EXPERIMENT - 1

Report Submission

<u>Topic</u> : Read, Write .bmp format images and apply geometrical transforms

GROUP - 18

Name	Sumegh Roychowdhury	Arun Sammit Pandey
Roll Number	17EC35033	17EC35006

Table of Contents

Introduction:

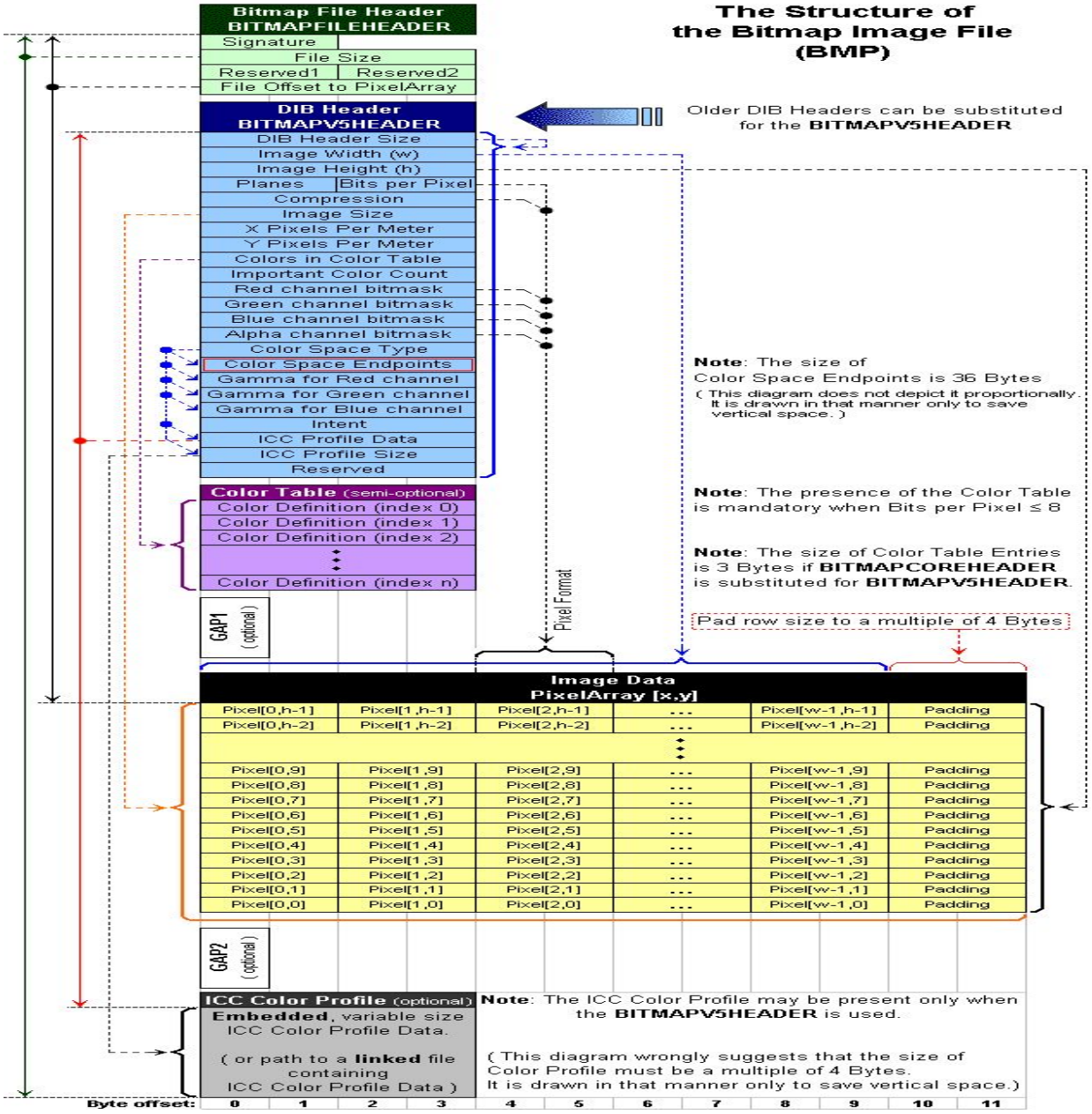
The **BMP file format** is known as the bitmap image file, **device independent bitmap (DIB) file format and bitmap**. It is a graphic image file format used to store bitmap digital images independent of the display device. It was used on early Windows OS.

It can store 2D images both in grayscale & colour along with depths. In this assignment we consider 8-bit grayscale images & 24-bit RGB images where each of the 8 bits have a depth of 3.

BitMap File Structure :

Structure Name	Optional	Size	Purpose
Bitmap file header	no	14 bytes	General info about bitmap image file
DIB header	no	Fixed	Define pixel format
Extra bitmasks	yes	12 or 16 bytes	“”
Color table	semi-optional	variable	Define colors used by image data
Gap1	yes	variable	Structure align
Pixel array	no	variable	Actual pixel values
Gap2	yes	variable	Structure align
ICC color profile	yes	variable	Color profile

BMP Structure :



Geometrical Transforms :

Grayscale :

There are 3 ways we use to convert color images to grayscale :

- **Average** : Averaging the pixel values for all 3 R, G, B channels.
- **Min** : Take the minimum pixel value of all 3 aforementioned channels.
- **Max** : Take the maximum pixel value of all 3 aforementioned channels.

