Commentary (해설지)

1. 비트맵 & 비트맵 파일

BMP 는 비트맵(Bitmap)의 줄임말로, 어떤 디바이스에서도 정확하게 이미지를 보여줄 수 있는 파일 유형이다. 각각의 이미지 픽셀을 고유한 개체로 처리하는 특징 때문에 이미지 품질이 좋고 세부 정보도 많다. 따라서 파일 크기가 크지만 그만큼 편집하기도 쉽다.

비트맵 파일은 바이너리 형식이므로 메모장 등 텍스트 편집기로 열어도 내용을 알아볼 수 없다. 따라서 비트맵 파일에서 픽셀 정보를 읽으려면 먼저 비트맵 파일의 구조를 알아야 한다.

비트맵 파일은 픽셀 하나를 몇 비트로 저장하느냐에 따라 구조가 달라진다. 가장 널리 사용되는 비트맵은 24 비트 비트맵이다. 24 비트 비트맵은 한 픽셀을 나타내는데 24 비트가 쓰인다.

비트맵 파일 헤더는 비트맵 파일 식별 정보, 파일 크기, 데이터 위치 등의 정보를 담고 있으며 DIB(Device Independent Bitmap) 헤더는 가로, 세로 크기, 해상도, 픽셀의 비트 수 등 그림의 자세한 정보를 담고 있다.

픽셀 데이터에는 그림 파일의 실제 색상 정보가 들어간다. 단, 픽셀당 색을 표현하는데 필요한 비트가 8 비트 미만일 때는 색상 테이블에 따로 색상 정보를 필수로 저장하고, 24 비트 비트맵 파일을 사용하는 경우, 색상 테이블은 처리하지 않는다.

2. 비트맵 구조

비트맵 파일 헤더(BITMAPFILEHEADER)의 구조

멤버	크기(바이트)	설명
bfType	2	BMP 파일 매직 넘버 비트맵 파일이 맞는지 확인하는데 사용 비트맵 파일 매직 넘버 : 0x424D
bfSize	4	파일의 크기(바이트)
bfReserved1	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfReserved2	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfOffBits	4	비트맵 데이터의 시작 위치

비트맵 정보 헤더(BITMAPINFOHEADER)의 구조

멤버	크기(바이트)	설명
biSize	4	현재 비트맵 정보 헤더(BITMAPINFOHEADER)의 크기
biWidth	4	비트맵 이미지의 가로 크기(픽셀)
beHeight	4	비트맵 이미지의 세로 크기(픽셀) 양수: 이미지의 상하가 뒤집혀서 저장된 상태 음수: 이미지가 그대로 저장된 상태 보통 세로 크기는 양수로 저장
biPlanes	2	사용하는 색상판의 수. 항상 1
biBitCount	2	픽셀 하나를 표현하는 비트 수
biCompression	4	압축 방식. 보통 비트맵은 압축을 하지 않으므로 0
biSizeImage	4	비트맵 이미지의 픽셀 데이터 크기(압축 되지 않은 크기)
beXPelsPerMeter	4	그림 가로의 해상도(미터 당 픽셀)
biYPelsPerMeter	4	그림 세로의 해상도(미터 당 픽셀)
biClrUsed	4	색상 테이블에서 실제 사용되는 색상 수
biClrImportant	4	비트맵을 표현하기 위해 필요한 색상 인덱스 수

24 비트 비트맵은 픽셀(RGBTRIPLE)을 파랑(B), 초록(G), 빨강(R) 순서로 저장하며 각 색상의 크기는 1 바이트이다. 따라서 픽셀당 3 바이트를 사용한다. 즉, 우리가 화면에서 보는 24 비트 비트맵 파일의 한 픽셀은 3 바이트로 되어 있다.

```
typedef struct s_file_header
{
	unsigned short		bf_type;
	unsigned int		bf_size;
	unsigned short		bf_reserved1;
	unsigned short		bf_reserved2;
	unsigned int		bf_off_bits;
}			t_file_header;

typedef struct s_info_header
{
	unsigned int		bi_size;
	int			bi_width;
	int			bi_height;
	unsigned short		bi_planes;
	unsigned short		bi_bit_count;
	unsigned int		bi_compression;
	unsigned int		bi_size_image;
	int			bi_x_pels_per_meter;
	int			bi_y_pels_per_meter;
	unsigned int		bi_clr_used;
	unsigned int		bi_clr_important;
}			t_info_header;
```

컴파일러 옵션 중에 -fpack-struct[=n]을 통해 패딩비트를 사용하지 않을 수 있다. n 의 값에 따라 구조체 크기를 정하여 정렬시킬 수 있다.

따라서 pragma 전처리문을 사용하지 않고도 norminette 기준을 만족시킬 수 있다.

3. 비트맵 읽기 & 쓰기

읽기

BMP 파일을 fopen()을 통해 이진 형식 읽기 전용으로 연다. fread()를 통해 읽은 내용을 미리 선언해 둔 비트맵 파일 헤더와 비트맵 정보 헤더 구조체에 저장한다. fseek()를 통해 파일의 읽기·쓰기 위치를 파일의 처음 위치로 초기화, 이미지 데이터가 시작되는 지점을 가리키는 bfOffBits 만큼 포인터의 위치를 이동한다. 이미지 크기만큼 동적 할당한 unsigned char * 타입의 image 변수에 다시 fread()를 통해 이미지 크기만큼 파일에서 읽어온다.

쓰기

fopen()을 통해 이진 형식 쓰기 전용으로 파일을 생성해서 열고 임시 버퍼에 수정 전의 기존 BMP 파일을 읽어서 내용을 쓰고, 수정된 데이터를 가지고 있는 image 변수의 내용들도 fwrite()을 통해 쓴다.

메모리 동적할당했던 부분들은 사용 후 반드시 메모리 해제해주고, open 했던 파일들도 모두 close 해주어야한다.

4. 색상 바꾸기

비트맵은 원래 1 픽셀에 3 가지 색상 정보가 BGR 순으로 저장되어 있다. 그 순서를 바꿔서 모든 픽셀의 색상 정보를 원래의 역순인 RGB 로 수정하면 색상이 좌우반전되어 표현된다. RGB 순으로 저장되어 있는 BMP 파일을 제공하면 프로젝트 진행자는 BGR 순으로 색상을 돌려 원래 이미지의 색상을 복구해야한다.

5. 상하/좌우 반전

이미지의 정보가 담겨있는 image 변수를 하나 복사해두고 좌우반전은 image 를 세로로 반절 나눠 데이터들을 swap 하고 상하반전은 가로로 반절 나눠 데이터들을 swap 해주면 이미지가 좌우/상하로 뒤집힌다. 대신 반전시킬 때 색상 정보는 BGR 순으로 담겨있는 그대로 저장해야 색상이 변하지 않는다.

6. 확대

원본과 똑같은 이미지 크기에 정수 n 배만큼 확대하여 저장한다. 원본의 한 픽셀을 n 의 제곱만큼 복사한다. 이때 기준점에 따라 이미지에 저장되지 못한 픽셀들은 버려지게 된다. 비트맵의 특성상 데이터가 반대로 저장되어 있을 수 있는데, 알맞은 처리를 하여 올바른 이미지를 만들 수 있다.

Reference

- https://dojang.io/mod/page/view.php?id=702
- https://en.wikipedia.org/wiki/BMP_file_format