Final Project
影像輸入和輸出

學生

106360231黃思齊

106360208陳　崴

動機

雖然在這學期的高階程式語言課程中學到了許多東西，但我們面對的結果總是……


台北市
記者 vs. 最高軍事檢察長 曹金生
國稅局盯上清玉
完全沒有畫面

概念

開啟圖檔(BMP)

- BMP檔案由四個部分組成:
  1.Bitmap File Header
  2.Bitmap Info Header
  3.Color Table(Palette)
  4.Bitmap Array

讀取檔案資訊

	位置	名稱	大小(Bytes)
File Header	0x000A	Bitmap Data Offset	4
Info Header	0x0012	Width	4
	0x0016	Height	4
	0x001C	Bits Per Pixel	2

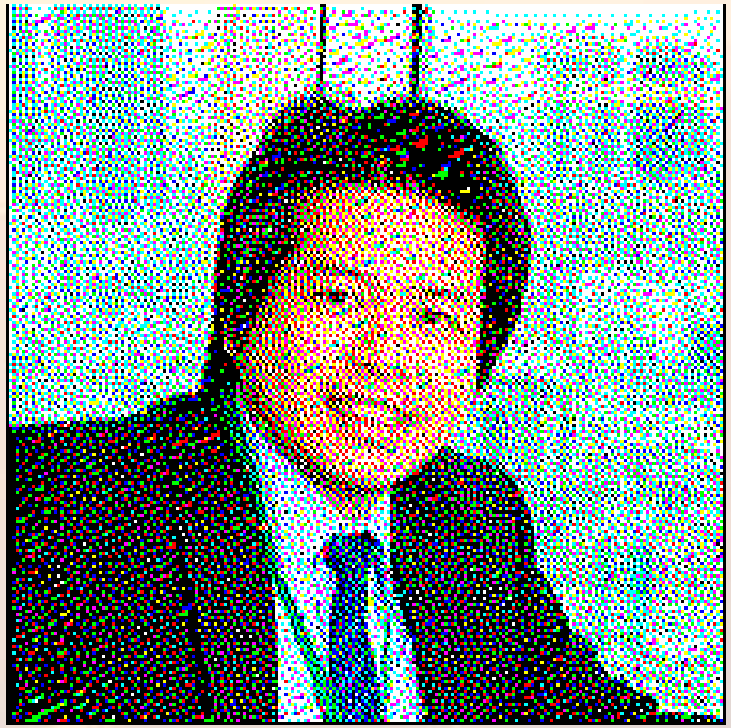
顯示檔案

- #include <windows.h>
- SetPixel(HDC hdc, int x, int y, COLORREF crCOLOR);

灰階 → (R+G+B) /3

顛倒色彩 → 255 - color

失焦 → 和周圍取平均值



抖色處理(Floyd–Steinberg dithering)

程式

標頭檔

- BMPLibrary.h → 讀取和輸出BMP檔
- HSLconvert.h → 將RGB數值轉換成 HSL
- printImage.h → 顯示圖檔和運算各種特效

BMPLibrary.h


```
typedef struct _BMP_Header {
        /*Bitmap File Header*/
        uint16_t Identifier; uint32_t FileSize;
        uint32_t Reserved; uint32_t BitmapDataOffset;
        /*Bitmap Info Header*/
        uint32_t BitmapHeaderSize;
        uint32_t Width; int32_t Height;
        uint16_t Planes; uint16_t BitsPerPixel;
        uint32_t Compression;
        uint32_t BitmapDataSize;
        uint32_t HResolution;
        uint32_t VResolution;
        uint32_t UsedColors;
        uint32_t ImportantColors;
}BMP_Header;
```

BMPLibrary.h

```
void BMPHeaderRead(FILE*, BMP_Header*);

void BMPPrintHeader(BMP_Header*);

color *getBitMap(BMP_Header, FILE*);

void printPixelData(BMP_Header, color*);

void BMPOutput(FILE*, BMP_Header, color*);
```

BMPLibrary.c → void BMPHeaderRead

```
void BMPHeaderRead(FILE *BMPFILE, BMP_Header *Header) {
        fseek(BMPFILE, 0x00, SEEK_SET);
        fread(&Header->Identifier, sizeof(Header->Identifier), 2, BMPFILE);

        fseek(BMPFILE, 0x02, SEEK_SET);
        fread(&Header->FileSize, sizeof(Header->FileSize), 4, BMPFILE);

        fseek(BMPFILE, 0x0A, SEEK_SET);
        fread(&Header->BitmapDataOffset, sizeof(Header->BitmapDataOffset), 4, BMPFILE);

        fseek(BMPFILE, 0x0E, SEEK_SET);
        fread(&Header->BitmapHeaderSize, sizeof(Header->BitmapHeaderSize), 4, BMPFILE);
                                                                .
                                                                .
                                                                .
```

