중간 - 2주차 정리

디지털 영상파일 형식의 종류 및 특징

디지털 영상 파일 형식

[Macromedia Director File Format]

File Import					File Export		Native
Image	Palette	Sound	Video	Anim.	Image	Video	
.BMP, .DIB, .GIF, .JPG, .PICT, .PNG, .PNT, .PSD, TGA, .TIFF, .WMF	.PAL .ACT	.AIFF .AU .MP3 .WAV	.AVI .MOV	.DIR .FLA .FLC .FLI .GIF .PPT	.BMP	.AVI .MOV	.DIR .DXR .EXE

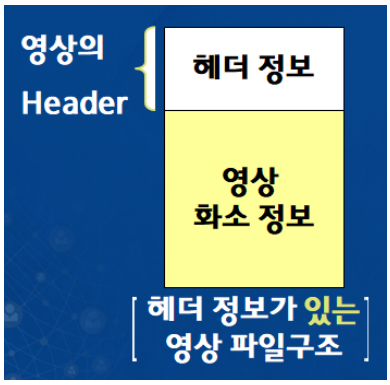
- 가장 많이 상용화 되어 있는 형식 : **JPEG** (압축 O)

→ JPEG은 영상 압축에서 가장 기본이 되는 압축 표준 (Lossy 압축, 손실 O)

- 수업 시간에 가장 많이 사용하게 될 형식 : **BMP** (압축 X)

→ BMP 파일은 어떤 기법도 사용하지 않고 픽셀값을 그대로 저장 (파일 용량 큼)

영상 파일의 구조



영상 처리 시, Header 검사는 필수

영상의 Header 정보 ⇒ 1. 영상의 가로/세로 길이 2. 영상의 칼라/흑백 여부

컬러 영상 표현 방법

1. `트루컬러영상` (세상의 모든 컬러를 표현 가능) = `24비트 컬러 영상`

: RGB를 각 8bits (총 24bits) 씩 사용하여 한 픽셀을 표현 할 수 있음 ⇒ **2^24 개의 컬러**를 사용

⇒ 640 x 480 영상을 트루 컬러로 표현하면 사이즈가 921KB 가 나온다.

⇒ 매우 많은 용량을 차지 (그레이 컬러 영상의 약 3배)

따라서 트루컬러를 사용하기 보다 인덱스 컬러를 사용 하게 된다.

