Image Processing and Steganography

Melvyn Ian Drag

October 16, 2019

Abstract

In this lecture we will learn about how computers store images and what RGB values are. We will warm up by doing simple image manipulation tasks. We will then use what we know about a few Java classes, bit manipulation, and text encodings to hide a message in an image.

1 Introduction

I will show you two examples that motivate the discussion we will have and the exercises you will perform for the next few hours.

1.1 Green Screen

Show how a program can filter out certain pixels in an image and make them transparent. This is how they make movies - the actors stand in front of a green screen and the computer eliminates the green stuff and replaces it with a battle scene or whatever. Something like this: `https://www.youtube.com/watch?v=TrgQuJJxRW4`

Now we don't have time to make a blockbuster movie this evening, but we can learn a little bit about how you might do it if you wanted to.

Show Code/Greenman/greenman.png *Actually I was pressed for time so I'm going to give you the inverse example. In this image I have a greenman, taken from a men's restroom logo. We can make the greenman disappear and turn into transparent pixels.*

Then run the Code/Greenman/Greenman code and show the output image. Then open it with Inkscape and show that the image is transpare t where there was a Green person before. Note there is some bugginess in the way I did this. That is because I'm lazy with the way I wrote the code. All I wanted to show you was that the image was transparent and whatever glaring imperfections you might see are minor imperfections. The concept is clear - we are able to turn some pixels in an image transparent. If you want to help me perfect my example I'd really appreciate it.

1.2 Steganography

As I've mentioned before, you can hide text in the pixels of an image. See these two pictures? They look the same. And while one of them is just a boring old image, the other one contains text. I have concealed a java program in the second file. Of course, you could conceal whatever text you wanted. In this file, I have taken the last 2 bits from every pixel in the image and stored two bits of data in them. The java program I concealed is encoded in UTF-8. I won't tell you what it does just yet, I'll just tell you that it's there. In fact. I fell that what I've just said might be enough to end lecture just now and send you all of on missions to discover the hidden treasure. Nevertheless, we'll work through this together, because I have another task planned for you for your midterm.

2 RGBA / ARGB

In grammar school they told us that the primary colors are:

1. Red
2. Blue
3. Yellow

That's one of the many half truths that we tell children to help them understand the world without giving them too much detail that would be impossible for their young minds to understand. In computer science we look at the primary colors as Red, blue, green. If you want to read more about the differences between the RGB and RBY systems, you can go on google as we don't have time for that now. In computer science we also say that images have an alpha channel that measures how opaque/transparent a pixel in an image is. So, while there is more to the story, we will agree right now that images are made up of 4 channels:

1. Red
2. Green
3. Blue
4. Alpha

3 Loading images in Java

Show the code that loads images and point out some relevant parts of it. This program simply reads a png image into memory and then writes it back out to a file.

```
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.io.IOException;
import java.io.File;
import javax.imageio.ImageIO;

public class ReadAndWritePNG{

  public static void main(String[] args) throws IOException {
    BufferedImage hugeImage =
       ImageIO.read(ReadAndWritePNG.class.getResource("greenman.png"));
    int[][] result = null;
    try{
      result = convertToIntArray(hugeImage);
    }
    catch(NoAlphaChannelException ex){
      System.err.println(ex.getMessage());
      return;
    }
    writeImage(result);
  }

  private static int[][] convertToIntArray(BufferedImage
     image) throws NoAlphaChannelException{
    final byte[] pixels = ((DataBufferByte)
       image.getRaster().getDataBuffer()).getData();
    final int width = image.getWidth();
    final int height = image.getHeight();
    final boolean hasAlphaChannel = image.getAlphaRaster() !=
       null;
    if( !hasAlphaChannel ){
      throw new NoAlphaChannelException("This simple code can
         only handle pngs with an alpha channel. Yours has
         none.");
    }
    int[][] result = new int[height][width];
    final int pixelLength = 4;
    for (int pixel = 0, row = 0, col = 0; pixel + 3 <
       pixels.length; pixel += pixelLength) {
      int argb = 0; // argb = Alpha,Red, Green, Blue
      int blue = ((int) pixels[pixel + 1] & 0xff);
      int green = (((int) pixels[pixel + 2] & 0xff) << 8);
      int red = (((int) pixels[pixel+3] & 0xff ) << 16 );
      int alpha =  (((int) pixels[pixel] & 0xff) << 24);
      argb += blue;
      argb += green;
      argb += red;
      argb += alpha;
      result[row][col] = argb;
      col++;
      if (col == width) {
        col = 0;
        row++;
      }
    }
    return result;
  }

  public static void writeImage(int[][] color) {
    String path = "output.png";
    BufferedImage image = new BufferedImage(color[0].length,
       color.length, BufferedImage.TYPE_INT_ARGB);
    for (int x = 0; x < color.length; x++) {
      for (int y = 0; y < color[x].length; y++) {
        image.setRGB(y,x, color[x][y]);
      }
    }
    File ImageFile = new File(path);
    try {
      ImageIO.write(image, "png", ImageFile);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

}

class NoAlphaChannelException extends Exception{
  NoAlphaChannelException(String s){
    super(s);
  }
}
```

