Assignment 5– Color Blindness Simulator

Kevin Gallegos

CSE 13S – Spring 2023

Purpose

The purpose of this assignment is meant to simulate deuteranopia, a type of color blindness which impairs someone from being able to see any sort of green. In this program we will take BMP images and convet them into an image lacking certain colors to simulate deuteranopia.

How to Use the Program

To use the program make sure to run colorb.c by compiling it first and using commands. This program uses command line prompts and while be used through the method of entering this in your command line "./sorting -(command)" Below is the list of every command that can be used in this program. "-i" sets input file, requires filename as argument "-o" sets output file, requires filename as argument "-h" prints a help message.

Program Design

The program will use command line options to initialize any necessary loops for the functions to work and take an input and or output as specified by the command line options as well as a help menu to help guide those for command line options. The program will simply take a input and a output, having the input ran through a buffer and manipulated to change colors on the image. The ouput will then obtain this modified image and export it as the input name specified.

Data Structures

The program will use read open to read from optarg that was inputted during command line options. From here a buffer will be created to store the information from the image. Soon a bmp struct is created to organize the information and pass it through a function that reduces the pallet of the image. The user would've also specified an output file using command line options in which after the pallet reduction will be then ran through the write function, constructing it back into an image and exported under the filename specified under the command line for output.

Algorithms

```
io.h
    buffer create struct
        int fd, offset, int a
        a[buffer size]

    Define Buffer read func
        open file, if <0 retunr NULL
        //define struct variables
        fd, offset, num_remaining
        return b;
    Define read close func
        close buffer pointer
        buffer pointer = NULL
    Define read uint8 func
        check if num remaining == 0, if so:
            rc = read(fd, a, a)
            check if rc<0 if so print to stderr
            check if rc = 0 is so, return false
            num remaining = fc
            offset = 0
        a[offset] = x;
        offset++

    Define read uint16 func
        var 1 =read uint8(buf x)//make sure it returns true
        var 2 = read uint8(buf x)//make sure it returns true
        left shift var 2 by 8
        var2 = var1 or var2
        x = var2
        return true
    Define read uint32
        var 1 =read uint16(buf x)//make sure it returns true
        var 2 = read uint16(buf x)//make sure it returns true
        left shift var 2 by 16
        var2 = var1 or var2
        x = var2
        return true
    Define write uint8
        if offset == buffer size:
            create pointer start = a
            create int num bytes = offset
            do:
                rc = write(fd, start, num bytes)
                check if rc<0 if so, print to stderr
                start += rc
                num bytes -= rc
            while(num bytes > 0)
            offset = 0
        a[offset] = x
        offset++
    Define write uint 16 func
        write_uint8(buf, x)
        write_uint8(buf, (x right shift by 8))
    Define write uint 32 func
        write_uint8(buf, x)
        write_uint8(buf, (x right shift by 16))
```

```
bmp.c
    bmp_create func:
        read_uint8(type1)
        read_uint8(type2);
        read_uint32(skip2);
        read_uint16(skip1);
        read_uint16(skip1);
        read_uint32(skip2);
        read_uint32(bitmap_header_size);
        read_uint32((bmp)->width);
        read_uint32(bmp->height);
        read_uint16(skip1);
        read_uint16(bits_per_pixel=0);
        read_uint32(compression);
        read_uint32(skip2);
        read_uint32(skip2);
        read_uint32(skip2);
        read_uint32(colors_used);
        read_uint32(skip2);
        num colors = colors used
        if num colors == 0, num colors = (1 left shift by bits per pixel)
        for (i < num_colors-1)
            read_uint8(palette[i].blue);
            read_uint8(palette[i].green);
            read_uint8(palette[i].red);
            read_uint8(skip);
        rounded_width = (bmp->width +3) & ~3
        bmp->a = calloc(rounded_width, sizeof(bmp->a[0])
        for x to rounded_width -1
            a[x] = calloc(height, sizeof(a[x][0])
        for y < bmp->height-1
            for x < rounded_width -1
                read_uint8(a[x][y])
        return bmp;

    bmp_write func:
        write_uint8(B)
        write_uint8(M);
        write_uint32(file size);
        write_uint16(0);
        write_uint16(0);
        write_uint32(offset);
        write_uint32(header_size);
        write_uint32(width);
        write_uint32(height);
        write_uint16(1);
        write_uint16(8);
        write_uint32(0);
        write_uint32(image size);
        write_uint32(2835);
        write_uint32(2835);
        write_uint32(colors);
        write_uint32(colors);
        for (i < num_colors-1)
            write_uint8(palette[i].blue);
            write_uint8(palette[i].green);
            write_uint8(palette[i].red);
            write_uint8(0);
        for y to height -1
            for x < width-1
```

```
                write_uint8(a[x][y]);
            for x < rounded_width -1
                write_uint8(0)
```

```
colorb.c
    initializer all vars
    allocate memory for p and copyp
    bit track is 0
    get opt:
    case 1 add 0001 to bit track with bit or
        filname = optarg(user input)
        filenameout = optarg(second user input)
        b* = read open(filename)
        bmp* = bmp_create(b)
        reduce pallette(bmp)

        wb* = write_open(bmp)
        bmp_write(wb)
        read_close(b)
        bmp free(bmp)
        write close(wb)
    case 2 add 0100 to track with bit or
        print help message
```

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- Buffer *read_open(const char *filename) Open the filename using a system call, returns null if unsuccessfull. Returns pointer to new buffer
- void read_close(Buffer **pbuf) calls close and frees the buffer
- Buffer *write_open(const char *filename) opens file using creat(), returns NULL unsuccessful, otherwise mallocs a new buffer and closes it later
- void write_close(Buffer **pbuf) Write any bytes that are in buffer a[] to the file then close, and frees buffer.
- bool read_uint8(Buffer *buf, uint8_t *x) reads the integers and refills buffer if buffer is empty then sets *x to next byte in buffer.
- bool read_uint16(Buffer *buf, uint16_t *x) deserializes uint16 and reads two bytes, copying the second byte to a uint16 and shift left by 16bits, ors the first byte into uint16. returns TRUE.
- bool read_uint32(Buffer *buf, uint32_t *x) deserializes uint16 and reads two bytes, copying the second byte to a uint32 and shift left by 16bits, ors the first byte into uint32. returns TRUE
- void write_uint8(Buffer *buf, uint8_t x) if buffer is full, calls write to empty buffer using a loop to completly empty it.
- void write_uint16(Buffer *buf, uint16_t x) Serializes uint16 x, calls write_uint8() twice
- void write_uint32(Buffer *buf, uint32_t x) Serializes uint32 x, calls write_uint16() twice
- void bmp_write(const BMP *bmp, Buffer *buf) writes a bmp file.

- BMP *bmp_create(Buffer *buf) Creates a BMP struct, reads a bmpt file to it and returns a pointer to the new struct.
- void bmp_free(BMP **bmp) frees bmp struct
- void bmp_reduce_palette(BMP *bmp) reduces the color pallete to simulate deuteranopia

Results

The program compiles and executes correctly, properly managing to simulate deuteranopia using the functions getting a file and editing the colors from the file and output the new image.

Sources used

- Jessie Tutoring sections Monday 12pm-2pm, Thursday 10pm-12pm
- Jessie Extra Office Hours, Friday 5/26 1:00pm-2:30PM