千 Line으로 비디오플레이어 만들기 7

멀티미디어 : 2009/03/09 14:03

ffmpeg 천줄로 비디오플레이어 만들기
http://www.dranger.com/ffmpeg/ffmpeg.html
http://hybridego.net/entry/천줄로-비디오플레이어-만들기
http://hybridego.net/entry/千-Line-으로-비디오플레이어-만들기-1
http://hybridego.net/entry/千-Line-으로-비디오플레이어-만들기-2
http://hybridego.net/entry/千-Line-으로-비디오플레이어-만들기-3
http://hybridego.net/entry/千-Line-으로-비디오플레이어-만들기-4
http://hybridego.net/entry/千-Line-으로-비디오플레이어-만들기-5
http://hybridego.net/entry/千-Line으로-비디오플레이어-만들기-6
http://hybridego.net/entry/千-Line으로-비디오플레이어-만들기-7
http://hybridego.net/entry/千-Line으로-비디오플레이어-만들기-8
http://hybridego.net/entry/千-Line으로-비디오플레이어-만들기-마무리
별로 중요하지 않은 부분중 생략된 부분이 있고 의역이 많이 있습니다.
군데군데 틀린 번역이 있을수 있습니다. (영어가 후달려서)

하지만 소스코드 설명 부분은 틀리지 않도록 노력했습니다.

위의 원문을 번역한 것입니다.
잘못된 부분이 있을지도 모르겠습니다.
혹시 잘못된 부분을 발견하시면 댓글로 알려주시면 감사하겠습니다.

Tutorial 07: Seeking

tutorial07.c

Handling the seek command
이제 우리는 플레이어에다가 찾기 능력을 추가해보도록 하고 이것을 만들면서 av_seek_frame 함수를 사용하는 법을 익혀보겠습니다.

우리는 좌우 화살표는 영상의 앞뒤로 조금씩 움직이고 상하 화살표는 만ㅍ이 움직이게 하도록 만들겠습니다. (여기서 조금은 10초 그리고 많이는 60초 입니다.) 그래서 우리의 main loop에서 키보드입력을 감지할수 있게 설정해야 합니다. 하지만 우리가 키 입력을 받았을 떄, 바로 `av_seek_frame` 함수를 호출할수 없습니다. 그래서 이것은 main decode loop(decode_thread)에서 처리해야 합니다. 그래서 그대신에 우리 큰 구조체에다가 몇가지 값을 추가하겠습니다. 우리가 찾을 위치와 찾는중인지를 표시하는 플래그 값을요.

view plain copy to clipboard print ?

```
01.  int             seek_req;
02.  int             seek_flags;
03.  int64_t         seek_pos;
```

이제 우리 메인 루프에다가 키입력을 감지할 수 있도록 설정합니다.

```
view plain  copy to clipboard  print  ?
 for(;;) {
   double incr, pos;

   SDL_WaitEvent(&event);
   switch(event.type) {
   case SDL_KEYDOWN:
     switch(event.key.keysym.sym) {
     case SDLK_LEFT:
incr = -10.0;
goto do_seek;
     case SDLK_RIGHT:
incr = 10.0;
goto do_seek;
     case SDLK_UP:
incr = 60.0;
goto do_seek;
     case SDLK_DOWN:
incr = -60.0;
goto do_seek;
     do_seek:
if(global_video_state) {
  pos = get_master_clock(global_video_state);
  pos += incr;
  stream_seek(global_video_state,
                    (int64_t)(pos * AV_TIME_BASE), incr);
}
break;
     default:
break;
     }
     break;
```

SDL_WaitEvent()
키입력을 감지하기 위해서는 우선 SDL_KEYDOWN 이벤트를 받아야 합니다. 그러면 event.key.keysym.sym 을 체크해서 어떤키가 눌려졌는지 찾아냅니다. 찾는 법을 알았으니까 이제 새로운 get_master_clock 함수의 값을 증가시킨 새로운 time을 계산해야 합니다. 그리고 seek_pos 등을 설정하기 위한 stream_seek 함수를 호출합니다. 그다음엔 우리가 얻은 새로운 time을 avcodec의 내부 timestamp 로 변환합니다. 스트림에서 timestamp 는 second 보다는 frame으로 측정되는데 공식은 seconds=frames*time_base(fps) 입니다. avcodec은 기본으로 한 값이 1,000,000fps 입니다. (2초의 1 pos는 2,000,000 timestamp 입니다.)