4 Turn Image Red

To turn a png red all we have to do is access all the pixels and make the red channel 0xff, while we make the green and blue channels 0x00.

```
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.io.IOException;
import java.io.File;
import javax.imageio.ImageIO;
```

```
public class TurnImageRed{

  public static void main(String[] args) throws IOException {
    BufferedImage hugeImage =
       ImageIO.read(TurnImageRed.class.getResource("greenman.png"));
    int[][] result = null;
    try{
      result = convertToModifiedArray(hugeImage);
    }
    catch(NoAlphaChannelException ex){
      System.err.println(ex.getMessage());
      return;
    }
    writeImage(result);
  }

  private static int[][] convertToModifiedArray(BufferedImage
      image) throws NoAlphaChannelException{
    final byte[] pixels = ((DataBufferByte)
       image.getRaster().getDataBuffer()).getData();
    final int width = image.getWidth();
    final int height = image.getHeight();
    final boolean hasAlphaChannel = image.getAlphaRaster() !=
       null;
    if( !hasAlphaChannel ){
      throw new NoAlphaChannelException("This simple code can
         only handle pngs with an alpha channel. Yours has
         none.");
    }
    int[][] result = new int[height][width];
    final int pixelLength = 4;
    for (int pixel = 0, row = 0, col = 0; pixel + 3 <
       pixels.length; pixel += pixelLength) {
      int argb = 0; // argb = Alpha,Red, Green, Blue
      int blue = 0x00;
      int green = 0x00;
      int red = 0x00FF0000 ;
      int alpha = 0xFF000000;
      argb += blue;
      argb += green;
      argb += red;
      argb += alpha;
      result[row][col] = argb;
      col++;
      if (col == width) {
        col = 0;
        row++;
      }
    }
    return result;
  }

  public static void writeImage(int[][] color) {
    String path = "output.png";
    BufferedImage image = new BufferedImage(color[0].length,
       color.length, BufferedImage.TYPE_INT_ARGB);
    for (int x = 0; x < color.length; x++) {
      for (int y = 0; y < color[x].length; y++) {
        image.setRGB(y,x, color[x][y]);
      }
    }
    File ImageFile = new File(path);
    try {
      ImageIO.write(image, "png", ImageFile);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

}

class NoAlphaChannelException extends Exception{
  NoAlphaChannelException(String s){
    super(s);
  }
}
```

5 Turn Image Green

To turn a png green, all we need to do is accell all the pixels in the image and turn the green channel to 0xff, while we make the red and blue channels 0x00.

```
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.io.IOException;
import java.io.File;
import javax.imageio.ImageIO;