BMPLibrary.c → void BMPPrintHeader

```
void BMPPrintHeader(BMP_Header *Header) {
        printf("\n--------------BMP Header Info----------------\n");
        printf("Identifier \t\t:");
        printf("'%c", Header->Identifier & 0xff);printf("%c'\n", Header->Identifier >> 8);

        printf("FileSize \t\t: ");
        printf("%zu Byte\n", Header->FileSize);

        printf("Bitmap Data Offset\t: ");
        printf("0x%x Byte\n", Header->BitmapDataOffset);
                                                    .
                                                    .
                                                    .
```

BMPLibrary.c → color *getBitmap

void *malloc(size_t size);

規劃一塊記憶體出來給程式使用

void *calloc(size_t num, size_t size) ;

規劃一塊記憶體出來給程式使用，並且初值為0

void *free(void *ptr);

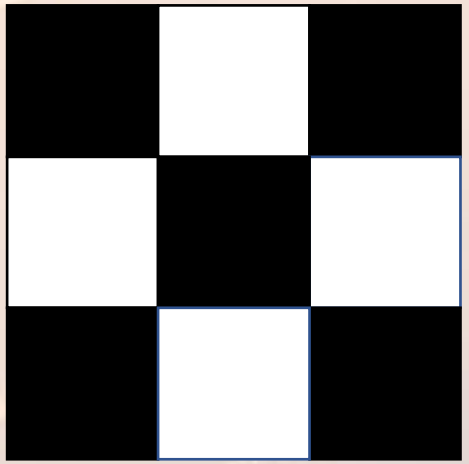
釋放掉malloc 跟 calloc 規劃出來的記憶體

BMPLibrary.c → color *getBitmap

```
color *getBitMap(BMP_Header Header, FILE *BMPFILE) {
        color *BMPColor = calloc(Header.Height*Header.Width, sizeof(color));
        int address = Header.BitmapDataOffset;
        for (int i = 0; i < Header.Height; i++) {
                for (unsigned int j = 0; j < Header.Width; j++) {
                        int index = i*Header.Width + j;
                                讀取RGB的值
                }
                address += Header.Width % 4;
        }
        return BMPColor;
}
```

BMPLibrary.c → void printPixelData

```
void printPixelData(BMP_Header Header, color *BMPColor) {
        for (int i = 0; i < Header.Height; i++) {
                for (unsigned int j = 0; j < Header.Width; j++) {
                        int index = i*Header.Width + j;
                        printf("%02X,", BMPColor[index].R);
                        printf("%02X,", BMPColor[index].G);
                        printf("%02X\t", BMPColor[index].B);
                }
                printf("\n");
        }
}
```



BMPLibrary.c → void BMPOutput

```
void BMPOutput(FILE *Output, BMP_Header Header, color *color) {
/*----------------Header--------------------*/
        fseek(Output, 0x00, SEEK_SET);
        fwrite(&Header.Identifier, sizeof(Header.Identifier), 2, Output);

        fseek(Output, 0x02, SEEK_SET);
        fwrite(&Header.FileSize, sizeof(Header.FileSize), 4, Output);
                .
                .
                .
/*----------------BitMap--------------------*/
        int address = Header.BitmapDataOffset;
```

```
        for (int i = 0; i < Header.Height; i++) {
                for (unsigned int j = 0; j < Header.Width; j++) {
                        int index = i*Header.Width + j;

                        fseek(Output, address++, SEEK_SET);
                        fwrite(&color[index].B, 1, 1, Output);

                        fseek(Output, address++, SEEK_SET);
                        fwrite(&color[index].G, 1, 1, Output);

                        fseek(Output, address++, SEEK_SET);
                        fwrite(&color[index].R, 1, 1, Output);
                }
                address += Header.Width % 4;
        }
        putc(0x00,Output);
}
```

