Encoding & Decoding

Coding Technology Document

- FFMPEG를 활용한 Visual Studio 내 Video Encoding & Decoding –

이타기술

목차

1. 개요
2. Header Include
3. Library 초기화
4. Media 파일 열기
5. Stream Demuxing
6. Decoding
7. Encoding
8. 결과
9. 시스템 환경
10. 부록
11. 개요

본 문서에서는 FFMPEG의 라이브러리를 이용해 Window 환경에서 Media file의 Decoding후, 다시 Encoding을 하는 Re\_Encoding 과정을 다룬다. 과정은 소스코드를 중심으로 설명하며, 자세한 정보와 오류보고는 하단의 부록에 포함된다.

1. Header Include

* FFMPEG를 사용하기 위해, 필요한 헤더 파일들을 포함한다.
* #include <libavformat/avformat.h> // Basic Format manage header
* #include <libavcodec/avcodec.h> // For using codec header
* #include <libavutil/avutil.h> // AV Convenience header
* #include <libavutil\imgutils.h> // Img Convenience header
* #include <libswscale/swscale.h> // External API header
* #include <libavformat/avio.h> // Input & output header
* #include <libavutil/opt.h> // option header
* #include <libavutil/mathematics.h> // math header
* #include <libswresample/swresample.h> // Resample header

1. Library 초기화 하기

* FFMPEG을 사용하기 전에, 지원하는 모든 Component를 사용 할 수 있게 먼저 등록한다.
* av\_register\_all();

1. Media 파일 열기

* Media 파일을 다루기 위해 먼저, 파일을 Open 한다.
* avformat\_open\_input (AVFormatContext \*\*ps, const char \*filename, AVInputFormat \*fmt, AVDictionary \*\*options);
* AVFormatContext 파라미터는 해당 Media의 정보를 담은 구조체이다.
* Filename 파라미터는 Open할 파일의 이름이다.
* fmt 파라미터는 강제로 InputFormat을 지정할 경우 사용한다. (보통 NULL)
* Options 파라미터는 Demuxer의 추가 옵션을 설정할 때 사용한다. (보통 NULL)

1. Stream Demuxing

대부분의 Media는 Video Stream 하나와 Audio Stream 하나로 구성 되어있으며, 보통 Video는 0번 Stream\_Index를 가지고, Audio는 1번 Stream\_Index를 가진다.

하지만, 복수의 Stream을 가지는 Media도 있을 수 있기에, 다음의 함수를 사용한다.

nVSI = av\_find\_best\_stream(f\_ctx, AVMEDIA\_TYPE\_VIDEO, -1, -1, NULL, 0); 사용한다.

위 함수는 현재 FormatContext에서 다음 Stream의 Index를 반환하는 함수이다.

찾은 Stream Index를 이용해, 다음의 변수들을 초기화 한다.

AVCodecContext \*pVCtx = f\_ctx->streams[nVSI]->codec; // 비디오 코덱의 Context

AVCodecContext \*pACtx = f\_ctx->streams[nASI]->codec; // 오디오 코덱의 Context

AVFrame \*pVFrame = avcodec\_alloc\_frame(); // 비디오 Frame (Allocation)

AVFrame \*pAFrame = avcodec\_alloc\_frame(); // 오디오 Frame (Allocation)

int bGotPicture , bGotSound = 0; // Data 획득 구분

Media file의 Video Stream 과 Audio Stream을 구분해서 사용할수 있는 것을 확인 할 수 있다.

1. Decoding

Decoding 하기에 앞서 이에 사용 할 Decoder를 설정한다.

* 1. CODEC\_ID에 맞는 디코더를 찾기 위해, avcodec\_find\_decoder 함수를 사용한다.

이를 통해 Codec을 초기화 한다.

AVCodec \*pVideoCodec = avcodec\_find\_decoder(f\_ctx->streams[nVSI]->codec->codec\_id);

5.2 avcodec\_open2를 사용해 CodecContext를 Codec으로 초기화한다.

if (avcodec\_open2(f\_ctx->streams[nVSI]->codec, pVideoCodec, NULL) < 0)

5.3 Decoder 설정이 완료 되면, Decoding을 한다.

av\_read\_frame 함수를 이용해 FormatContext 와 Packet을 전달, frame을 읽어온다.

이후 Packet의 Stream을 구분하여 Video, Audio를 나눈다.

각각의 Packet에 Decode 함수인 avcodec\_decode\_video2를 사용한다.

다음은 선언과 초기화를 제외한 Decoding 과정의 코드 예시이다.

while (av\_read\_frame(f\_ctx, &packet) >= 0) {

if (packet.stream\_index == nVSI) {

// Decode Video

avcodec\_decode\_video2(dec\_ctx, pVFrame, &bGotPicture, &packet);

if (bGotPicture) {

// Encode or Rendering

}

}

}

1. Encoding
   1. Encoding을 하기에 앞서 필요한 변수들을 초기화 해준다.

AVCodec \*codec; // 인코딩용 코덱

AVCodecContext\* c = NULL; // 코덱의 Context

* 1. Decoding과 마찬가지로

Codec에 원하는 Encoder를 찾기 위해, avcodec\_find\_encoder 함수를 사용한다.

codec = avcodec\_find\_encoder(Codec\_id);

* 1. CodecContext를 Codec에 맞게 Allocation한다.

c = avcodec\_alloc\_context3(codec);

* 1. CodecContext c 에 원하는 인코딩 정보를 입력한다.