```

```
public class TurnImageGreen{

  public static void main(String[] args) throws IOException {
    BufferedImage hugeImage =
       ImageIO.read(TurnImageGreen.class.getResource("greenman.png"));
    int[][] result = null;
    try{
      result = convertToModifiedArray(hugeImage);
    }
    catch(NoAlphaChannelException ex){
      System.err.println(ex.getMessage());
      return;
    }
    writeImage(result);
  }

  private static int[][] convertToModifiedArray(BufferedImage
     image) throws NoAlphaChannelException{
    final byte[] pixels = ((DataBufferByte)
       image.getRaster().getDataBuffer()).getData();
    final int width = image.getWidth();
    final int height = image.getHeight();
    final boolean hasAlphaChannel = image.getAlphaRaster() !=
       null;
    if( !hasAlphaChannel ){
      throw new NoAlphaChannelException("This simple code can
         only handle pngs with an alpha channel. Yours has
         none.");
    }
    int[][] result = new int[height][width];
    final int pixelLength = 4;
    for (int pixel = 0, row = 0, col = 0; pixel + 3 <
       pixels.length; pixel += pixelLength) {
      int argb = 0; // argb = Alpha,Red, Green, Blue
      int blue = 0x00;
      int green = 0x0000FF00;
      int red = 0x00 ;
      int alpha = 0xFF000000;
      argb += blue;
      argb += green;
      argb += red;
      argb += alpha;
      result[row][col] = argb;
      col++;
      if (col == width) {
        col = 0;
        row++;
      }
    }
    return result;
  }

  public static void writeImage(int[][] color) {
    String path = "output.png";
    BufferedImage image = new BufferedImage(color[0].length,
       color.length, BufferedImage.TYPE_INT_ARGB);
    for (int x = 0; x < color.length; x++) {
      for (int y = 0; y < color[x].length; y++) {
        image.setRGB(y,x, color[x][y]);
      }
    }
    File ImageFile = new File(path);
    try {
      ImageIO.write(image, "png", ImageFile);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

}

class NoAlphaChannelException extends Exception{
  NoAlphaChannelException(String s){
    super(s);
  }
}
```

6 Exercise: Turn Image Blue

To test your understanding I want you to now turn an image blue, using the provided code an modifying it as appropriate.

7 What's in a PNG?

Show a HDD, SDD, Ram Sticks, SD Card, USB Stick *All of these devices store computer data as bytes. The hard drive has magnets inside, the ssd, sd card, usb stick work a bit differently. The Ram works still differently, and there are many types of RAM. You might have heard of DDR3 and DDR4 if you've built a Gaming PC or something.*

Computers store files as bytes. Collections of 1s and 0s. Then, the computer is taught to understand these bytes in a specific way when it reads them. RAM, for example, might store some 1s and 0s as high and low voltages, and then features some sort of circuitry that will understand highs and lows. This is an enormously challenging task, and thanks to Electrical Engineers, we software developers don't have to understand how all that works.

I've already shown you one example of how a the bits in a file can be interpreted in different ways according to what you teach your computer.We saw this via our exploration of UTF-8, ASCII and UTF-16 encodings. Now we will see a different binary format. So far we have seen hte binary formats for text and numbers. We will now take a breif look at the binary format for PNG images.

Show this link: `https://en.wikipedia.org/wiki/Portable_Network_Graphics`. Show the section about header data. This describes the first few bytes a computer should expect when reading a png file. Commisions of professional prgrammers have gotten together at some point in history to decide that this sequence of bytes is what should be stored in the beginning of a png file. And if we look at a few png files, we'll see that they all satisfy the sequence of bytes specified by the standard.

Take a few minutes to look at the first few bytes of some png files with xxd.

8 Green Screen

Change the alpha value to zero for any pixel that is thoroughly green. This is (I think) how hollywood does it. I'm not sure since I never made a movie, but I have heard about green screens and my guess is that this is how they work. If there are some pixels in an image that are green and the rest are not, you can "easily" remove the green ones. I say "easily" because there are a bunch of pther considerations. The "greenness" recorded in an image depends on external factors like the camera, lighting, shadows, etc. I would guess that they would want to eliminatie shadows.

