Digital Image Processing HW1

鄧奕辰 311605014

1. Image input/flip/output
   - BMP format
     在 BMP 的圖片資訊中，會包含 54 Bytes 的 Header 資訊，用以表明該圖檔的大小、顏色格式、資訊格式、長寬等資訊。

     ■ 檔頭（File Header）：BMP 檔案的開頭包含一個 14 Bytes 的檔頭，這個檔頭描述此 BMP 檔案的大小與保留位元。
     文件類型（2Bytes）： 用來標識文件類型，一般為 BM。
     檔案大小（4Bytes）： 整個 BMP 檔案的大小，以位元組為單位。
     保留位元（4Bytes）： 保留位元，一般設為 0。
     圖像數據的偏移位（4Bytes）： 圖像數據相對於檔頭的偏移量。

     ■ 影像資訊頭（DIB Header）：接下來的 40 位元組是影像資訊頭，它描述了圖像的詳細資訊。
     頭檔的大小（4Bytes）：通常為 40，表明接下來的資訊為 40Bytes。
     影像寬度（4Bytes）： 圖片的寬度，以像素為單位。
     影像高度（4Bytes）： 圖片的長度，以像素為單位。
     色板數目（2Bytes）： 顏色索引的數目，如果為 0，表示使用所有可能的顏色。
     位元平面數（2Bytes）： 一般為 1。
     每像素位元數（2Bytes）： 表示每個像素使用多少位元，通常為 1、4、8、16、24 或 32，若為 24 則為 RGB，若為 32 則為 RGBA。
     壓縮方式（4Bytes）： 0 表示不壓縮，1 表示使用 RLE-8 壓縮，2 表示使用 RLE-4 壓縮。
     圖像數據的大小（4Bytes）： 圖像數據部分的大小。
     水平和垂直的解析度（4Bytes）： 以像素為單位。
     使用的顏色索引數目（4Bytes）： 重要的顏色索引的數目，如果為 0，表示都重要。
     顏色表部分：如果影像使用調色板，則會包含一個顏色表，其大小為每像素位元數乘以顏色索引數目。

     ■ 影像資料：這部分包含圖片的像素數據。每個像素的 RGB(A)的方式依序排列。

- Flip

Code	說明

`// Read the file header`
`fread(&fileHeader.bfType, sizeof(unsigned short), 1, fpr);`
`fread(&fileHeader.bfSize, sizeof(unsigned int), 1, fpr);`
`fread(&fileHeader.bfReserved1, sizeof(unsigned short), 1, fpr);`
`fread(&fileHeader.bfReserved2, sizeof(unsigned short), 1, fpr);`
`fread(&fileHeader.bfOffBits, sizeof(unsigned int), 1, fpr);`
 	
`// Read the info header`
`fread(&infoHeader.biSize, sizeof(unsigned int), 1, fpr);`
`fread(&infoHeader.biWidth, sizeof(int), 1, fpr);`
`fread(&infoHeader.biHeight, sizeof(int), 1, fpr);`
`fread(&infoHeader.biPlanes, sizeof(unsigned short), 1, fpr);`
`fread(&infoHeader.biBitCount, sizeof(unsigned short), 1, fpr);`
`fread(&infoHeader.biCompression, sizeof(unsigned int), 1, fpr);`
`fread(&infoHeader.biSizeImage, sizeof(unsigned int), 1, fpr);`
`fread(&infoHeader.biXPelsPerMeter, sizeof(int), 1, fpr);`
`fread(&infoHeader.biYPelsPerMeter, sizeof(int), 1, fpr);`
`fread(&infoHeader.biClrUsed, sizeof(unsigned int), 1, fpr);`
`fread(&infoHeader.biClrImportant, sizeof(unsigned int), 1, fpr);`
	將頭檔資訊依序且大小固定的方式存取下來。

`RGBA **img;`
 	
`img = new RGBA*[imgHeight];`
`for(int i = 0; i < imgHeight; i++){`
`    img[i] = new RGBA[imgWidth];`
`}`
	定義圖片格式，且利用動態記憶體的方式將圖片存取下來。

`void flipHorizontal(RGBA** pixelData, int width, int height, int bytesPerPixel) {`
 	
`    for (int x = 0; x < height; ++x) {`
`        for (int y = 0; y < width / 2; ++y) {`
`            swap(pixelData[x][y].b, pixelData[x][width - y - 1].b);`
`            swap(pixelData[x][y].g, pixelData[x][width - y - 1].g);`
`            swap(pixelData[x][y].r, pixelData[x][width - y - 1].r);`
`            swap(pixelData[x][y].a, pixelData[x][height - y - 1].a);`
`        }`
`    }`
`}`
	將圖片與長寬傳到副函式中，將每列中左右交換。

