影像處理_HW1_312513022_周聖喆

BMP Format:

BMP（Bitmap）檔案格式是一種無壓縮的圖像文件格式，檔案內容主要是由「Bitmap File Header + Bitmap Info Header + Pixel Data」所組成的。

- ### Bitmap File Header:

長度為 14 Bytes，包含有關整個檔案的基本信息。

Name	Size (Byte)	Content
ID	2	文件類型，固定為 0x4D42，即字符 'BM'
File Size	4	BMP 檔案的總大小
Reserved 1	2	保留字段，預設為 0
Reserved 2	2	保留字段，預設為 0
Bitmap Data Offset	4	點陣圖資料的起始位址

- ### Bitmap Info Header

長度為 40 Bytes，用來描述圖像本身的屬性。

Name	Size (Byte)	Content
Bitmap Info Header Size	4	Bitmap Info Heade 大小
Width	4	圖像的寬度
Height	4	圖像的高度，正數表示從下到上存儲，負數表示從上到下存儲。
Planes	2	目標設備的色彩平面數，固定為 1。
Bits Per Pixel	2	每像素的位元數。 • 1：單色點陣圖（使用 2 色調色盤） • 4：4 位元點陣圖（使用 16 色調色盤） • 8：8 位元點陣圖（使用 256 色調色盤） • 16：16 位元高彩點陣圖 • 24：24 位元全彩點陣 • 32：32 位元全彩點陣圖
Compression	4	壓縮方式 • 0:BI_RGB，無壓縮 • 1:BI_RLE8，8 位 RLE 壓縮 • 2:BI_RLE4，4 位 RLE 壓縮
Bitmap Data Size	4	圖像數據的大小
H-Resolution	4	圖像的水平分辨率
V-Resolution	4	圖像的垂直分辨率

Used Colors	4	使用的實際顏色數量，設為 0 時表示使用全部顏色。
Important Colors	4	顯示時重要的顏色數量，設為 0 表示所有顏色都重要。

了解完 bmp format 後就可以根據這個格式讀取相對應的資料了。

Flip:

Flip 的作法很簡單，就是把圖片的每一個 row 的資料相反，每個 row 資料都相反的話就是 Flip 的效果。

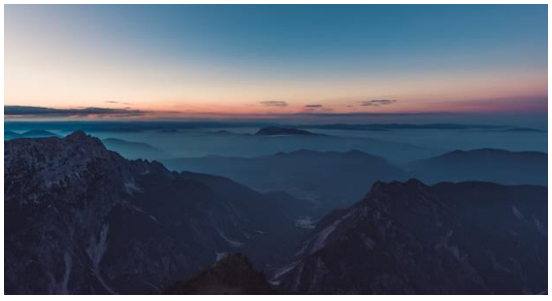
```
for(int r=0;r<(Copy_Img->infoHeader.biHeight);r++){
    for(int c=0;c<(Copy_Img->infoHeader.biWidth/2);c++){
        for(int ch=0;ch<(Copy_Img->infoHeader.biBitCount/8);ch++){
            swap(&Copy_Img->pixelMatrix[ch][r][c],&Copy_Img->pixelMatrix[ch][r][Copy_Img->infoHeader.biWidth-c-1]);
        }
    }
}
```

Result:

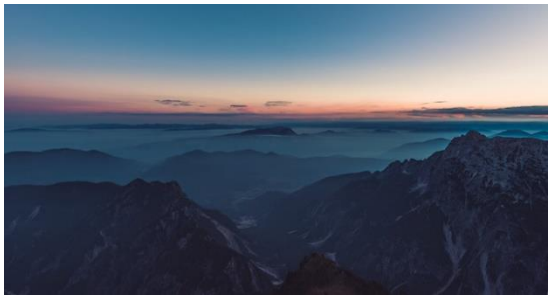
Origin

Flip

Origin



Flip



Resolution:

對兩個不同的 BMP 圖片檔案進行了解析度的量化處理。每個輸入圖像的解析度最初以像素的位數表示（如 3*8bits 或 4*8bits），表示每個像素的顏色位數（如 RGB 或 RGBA）。接著，根據實驗需求，逐步減少每個像素的位數，從而產生不同解析度的輸出圖像。

可觀察出解析度的減少會使圖片的顏色表現能力下降。在 RGB 及 RGBA 圖像中，顏色位元數量越多，圖像顏色越豐富、越細緻；位元數越少，圖像的顏色會變得更加有限，可能產生顏色失真或降低圖像品質。

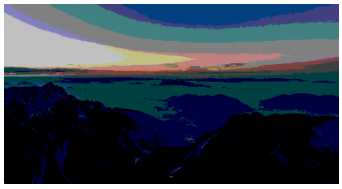
Result:

origin 6 bit 4 bit 2 bit




origin 6 bit 4 bit 2 bit



Resolution_function:

```
int Quantization_func(unsigned char pixel,int quantize_bit){
    // 8 bit -> x bit  =  (p>>(8-x))<<(8-x)
    return ((pixel>>(8-quantize_bit))<<(8-quantize_bit)) ;
}
```

把低位元的資料去掉，只保留高位元的資訊。

Cropping:

原始圖像中選擇一個矩形區域，並將這個選定的區域作為輸出圖像。

輸入起始位置(x、y)和剪裁後的圖片寬和高(w、h)

```
CropBMP(Input_BmpFile1,Crop_Img1,120,150,400,399);
CropBMP(Input_BmpFile2,Crop_Img2,120,150,100,100);
```

開始剪裁之前要先確定剪裁後的大小要比原本圖片大小要小:

```
if (x < 0 || x + w > Img->infoHeader.biWidth || y < 0 || y + h > Img->infoHeader.biHeight) {
    printf("Crop area exceeds the image boundaries!\n");
    freeBMP(Img);
    return;
}
```

從原本的圖片起始位置(x、y)，開始讀取寬和高大小:

```
// read crop img size  from origin img
for (int row = y; row < y+new_height; row++) {
    for (int col = x; col < x+new_width; col++) {
        for(int ch=0;ch<new_channel;ch++){
            Copy_Img->pixelMatrix[ch][row-y][col-x] = Img->pixelMatrix[ch][row][col];
        }
    }
}
```

由於更改了 bmp 檔案的圖片大小，要去更改 bmp format 裡面的資料:

```
// change some parameter by bmp file setting
Copy_Img->infoHeader.biHeight = new_height;
Copy_Img->infoHeader.biWidth = new_width;
Copy_Img->infoHeader.biSizeImage = new_ImageSize;
Copy_Img->fileHeader.bfSize = Copy_Img->fileHeader.bfOffBits + new_ImageSize;  // 更新文件大小
```

Result:

Input1 after Crop

Input2 after Crop