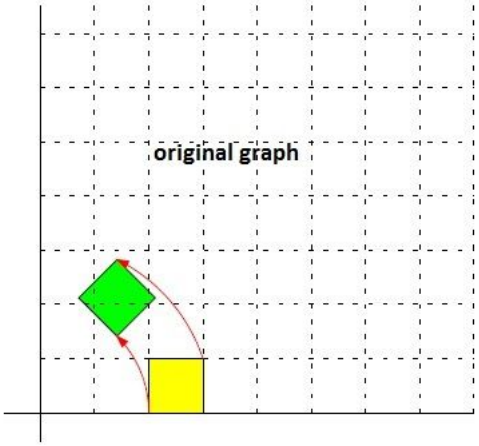
Rotation (45 & 90°):

To rotate 45° about the origin, we apply the matrix

$$\begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix} = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

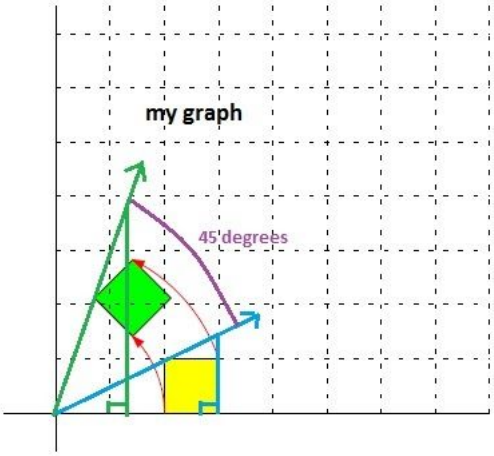
Note: $\frac{\sqrt{2}}{2} = \cos 45^\circ = \sin 45^\circ$, so this is the same as

$$\begin{pmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{pmatrix}$$

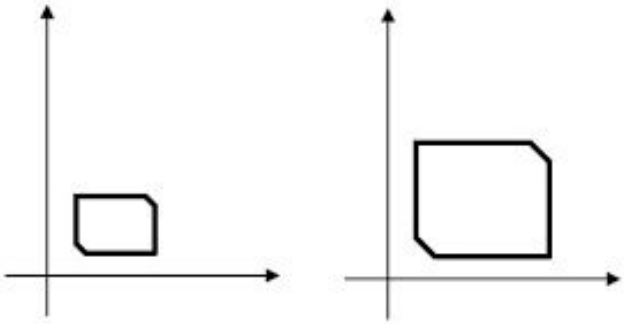


Counter Clockwise $\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$

Clockwise $\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}$



Scaling :



Algorithm :

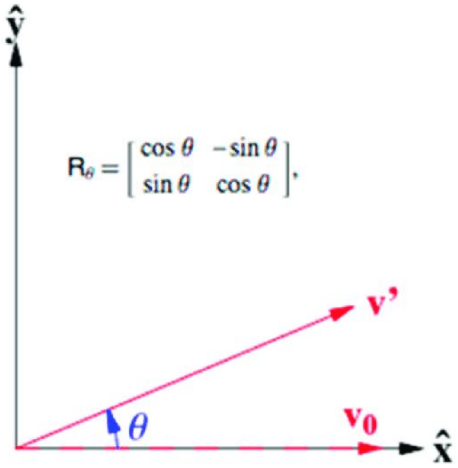
- First we define **BitMap Header** as a C++ struct.

```
struct BMPHeader{
    unsigned char file_type[2];
    unsigned int file_size;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int offset_data;
    unsigned int size;
    int width;
    int height;
    unsigned short planes;
    unsigned short bit_count;
    unsigned int compression_type;
    unsigned int image_size;
    int x_res, y_res;
    int y_res;
    unsigned int colors_used, colors_imp; };
```

- Next we **read the BMP input image** making use of the file structure as shown in Introduction.

  ```
  void read(string fimyImgme)
  ```

- We detect if it's a colour image or not based on the struct member - **bit_count**.
  if bit_count == 24:
  #Colour - stored in 3D array
  else
  #No Colour - stored in 2D array
- Next we implement the `BitMap bgr_to_gray(string mode)` function for converting color -> grayscale based on 3 options - avg, min, max of RGB channels.
- Next we implement **rotation functions** using the following **matrix transformation** :



  We do **rounding-off** of pixel indexes which serves as **Nearest-Neighbor Interpolation**. The function signatures are `BitMap rotate90clockwise()` and `BitMap rotate45clockwise()`.
- Finally we also implement the **Scaling function** as `BitMap scale(int scaleX=2, int scaleY=2)` using the same interpolation technique as in case of rotation.

Results :

Original Images :


Grayscale :


(MIN)

(AVG)


(MAX)

Rotation :


(45°)


(90°)


(90°)


(45°)

Scaling:

Flipped:


Conclusion :

In this experiment we saw how to read BMP image files based on their file structure and also save them back after processing. We also applied several geometrical transforms like scaling, flipping, rotation, etc. and used nearest-neighbor interpolation as well. All was done respecting C++11 standards and the code is modular and readable with structs defined for BitMaps.

THANK YOU
