HW1

311511040 曾薔

Image input/flip/output - BMP format

.bmp file 分成幾個不同的部分:

1. BmpFileHeader (14 bytes): 檔案的 header，紀錄檔案的資訊
   - type(2 bytes): 一般都是 0x42 0x4D，為 ASCII 裡的'BM'
   - Size(4 bytes): 是整個檔案的大小，總共有幾個 bytes
   - Reserve1/2(2+2 bytes): 保留值，沒什麼用
   - offset(4 bytes): pixel array 開始的地方

   讀取時要把 type 分開讀取，不然電腦讀取資料時 structure 會一次讀到 4 個 bytes，造成下面 BmpInfoHeader 讀取錯誤。
2. BmpInfoHeader (40 bytes): 圖片資訊的 header
   - bmp info size(4): 圖片 header 的大小
   - Width&Height(4+4): 圖片寬度及高度，為 signed int，負的話表示從另一個方向開始計算
   - Planes(2): 圖片層數，通常是 1
   - Bit Count(2): 每個 pixel 用多少個位元來表示

     24 = 3*8(bits) 為 RGB；32 = 4*8(bits)為 RGB+透明度
   - Compression(4): 圖片的壓縮方式
   - SizeImage(4): 圖片大小，為 Width*Height*Bytes per pixel

- XPelsPerMeter(4): x 方向的解析度
- YPelsPerMeter(4): y 方向的解析度
- ClrUsed(4): 調色盤的顏色數
- ClrImportant(4): 重要顏色的數量，通常不會用到，設為 0

讀取時主要透過 Width, Height 跟 Bit Count 來判斷接下來 Pixel Array 的資料排序方式。

3. Pixel Array

   從左下依序到右上的每個 pixel 的資訊，若 Bit Count 是 24 則一個 pixel 用三個 bytes 來表示；32 則是 4 個 bytes。是一個 height*weight*size_of_pixel 的陣列，做 flip 時將 input array 的每一行做 reverse 再輸出到 output file 當中。

4. Color Table

   BmpInfoHeader 跟 Pixel Array 中間可能會有調色盤，紀錄檔案中數值的真正顏色，如果每個 pixel 只用一個 byte 來紀錄，利用 color map 中的值可以把一個 byte 表示的值轉成所對應的 RGB 三個 bytes 的值。

Resolution

因為要將每個 pixel 中 3 或 4 個原本 8bits 的值降低，故我將 8 個 bits 中較低位數的值去除掉（在這邊是變成 0 來模擬 resolution），因為要讓 resolution 之後跟原本顏色較接近，所以是將小位元的值去掉。

- b = input.b >> (bits_number) << (bits_number);
  // 去掉後面 bits_number 個 bits

經過 resolution 後因為缺乏細微的變化值，總共只有 2^(bits num)種顏色表示法，故得到的結果會在色彩交界的地方有較明顯的界線，這個情況隨著 bits 的降低會越來越明顯，6bits 的時候看起來還跟原圖差不多，但在 4bits 跟 2bits 就可以很明顯看出顏色變化交界處的界線。

Scaling - How Bilinear interpolation works

Linear interpolation 就是假設兩個點之間的數值變化是以線性的方式，以此去估算其中未知點的數值。而當我們想要找的一個點座標坐落在四個點之中，並假設他只會與這四個點以及與這四個點的距離有關係，那可以先沿 x 方向兩組點做 linear interpolation 找出對應 x 值中的兩個數值，再沿 y 方向做一次 interpolation，就可以得到對應座標用四個點作 Bilinear interpolation 的值。

1. 先固定 y0 跟 y1，做 x 方向的 interpolation
   - data^(p(x, y0)) = data(p(x0, y0)) * length(p(x, y0), p(x1, y0)) + data(p(x1, y0)) * length(p(x0, y0), p(x, y0))
   - data^(p(x, y1)) = data(p(x0, y1)) * length(p(x, y1), p(x1, y1)) + data(p(x1, y1)) * length(p(x0, y1), p(x, y1))
2. 再由上面得到的兩個值做 y 方向的 interpolation
   - data^(p(x, y)) = data^(p(x, y0)) * length(p(x, y0), p(x, y)) + data^(p(x, y1)) * length(p(x, y), p(x, y1))