Assignment 5
Color Blindness Simulator

Eric Lee

CSE 13S – Spring 2023

Purpose

This program takes in any image in a bmp format and outputs a new image that has been color corrected. The new image simulates what the original would look like to someone with red-green color blindness, or deuteranopia. People with deuteranopia don't have green-sensitive light sensors in their eyes so they can't see that color.

How to Use the Program

To run the program, the user needs to use the makefile to compile and run the file colorb.c which is linked to two other files bmp.c and io.c. The user can use command line options with colorb.c to specify the files it will modify. The command "-i:" lets the user define the input file and "-o" lets the user define the output file. The extra command "-h" prints a help message for the user. A group of template photos can be run using the shell script cb.sh.

Program Design

Data Structures

Buffer is a struct data type that is used to store the bytes being read from the BMP file as well as an offset integer and a num_remaining integer. The offset integer keeps track of how many bytes have already been read in the buffer and num_remaining keeps track of the number of unread bytes in the buffer. It also has an integer called fd used to hold the file descriptor. The actual buffer is initialized as uint8_t a[].

The program uses two struct data types for the BMP file, one called color and the other called bmp. The struct "color" stores the rgb values from the image. BMP stores the height and width as well as the color struct mentioned earlier.

Algorithms

The read and write operations are carried out by functions that read bytes of various sizes. The sizes are 8 bits, 16 bits, and 32 bits. To read them, the 32 bit sized functions call the 16 bit sized functions twice and the 16 bit sized functions call the 8 bit sized functions twice. To store the bits together, the 16 bit sized functions and the 32 bit sized functions use bitwise functions.

The main function uses a switch case while loop to take in the command line options from the user. Then it calls the read_open function to create the read buffer from the input file, which is passed into bmp_create. The buffer is then closed with read_close and the bmp color values are modified using bmp_reduce_pallete. The write buffer is then created using write_open using the output file and bmp_write writes the new BMP image file. Finally, the bmp and buffer data is freed using bmp_free and write_close.

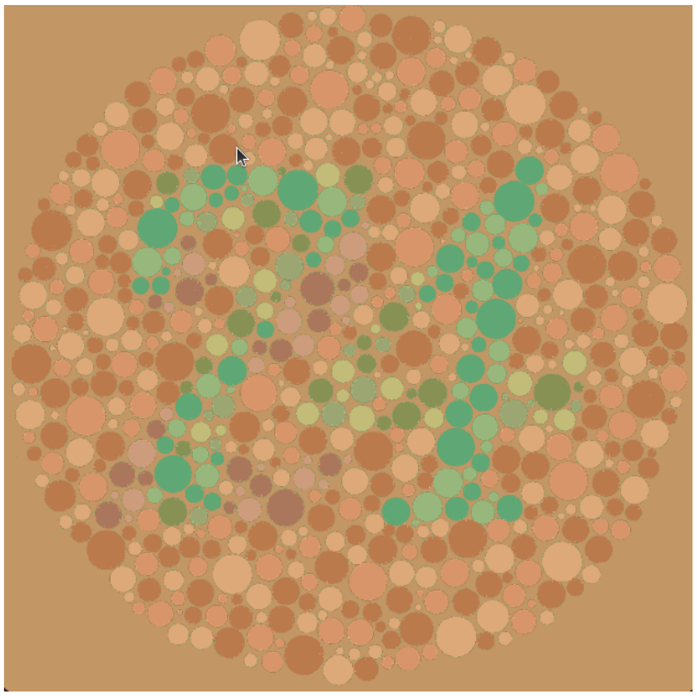
Function Descriptions

The file io.c uses Unix file-I/O functions to read from and write to files.instead of regular file-I/O functions to improve performance. The Unix functions read and write bytes of raw data instead of the formatting that the other functions do. They write and read from a buffer and return the number of bytes that are transferred. The parameters are a file descriptor, a pointer to the buffer, and the number of bytes to transfer. The file descriptor comes from using the system calls open() for reading and creat() for writing. The functions read_open, read_close, write_open, and write_close allocate and free a buffer to be read from or written to. The read_open and write_open functions take the file name as an argument and return a pointer to the newly created buffer. The read_close and write_close functions free the buffer and set its pointer to NULL.

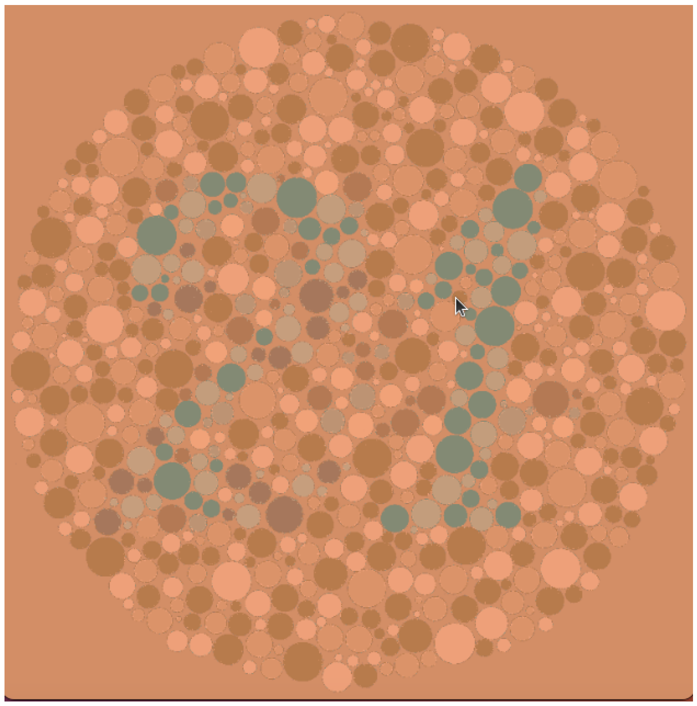
The bmp.c file uses a few functions to serialize and deserialize the BMP file as well as modify it. The function bmp_write writes a new BMP file using the buffer, and the function bmp_create creates a new BMP struct and stores the data read from a BMP file. The function bmp_reduce_palette does the actual work of modifying the rgb values from the BMP file to simulate deuteranopia. The bmp_write and bmp_create functions use the read and write functions from io.c to read and write BMP files.

Results

Everything works as it should, and the color corrected images have the exact pixel values they should according to the pipeline. The following are two before and after example pictures.



Before



After