManipulator.getRed(pixel); // given a pixel, get the red value
int green = ImageManipulator.getGreen(pixel); // get the green value
int blue = ImageManipulator.getBlue(pixel);  // get the blue value

// turn a number between 0-255 into a full 32 bit grayscale color.
int fullGray = ImageManipulator.toGrayscaleRGB(intensity);
```

Display an Image

`ImageData.show()` takes in an `int[][]` and displays it as an image.

```
int[][] img = new int[300][300];
for (int i = 0; i < 300; i++) {
    for (int j = 0; j < 300; j++) {
        int grayColor = (int) (255 * Math.sqrt(i * i + j * j) / 425);
        img[i][j] = ImageManipulator.toGrayscaleRGB(grayColor);
    }
}
ImageData.show(img);
```

Image Manipulations

Black and White

- Load the image as an `int[][]`
- Create a new `int[][]` of the same dimensions to copy into
- For each pixel, calculate its average *intensity*: average of the red, green, and blue values
- Use this intensity to calculate the grayscale value of the pixel and copy into output array

Image Manipulations

Vertical Flip

- Load the image as an `int[][]`
- Create a new `int[][]` of the same dimensions to copy into
- For each **row** in the image, copy the row into the new 2D in reverse order

Image Manipulations

Vertical Flip

- Given a rectangular `int[][]` representing an image, create a new int 2D array with the same dimensions
- The orders of the rows in the new array should be reversed
- The values within the rows should have the same order, but *row 0* should be *row (n-1)* in the flipped array, *row 1* should be *row (n-2)*, and so on