# 2D Image Lab Team Code

*Use this document to combine solutions and specify who did each method. Once completed, **1 member** should submit this document and **1 member** should combine all the code into an Images.java file.*

Team Member 1: Chi-Chi Nwosu
Team Member 2: Tomiwa Sodeinde
Team Member 3: Sahithra K

| **Team Member** | **#1 Negative - Java Code** |
|---|---|
| Tomiwa Sodeinde | public GImage negative(GImage source) {<br>int [][] pic = source.getPixelArray();<br>int height = pic.length;<br>int width = pic[0].length;<br>// new array for neg pixels<br>int[][] invert = new int[height][width];<br><br>for(int i = 0; i< pic.length; i++) {<br><br>for(int j = 0; j<pic[0].length; j++) {<br>// gets og color of pixel<br>int b = GImage.getBlue(pic[i][j]);<br>int g = GImage.getGreen(pic[i][j]);<br>int r = GImage.getRed(pic[i][j]);<br>//Inverse values are stored in the variables<br>r = 255-r;<br>g = 255-g;<br>b = 255-b;<br>// make the pixel a variable<br>invert[i][j] = GImage.createRGBPixel(r, g, b);<br>// puts pixel in new spot<br><br>}<br>}<br>// printing a new image<br>GImage neg = new GImage(invert);<br>return neg;<br><br>} |

| Team Member | #2 Green Screen - Java Code |
|---|---|
| Tomiwa Sodeinde | public GImage greenScreen(GImage source) {<br>int[][] pixels = source.getPixelArray();<br>int width = pixels[0].length;<br>int height = pixels.length;<br><br>// loops through each pixel in the image<br>for (int row = 0; row < height; row++) {<br>for (int col = 0; col < width; col++) {<br>// gets rgb of pixel<br>int pixel = pixels[row][col];<br>int red = GImage.getRed(pixel);<br>int green = GImage.getGreen(pixel);<br>int blue = GImage.getBlue(pixel);<br><br>// checks if it's green enough to be transparent… check on eclipse…<br>if (green >= 2 * Math.max(red, blue)) {<br>pixel = GImage.createRGBPixel(red, green, blue, 0);<br>pixels[row][col] = pixel;<br>}<br>}<br>}<br><br>// create a new image w the green-screened pixels<br>return new GImage(pixels);<br>} |

| Team Member | #3 Rotate Left - Java Code |
|---|---|
| Sahithra K | public GImage rotateLeft(GImage source) {<br><br>int [][] arr = source.getPixelArray();<br>int Rows = arr.length; //Getting the number of rows in the original array<br>int Cols = arr[0].length; //Getting the number of columns in the original array<br>int [][] newArr = new int [Cols][Rows]; //Making a new array to swap the columns and rows so the dimensions of the image are swapped<br><br>for(int r= 0; r < Rows; r++) //Going through the new rows and columns in the new array<br>{<br>for(int c = 0; c < arr[0].length; c++) |

| | |
|---|---|
| | {<br>newArr [Cols-1-c][r] = arr[r][c]; //Changing the location of the pixels. New row location is the old column location. The new column location is to the left of the original location<br><br>}<br>}<br><br>GImage newImage = new GImage(newArr); //Making a new image with the new rotated array<br>return newImage; // returning the new image<br>} |

**Team Member** **#4 Rotate Right - Java Code**

| | |
|---|---|
| Chi-chi Nwosu | public GImage rotateRight(GImage source) {<br>int [][] arr = source.getPixelArray();<br>int Rows = arr.length; //Getting the number of rows in the original array<br>int Cols = arr[0].length; //Getting the number of columns in the original array<br>int [][] newArr = new int [Cols][Rows]; //Making a new array to swap the columns and rows so the dimensions of the image are swapped<br><br>for(int r= 0; r < Rows; r++) //Going through the new rows and columns in the new array<br>{<br>for(int c = 0; c < arr[0].length; c++)<br>{<br>newArr [c][Rows-1-r] = arr[r][c]; //Changing the location of the pixels. New row location is the old column location. The new column location is to the left of the original location<br><br>}<br>}<br><br>GImage newImage = new GImage(newArr); //Making a new image with the new rotated array<br>return newImage; // returning the new image |