Have a look now at the Greenman code. See how it works. Run it and tweak the threshold values on the greenness to make it work better/worse.

```java
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.io.IOException;
import java.io.File;
import javax.imageio.ImageIO;

public class Greenman{

  public static void main(String[] args) throws IOException {
    BufferedImage hugeImage =
       ImageIO.read(Greenman.class.getResource("greenman.png"));
    int[][] result = null;
    try{
      result = convertToModifiedArray(hugeImage);
    }
    catch(NoAlphaChannelException ex){
      System.err.println(ex.getMessage());
      return;
    }
    writeImage(result);
  }

  private static int[][] convertToModifiedArray(BufferedImage
     image) throws NoAlphaChannelException{
    final byte[] pixels = ((DataBufferByte)
       image.getRaster().getDataBuffer()).getData();
    final int width = image.getWidth();
    final int height = image.getHeight();
    final boolean hasAlphaChannel = image.getAlphaRaster() !=
       null;
    if( !hasAlphaChannel ){
      throw new NoAlphaChannelException("This simple code can
         only handle pngs with an alpha channel. Yours has
         none.");
    }
    int[][] result = new int[height][width];
    final int pixelLength = 4;
    for (int pixel = 0, row = 0, col = 0; pixel + 3 <
       pixels.length; pixel += pixelLength) {
      int argb = 0; // argb = Alpha,Red, Green, Blue
      int blue = ((int) pixels[pixel + 1] & 0xff);
      int green = (((int) pixels[pixel + 2] & 0xff) << 8);
      int red = (((int) pixels[pixel+3] & 0xff ) << 16 );
      int alpha = 0;
      if (( (green>>8) > 100 ) && (( red>>16) < 100 ) ){
        alpha = 0x00;
      }
      else{
        alpha =  (((int) pixels[pixel] & 0xff) << 24);
      }
      argb += blue;
      argb += green;
      argb += red;
      argb += alpha;
      result[row][col] = argb;
      col++;
      if (col == width) {
        col = 0;
        row++;
      }
    }
    return result;
  }

  public static void writeImage(int[][] color) {
    String path = "output.png";
    BufferedImage image = new BufferedImage(color[0].length,
       color.length, BufferedImage.TYPE_INT_ARGB);
    for (int x = 0; x < color.length; x++) {
      for (int y = 0; y < color[x].length; y++) {
        image.setRGB(y,x, color[x][y]);
      }
    }
    File ImageFile = new File(path);
    try {
      ImageIO.write(image, "png", ImageFile);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

}

class NoAlphaChannelException extends Exception{
  NoAlphaChannelException(String s){
    super(s);
  }
}
```

if there is time find footage of moviemakers recording something with a green screen. Do they use a bunch of lights to minimize shadows and try to make the background somewhat equally green?

9 Hide a message in a PNG

9.1 Hiding Hi

Now this section is motivated by a video I saw on the computerphile channel a few months ago. I was preparing for this class over the summer and looking for fun stuff to do with Java and this just seemed irresistable. Check out the video here if you want: `https://www.youtube.com/watch?v=TWEXCYQKyDc` The process is relatively simple.

I will try to hide the message "Hi" in this file. I Will encode "Hi" in UTF-16, which will be 0x00, 0x48, 0x00, 0x69. To do this I will encode the bits in the last bit of the blue channel of the image. For every pixel, I will OR the red channel with 0x00 if the bit is zero. If the bit is 1 I will OR the red channel with 0x01. Then I will save the image. In UTF-16, "Hi" takes up 32 bits.

9.2 Weird stuff happening.

Note that the md5sum changes for the image the first time I run the Code/Steganography/backup example. Then when I run the output through it again and again the md5sum stops changing. The first iteration must truncate the data somehow.

We are storing the rgb values in 8 bits, but I suspect the original input has more bits than that.