2. `인덱스컬러영상` = `8비트 컬러 영상`

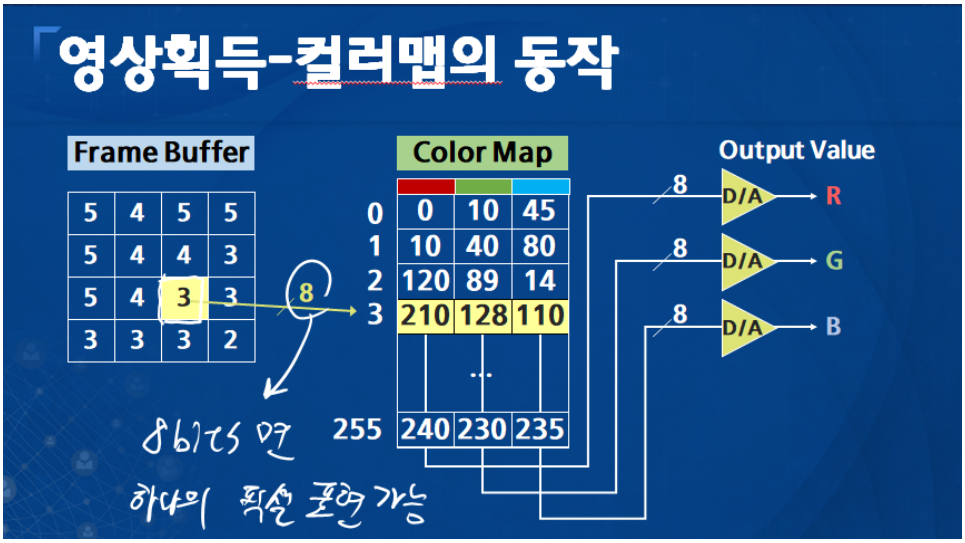
: **256 개의 컬러**만 뽑아서 사용

⇒ `컬러맵 (팔레트)` 를 사용, 24비트 컬러 영상과 시각적 차이가 미미

- `컬러맵 (팔레트,Look Up Table)`

: 집이 아무리 부자여서 많은 물감을 가지고 있다 해도 팔레트는 정해진 물감의 개수만큼 사용

BMP 파일에서 제공하는 인덱스 컬러 모델은 총 256칸



디지털 영상 파일 형식

BMP 파일 : `헤더` + `영상의 화소 정보`

1. 헤더 : **파일 헤더(FILE)** 14byte + **영상(INFO) 헤더** 40byte + **팔레트(RGBQUAD) 헤더** 1024byte
2. 영상의 화소 정보 : BMP 에는 영상이 뒤집혀서 저장 (y 좌표가 거꾸로)

`Windows 라이브러리에 존재`

파일 FILE 헤더 14바이트

- WORD 2바이트, DWORD 4바이트
- BMP 파일에는 bf대신 bm을 사용 (bmp파일인지 확인)

```
typedef struct tagBITMAPFILEHEADER
{
    WORD   bfType;           // 파일형식
    DWORD  bfSize;           // 이미지 파일의 사이즈
    WORD    bfReserved;  // 예약된 변수, 사용안함
    WORD    bfReserved;  // 예약된 변수, 사용안함
    DWORD  bfoffBits;     // 영상데이터 위치까지의 거리
                                    // (영상데이터의 시작포인터)
} BITMAPFILEHEADER;
```

영상 INFO 헤더 40바이트

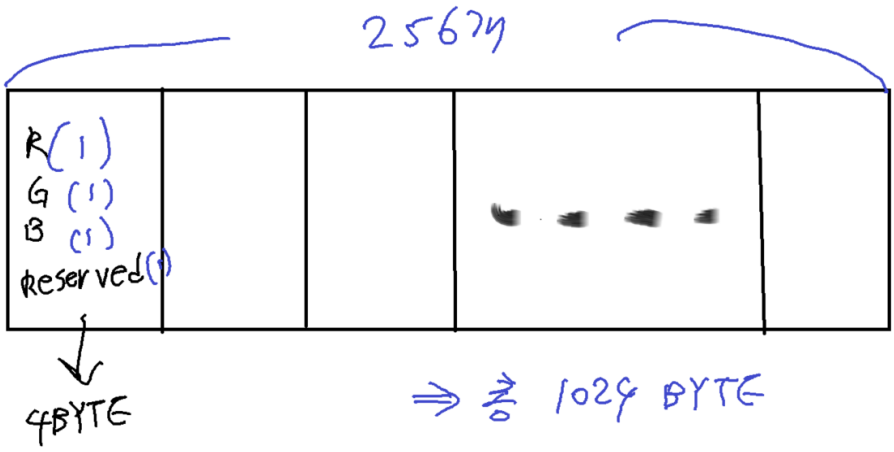
- biSize : 구조체의 크기 40 바이트
- biWidth, biHeight : 영상의 가로, 세로 길이
- biBitCount : 영상의 컬러와 흑백을 구분(컬러-24, 흑백-8)

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD    biSize;               // 이 구조체의 크기, 40바이트가 저장되어 있음
    LONG        biWidth;           // 영상의  가로 크기
    LONG        biHeight;          // 영상의 세로 크기
    WORD       biplanes;          // 비트플레인 수(항상 1)
    WORD       biBitCount;       // 화소당 비트수(컬러=24, 흑백구별=8)
    DWORD    biCompression;     // 압축유무
    DWORD    biSizeImage;          // 영상의 크기(바이트 단위)
    LONG        biXPelsPerMeter;  // 가로 해상도
    LONG        biYPelsPerMeter;  // 세로 해상도
    DWORD    biClrUsed;               // 실제 사용 색상수
    DWORD    biClrImportant;      // 중요한 색상인덱스
} BITMAPINFOHEADER;
```

팔레트 (RGB QUAD) 헤더 1024 바이트

- BYTE는 Unsigned Character Type 이기 때문에 1바이트
- 인덱스 컬러 팔레트 구성 시, 256개의 배열을 선언

: 4바이트 ⇒ 256개의 물감을 짜야하기 때문에 4 x 256 = 1024 바이트