예시)

c->bit\_rate = 500000;

c->width = 1280;

c->height = 528;

c->time\_base = { 1,25 };

c->gop\_size = 10; /\* emit one intra frame every ten frames \*/

c->max\_b\_frames = 1;

c->pix\_fmt = AV\_PIX\_FMT\_YUV420P;

* 1. Packet을 사용하기 전 초기화 해준다.

av\_init\_packet(&packet);

* 1. Encoding 함수인 avcodec\_encode\_video2를 사용한다.

Source Data인 pVFrame을 입력해, 인코딩 후 packet으로 반환한다.

avcodec\_encode\_video2(c, &packet, pVFrame, &got\_output);

* 1. Encoding 된 Packet을 본 문서에서는 Media File로 저장한다.

fwrite를 이용해 file로 Encoded Packet을 저장한다.

if (got\_output) {

printf("Write frame %3d (size=%5d)\n", i, packet.size);

fwrite(packet.data, 1, packet.size, f);

if (!ret && got\_output && packet.size) {

packet.stream\_index = 0;

/\* Write the compressed frame to the media file. \*/

//ret = av\_interleaved\_write\_frame(oc, &packet);

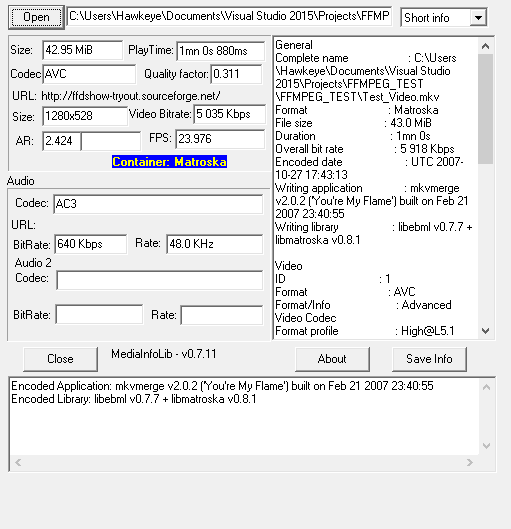
}

av\_free\_packet(&packet);

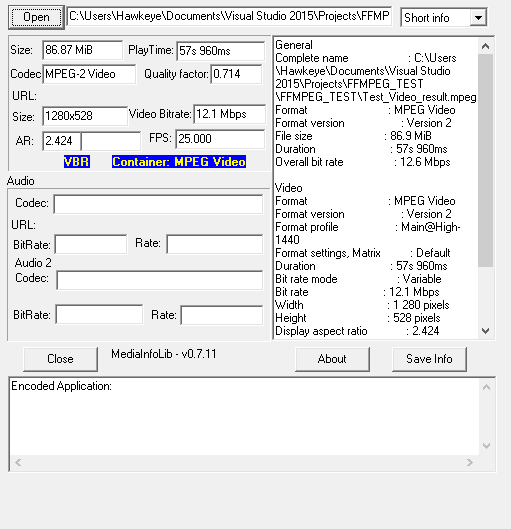
}

1. 결과

변경 된 Media File의 정보를 확인 할 수 있다.



<인코딩 전 저장된 Video의 Info>

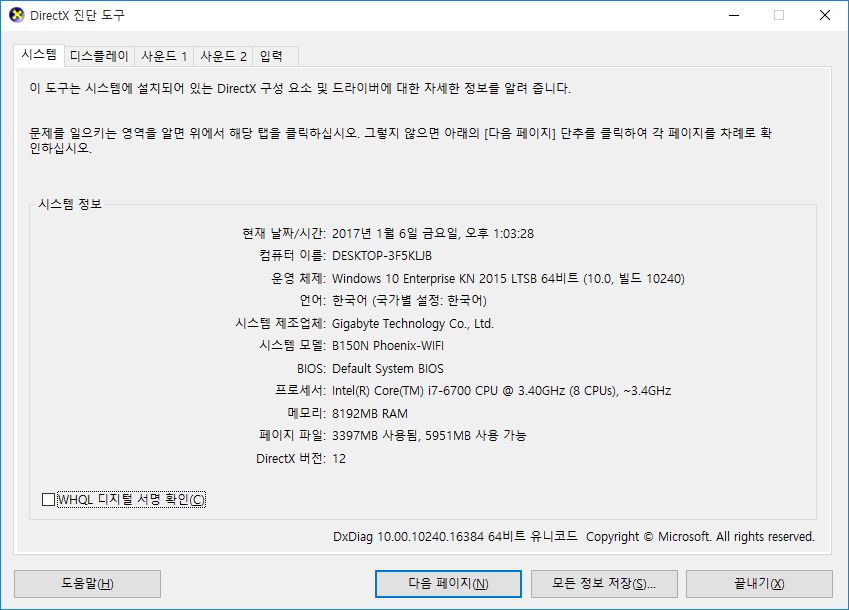


<인코딩 후 저장된 Video의 Info>

1. 시스템 환경

Microsoft Visual Studio 2015 (v140)

FFMPEG 2.7.1



1. 참고문헌

FFMPEG 활용 코딩시에 참고했던 자료의 링크를 첨부한다.

1. <http://aslike.egloos.com/3081447>

기본적인 FFMPEG 코딩의 기초를 확인 할 수 있다.

1. <https://www.ffmpeg.org/doxygen/2.1/index.html>

FFMPEG 라이브러리의 Document를 확인 할 수 있다.

1. <http://dg087.tistory.com/61>

MinGW를 이용한, 윈도우용 라이브러리 제작을 확인 할수 있다.

1. <http://ffmpeg.org/doxygen/trunk/decoding__encoding_8c-source.html>

Decoding & Encoding의 예제 소스를 참고 할 수 있다.

본 Document의 Source code는

Github에서 다운로드 할수 있다.

https://github.com/Phigaro/FFMPEG