10 PPM Format

10.1 Reading a PPM File

I won't show you how to do this 100% because that is a bit of code you need to write. I didn't want to leave you with nothing in case you were confused, so See Code/ReadByteFile

```java
import java.io.File;
import java.io.*;
import java.nio.charset.Charset;
import java.util.Arrays;

public class ReadByteFile{
  public static void main(String[] args) throws Exception{
    File f = new File("ByteFile.txt");
    byte[] fileArr = new byte[(int) f.length()];
    InputStream is = new BufferedInputStream(new
       FileInputStream(f));
    for( int i = 0, read_value = 0; ( read_value = is.read() )
       != -1; i++ ){
      fileArr[i] = (byte)read_value;

    }
    System.out.println("What bytes were read from the file?");
    for( byte i : fileArr ){
      System.out.println(String.format("0x%08X", i ) );
    }

    String fileString = new String(fileArr,
       Charset.forName("UTF-8"));
    System.out.println("What do those bytes translate to as
       text?");
    System.out.println(fileString);
    String[] allNumbers = fileString.split("\\s+");
    int[] ints = new int[allNumbers.length];
    int currIdx = 0;
    Arrays.fill(ints, 256); // should be no 256 in our arrays.
    for( String s : allNumbers ){
      try{
        int i = Integer.parseInt(s);
        ints[currIdx] = i;
        currIdx++;
      }
      catch(Exception e){
        System.out.println("Ignore this, can't convert to int:
           " + s );
      }
    }

    System.out.println("What does that text translate to as
       integers?");
    for( int i : ints ){
      System.out.println(String.format("0x%08X", i));
    }

    System.out.println("Can I set the last bit in every int to
       0?");
    for( int i: ints ){
      System.out.println(String.format("0x%08X", (i &
         0xFFFFFFFE)));
    }




  }
}
```

11 A look at the classes used in this class

Check out the TYPE_INT_BLAH options

What is ImageIO

What is a buffered image?

12 Pass by Value vs. Pass by Reference

A basic thing you will always need to know when coming to a new language is - 'are arguments to functions in this language passed by value, or by reference?' In Java, function arguments are passed by value.

Java is Strictly pass by Value. Repeat after me 10 times: 'Java is pass by value'.

12.1 Read this

1. https://www.geeksforgeeks.org/g-fact-31-java-is-strictly-pass-by-value/
2. https://www.javaworld.com/article/2077424/learn-java-does-java-pass-by-reference-o
   html
3. https://www.journaldev.com/3884/java-is-pass-by-value-and-not-pass-by-reference

12.2 Example Code

Fill out two examples. have students do the third

```
public class ByValueByRef{

  public static void printTestHeader(String s){
    System.out.println("\n\n*********************************************
    System.out.println(s);
    System.out.println("*********************************************
  }

  public static void f( int[][] arr ){
    arr[0][0] = 100;
  }

  public static void TestIntArr(){
    printTestHeader("Are int arrays passed to void functions
       changed in the functions they are passed to?");
    int[][] arr = {{1,2},{3,4}};
    System.out.println("Array before going to function");

    for(int[] row : arr){
      for( int col : row ){
        System.out.print(col + " ");
      }
      System.out.println("");
    }

    f(arr);
    System.out.println("Array after going to function");
    for(int[] row : arr){
      for( int col : row ){
        System.out.print(col + " ");
      }
      System.out.println("");
    }
  }

  // Do this in class
  public static void TestInt(){
    // Create an int, pass it to a function that changes it's
       value, then
  }

  // Do this in class
  public static void TestObject(){
    // Create a 'Container' object adn modify it in a function.
    // After leaving the function has the object changed?
  }

  public static void main(String[] args){
    TestInt();
    TestIntArr();
    TestObject();
  }
}


class Container{
  public int x;

  Container(int i){ x = i;}
}
```

13 Midterm Exam Discussion

See the exam write up in the Exams folder of this repository. Go over very carefully with class.

14 References

1. https://en.wikipedia.org/wiki/Netpbm_format

2. http://www.paulbourke.net/dataformats/ppm/