HSLconvert.h

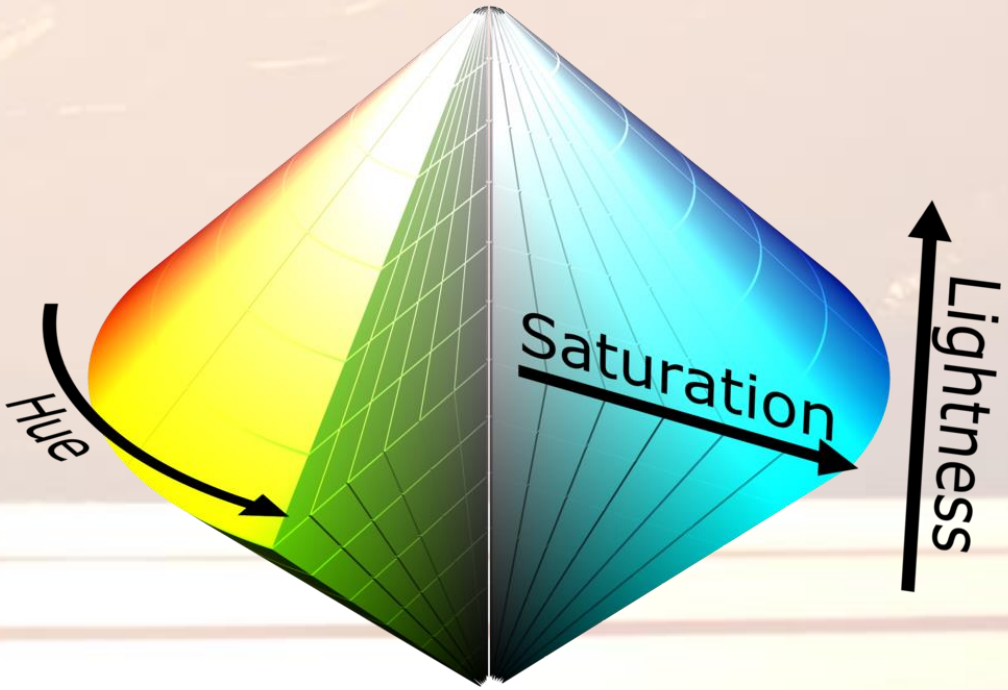
- H (色相)

  色彩的基本屬性，就是平常所說的顏色

  0-360°
- S (濃度)

  是指色彩的純度，越高色彩越純，低則逐漸變灰，取0-100%的數值。
- L (亮度)

  0-100%



HSLconvert.h

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$Cmax = \max(R', G', B')$$

$$Cmin = \min(R', G', B')$$

$$\Delta = Cmax - Cmin$$

$$H = \begin{cases} 0^\circ & ,\Delta = 0 \\ 60^\circ \times \left(\left(G' - B'\right) / \Delta \right) \bmod 6) & , Cmax = R' \\ 60^\circ \times \left(\left(B' - R'\right) / \Delta \right) + 2) & , Cmax = G' \\ 60^\circ \times \left(\left(R' - G'\right) / \Delta \right) + 4) & , Cmax = B' \end{cases}$$

$$L = (Cmax + Cmin) / 2$$

$$S = \begin{cases} 0 & , L = 1 \\ \Delta / (1 - | 2L - 1|) & , \text{otherwise} \end{cases}$$

HSLconvert.h

```
typedef struct _HSLColor {
        int H;
        double S;
        double L;
}HSLColor;

double _RGBmax(double , double , double );
double _RGBmin(double , double , double);

HSLColor *convertRGBtoHSL(BMP_Header, color*);
```

printImage.h

```
void printImagesInConsole(BMP_Header, color*, int, int);
color *Grayscale(BMP_Header, color*);
color   getAverColor(BMP_Header, color*, int , int , int);
color *LostFocus(BMP_Header, color*, int);
color *BinaryImage(BMP_Header, color*);
color *copyImage(BMP_Header, color*);
color *FloydSteinbergDithering(BMP_Header, color*);
color *Reverse(BMP_Header, color*);
```

printImage.h → printImagesInConsole

```
void printImagesInConsole(BMP_Header Header, color *BMPColor, int x, int y) {
		//Get a console handle
		HWND myconsole = GetConsoleWindow();
		//Get a handle to device context
		HDC mydc = GetDC(myconsole);
		COLORREF COLOR;
		for (int i = Header.Height - 1; i >= 0; i--) {
			for (unsigned int j = 0; j < Header.Width; j++) {
				int index = i*Header.Width + j;
				COLOR = RGB(BMPColor[index].R, BMPColor[index].G, BMPColor[index].B);
				SetPixel(mydc, j + x, Header.Height - 1 - i + y, COLOR);
			}
		}
		ReleaseDC(myconsole, mydc);
}
```