| | |
|---|---|
| | }

| Team Member | #5 Flip Horizontal - Java Code |
|---|---|
| Chi-Chi Nwosu | public GImage flipHorizontal(GImage source) {<br>  int[][] pic = source.getPixelArray();<br>  int height = pic.length;<br>  int width = pic[0].length;<br><br>  int[][] flipparoo = new int[height][width];  // new array for flipped pixels<br><br>  for (int i = 0; i < height; i++) {<br>  for (int x = 0; x < width; x++) {<br>  // Calculate the new column index for the flipped pixel<br>  int col = width - 1 - x;<br><br>  // Copy pixel color to the new position in the flipped array<br>  flipparoo[i][newColumn] = GImage.createRGBPixel(<br>  GImage.getRed(pic[i][x]),<br>  GImage.getGreen(pic[i][x]),<br>  GImage.getBlue(pic[i][x])<br>  );<br>  }<br>  }<br><br>  // Create a new GImage using the flipped pixel array<br>  GImage flipping = new GImage(flipparoo);<br>  return flipping;<br>} |

| Team Member | #6 Translate - Java Code |
|---|---|
| Sahithra K | public GImage translate(GImage source, int dx, int dy) {<br>  int[][] img = source.getPixelArray(); //Getting the og array<br>  int height = img.length; //Getting the height (rows) of the og array<br>  int width = img[0].length; //Getting the length (columns) of the og array<br>  int[][] newImg = new int[height][width]; //Making a new array with the og array's dimensions<br><br>  for(int r = 0; r < height; r++) {<br>  for(int c= 0; c<width; c++) { |

| | |
|---|---|
| | int Rows = (r+dy+height) % height; //Calculating the new pixel positions by the dy and dx of what the user puts in<br>int Cols = (c+dx+width) % width;<br>newImg [Rows][Cols] = img[r][c];<br>}<br>}<br>return new GImage(newImg); //Returning the new translated image<br><br>}<br> |

| Team Member | #7 Blur - Java Code |
|---|---|
| chi-chi nwosu | public GImage blur(GImage source) {<br>int[][] pic = source.getPixelArray();<br>int width = pic[0].length;<br>int height = pic.length;<br><br>int[][] blurry = new int[height][width];<br><br>for (int i = 0; i < height; i++) {<br>for (int j = 0; j < width; j++) {<br>// inside the loop so it can reset each pixel<br>int allR = 0;<br>int allG = 0;<br>int allB = 0;<br>int count = 0; // keeps track of the number of neighboring pixels<br><br>for (int die = -1; die <= 1; die++) {<br>for (int djo = -1; djo <= 1; djo++) {<br>int nig = i + die; // neighboring row index<br>int nje = j + djo;// neighnoring column index<br><br>// Check if the neighbor pixel is within the image boundaries<br>if (ni >= 0 && ni < height && nj >= 0 && nj < width) {<br>int pixel = pic[ni][nj];<br>// adding tg<br>allR += GImage.getRed(pixel);<br>allG += GImage.getGreen(pixel);<br>allB += GImage.getBlue(pixel);<br>count++;<br>}<br>}<br>}<br><br>// average values set them in the blurred image array |

| | |
|---|---|
| | int avgR = allR / count;<br>int avgG = allG / count;<br>int avgB = allB / count;<br>blurry[i][j] = GImage.createRGBPixel(avgR, avgG, avgB);<br>}<br>}<br><br>// Create a new GImage from the blurred image array<br>GImage cantSee = new GImage(blurry);<br>return cantSee;<br>} |

| Team Member | BONUS Equalize - Java Code |
|---|---|
| | |