```
typedef struct tagRGBQUAD // 총 4바이트
{
    BYTE   rgbBlue;              // B 성분(파란색)
    BYTE   rgbGreen;           // G 성분(녹색)
    BYTE   rgbRed;               // R 성분(빨간색)
    BYTE   rgbReserved1;   // 예약된 변수, 사용하지 않음
} RGBQUAD;
```

BMP 파일 입출력 실습

영상 처리 코드 구성 (3단계)

1. 영상 입력 (BMP 파일을 메모리에 올림)
2. 영상 처리 (메모리에 올라온 영상 정보를 프로그래밍 적으로 처리)
3. 처리된 결과 : 다시 BMP 파일로 출력

```
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h> // 파일 입출력
#include <Windows.h> // File, Info, 팔레트에 대한 구조체가 존재
void main()
{
  // 1. 영상 입력 단계
  BITMAPFILEHEADER hf; // BMP 파일헤더 14Bytes
  BITMAPINFOHEADER hInfo; // BMP 인포헤더 40Bytes
  RGBQUAD hRGB[256]; // 팔레트 (256 * 4Bytes) 1024bytes

  FILE *fp; // 파일 읽어오기
  fp = fopen("lenna.bmp", "rb"); // rb:read binary(읽기 모드), 텍스트 경우("rt")
  if(fp == NULL) return; // 예외 처리
  fread(&hf, sizeof(BITMAPFILEHEADER), 1, fp); //fp 가 가리키는 위치로 부터 14byte 읽어라
  fread(&hInfo, sizeof(BITMAPINFOHEADER), 1, fp); //fp는 읽어들인 위치부터 40byte만큼 읽어라
  fread(hRGB, sizeof(RGBQUAD), 256, fp); // 4byte씩 256번 읽어서 hRGB 배열에 담아라
  int ImgSize = hInfo.biWidth * hInfo.biHeight; // 영상 사이즈 = 가로 * 세로
  BYTE * Image = (BYTE *)malloc(ImgSize); // 영상 원본을 담을 배열 선언
```

```
  // malloc 함수 : 동적할당 한 메모리의 첫번째 주소를 반환한다.
  BYTE * Output = (BYTE *)malloc(ImgSize); // 영상의 처리결과를 담을 배열
  fread(Image, sizeof(BYTE), ImgSize, fp); // Image 동적할당에 Image를 담는다.
  fclose(fp); // fp의 관계를 끊어버리기

 // 2. 영상처리
  //영상반전
  for(int i=0; i<ImgSize; i++)
    Output[i] = 255 - Image[i];// 반전 (블랙->화이트, 화이트->블랙)

 // 3. 처리된 결과 출력
 // 헤더 정보는 그대로, 이미지 정보만 Output 배열로 넣어준다.
  fp = fopen("output.bmp", "wb"); // 출력하기 위한 파일을 생성, 쓰기모드("wb")
  fwrite(&hf, sizeof(BYTE), sizeof(BITMAPFILEHEADER), fp);
  fwrite(&hInfo, sizeof(BYTE), sizeof(BITMAPINFOHEADER), fp);
  fwrite(hRGB, sizeof(RGBQUAD), 256, fp);
  fwrite(Output, sizeof(BYTE), ImgSize, fp);
  fclose(fp); // 포인터 관계 끊기
  free(Image); // 동적할당 메모리 해제, 메모리 누수현상 방지
  free(Output);
}
```


BMP 파일은 이미지의 위아래가 뒤집혀 있는데 코드에 이런 것의 고려가 필요한가?

- 모든 화소에 동일한 연산을 진행하는 것은 상관 없다.
- 하지만, 좌표 값을 검출하는 것은 y값이 뒤집혀 있기 때문에, 영상의 세로 사이즈에서 계산한 y값을 빼주어서 좌표 값을 역으로 계산을 진행한다.