Dithering

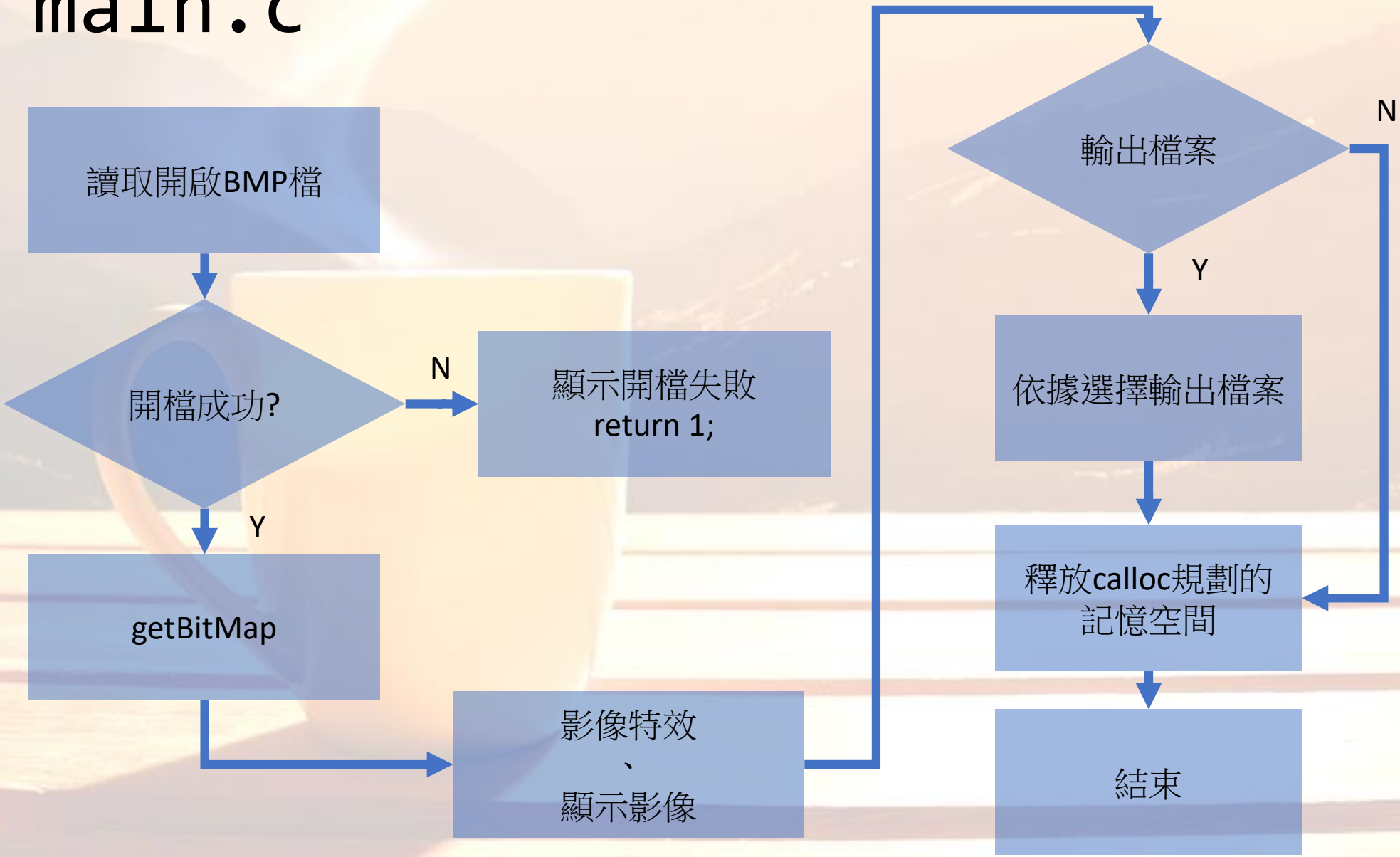
- 顫化。減少顏色種類，讓圖片觀感與原先相仿。每個像素依序重新設定顏色；每個像素的先後顏色誤差，分攤給鄰近的像素。
- dithering 是印刷和液晶顯示的重要技術。報紙上的圖片就用了 dithering ，用少量的單調顏色，調合出原本顏色；在原本像素的周圍點上單調顏色，宏觀望去宛如原本顏色。

Floyd-Steinberg dithering

其中x表示目前的處理點。當該處理點找到最接近色時，便將該點顏色與最接近色顏色的誤差值，以7/16、3/15、5/16、1/16比例擴散到鄰近的點。而鄰近的點在找最接近色前，先依先前擴散過來的顏色誤差值進行調整，最對調整後的顏色值找最接近色，之後再將顏色誤差值擴散出去。重複這個步驟，直到所有影像點處理完為止。Floyd-Steinberg的抖色方法迅速且效果良好，為相當常用的抖色方法。

$$\begin{pmatrix} & & X & 7/16 & \dots \\ \dots & 3/16 & 5/16 & 1/16 & \dots \end{pmatrix}$$

main.c


讀取開啟BMP檔
開檔成功?
N
顯示開檔失敗
return 1;
Y
getBitMap
影像特效
、
顯示影像
輸出檔案
N
Y
依據選擇輸出檔案
釋放calloc規劃的
記憶空間
結束

工作分配

106360231 黃思齊	106360208 陳 崴
簡報製作 規劃程式架構 影像輸入及輸出 影像特效運算	影像特效運算 GitHub檔案管理


END