2. Resolution

Code	說明

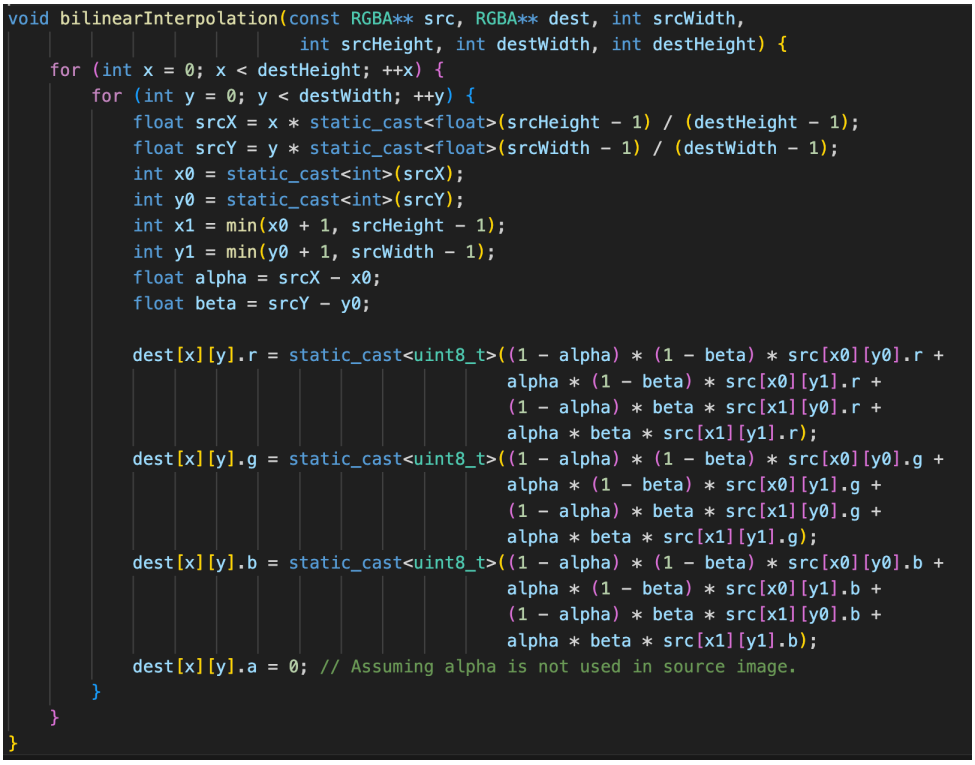
`RGBA **img_6;`
`RGBA **img_4;`
`RGBA **img_2;`
 	
`img_6 = new RGBA*[imgHeight];`
`img_4 = new RGBA*[imgHeight];`
`img_2 = new RGBA*[imgHeight];`
`for(int i = 0; i < imgHeight; i++){`
`  img_6[i] = new RGBA[imgWidth];`
`  img_4[i] = new RGBA[imgWidth];`
`  img_2[i] = new RGBA[imgWidth];`
`}`
	此小題需要輸出 3 張圖片，包含不同解析度的圖片，分別在 8bits 中的 6, 4, 2bits 的解析度。

`for(int i = 0; i <imgHeight; i++){`
`  for(int j = 0; j < imgWidth; j++){`
`  if(bitDepth == 24){`
`  RGB pixel;`
`  fread(&pixel, sizeof(RGB), 1, fpr);`
`  img_6[i][j] = resolution_24(pixel, 0b11111100);`
`  img_4[i][j] = resolution_24(pixel, 0b11110000);`
`  img_2[i][j] = resolution_24(pixel, 0b11000000);`
`  }`
`  if (bitDepth == 32){`
`  RGBA pixel;`
`  fread(&pixel, sizeof(RGBA), 1, fpr);`
`  img_6[i][j] = resolution_32(pixel, 0b11111100);`
`  img_4[i][j] = resolution_32(pixel, 0b11110000);`
`  img_2[i][j] = resolution_32(pixel, 0b11000000);`
`  }`
 	
`  }`
`}`
	將每一張圖片中的畫素，透過 resolution 的 function，利用位元遮罩的方式來降低解析度。

`RGBA resolution_32(RGBA pixel, uint8_t bit){`
`  RGBA resolution;`
`  resolution.b = pixel.b & bit;`
`  resolution.g = pixel.g & bit;`
`  resolution.r = pixel.r & bit;`
`  resolution.a = pixel.a & bit;`
`  return resolution;`
`}`
	利用位元遮罩的方式來降低解析度。例如：11111111&11000000 = 11000000 可得到只有最前面兩個 bits 的資訊。

此降低解析度的方法為，保留 8bits 中的前兩位元，也就是數字最大的兩位元，因此會形成類似於 threshold 的機制，將某個值以下全部都變為 0，因而照成顏色的分佈上較不連續。

3. Scaling

Code	說明
	這段程式碼實現了雙線性插值（Bilinear Interpolation）的功能。雙線性插值是一種用於圖像縮放的技術，通過對原始圖像的像素進行加權平均，生成一個新的目標圖像。函式的具體步驟尤以下所示。

- **bilinearInterpolation** 函式接受了源圖像 **src**，以及一個要生成的目標圖像 **dest** 的指標，還有源圖像和目標圖像的寬高等參數。以下是這個函數的主要步驟：
  1. 使用雙層迴圈遍歷目標圖像的每個像素。
  2. 計算源圖像中對應目標圖像位置的浮點坐標 **srcX** 和 **srcY**。
  3. 將浮點坐標 **srcX** 和 **srcY** 轉換為整數坐標 **x0**, **y0**, **x1**, **y1**，這是因為插值需要使用相鄰的四個像素。
  4. 計算插值的權重係數 **alpha** 和 **beta**，這些係數用於線性插值的計算。
  5. 進行雙線性插值計算，將插值結果存儲到目標圖像的對應位置。
  6. 最後，對於透明度（alpha）通道，這裡假設它在源圖像中未使用，因此將目標圖像的 alpha 設為 0。
- 在寫入輸出檔案時，需注意因為縮放圖片所造成的影響，包含：檔案大小，圖片長度與寬度，因此需在寫入檔案時將以上資訊調整成更改後的數字。