다음은 stream_seek 함수입니다. 만약 반대로 재생하고 있다면 플래그를 설정해야 합니다.

```
view plain  copy to clipboard  print  ?
void stream_seek(VideoState *is, int64_t pos, int rel) {

  if(!is->seek_req) {
    is->seek_pos = pos;
    is->seek_flags = rel < 0 ? AVSEEK_FLAG_BACKWARD : 0;
    is->seek_req = 1;
  }
}
```

이제 decode_thread 를 지나서 실제적으로 탐색을 수행하는 부분을 보겠습니다. 소스파일 안에 "seek stuff goes here"라고 마킹을 해두었습니다. 우리는 이것을 거기에 넣습니다.

탐색의 중심부분에는 av_seek_frame 함수가 있습니다. 이 함수는 format context, stream, timestamp, 아규먼트 같

은 플래그 세트들을 얻을수 있습니다. 이 함수는 여러분이 지정한 timestamp를 찾을것입니다. timestamp의 유닛은 함수에 보내는 stream의 time_base 입니다. 하지만 여러분은 이것에 스트림을 보내서는 안됩니다.(indicated by passing a value of -1). 만약 여러분이 그렇게 하면 time_base는 avcodec의 내부 timestamp 유닛이 되거나 1000000fps가 될것입니다. 이것이 seek_pos를 설정할때 AV_TIME_BASE당 포지션을 곱했던 이유입니다.

만일 여러분이 av_seek_frame -1 을 스트림에 보내면 몇몇 파일에서 문제에 부딪힐수 있습니다. 그래서 우리는 파일에서 첫번째 스트림을 끄집어 내서 이것을 ac_seek_frame에 보낼것입니다. 우리가 timestamp를 새로운 유닛으로 rescale 해야한다는것을 잊으면 안됩니다.

view plain copy to clipboard print ?

```
01.  if(is->seek_req) {
02.    int stream_index= -1;
03.    int64_t seek_target = is->seek_pos;
04.
05.    if     (is->videoStream >= 0) stream_index = is->videoStream;
06.    else if(is->audioStream >= 0) stream_index = is->audioStream;
07.
08.    if(stream_index>=0){
09.      seek_target= av_rescale_q(seek_target, AV_TIME_BASE_Q,
10.                        pFormatCtx->streams[stream_index]->time_base);
11.    }
12.    if(av_seek_frame(is->pFormatCtx, stream_index,
13.                     seek_target, is->seek_flags) < 0) {
14.      fprintf(stderr, "%s: error while seeking\n",
15.              is->pFormatCtx->filename);
16.    } else {
17.
```

av_seek_frame(), av_rescale_q(), AVRational
av_rescale_q(a,b,c) 는 하나의 timestamp를 다른것으로 rescale 하는 함수 입니다. 이건 a*b/c로 계산하는데 이함수는 그 계산이 오버플로우 될 수 있기떄문에 필요합니다. AV_TIME_BASE_Q 는 AV_TIME_BASE의 분수 버전입니다. 이것은 꽤 다른 차이가 있습니다. : AV_TIME_BASE * time_in_seconds = avcodec_timestamp 그리고 AV_TIME_BASE_Q * avcodec_timestamp = time_in_seconds (AV_TIME_BASE_Q 는 사실 하나의 AVRational 오브젝트 입니다. 그래서 여러분은 avcodec에서 이것을 제어하려면 득별한 q 함수를 써야 합니다.).

Flushing our buffers
그래서 우리는 올바른 탐색을 설정합니다. 하지만 아직 끝난것이 아닙니다. 우리가 갖고 있는 하나의 큐를 packet들을 모을수 있도록 설정해야 합니다. 지금 우리는 다른 장소에 있으면 우리는 큐를 비우거나 영상에서 찾기를 중지해야 합니다. 그뿐 아니라 avcodec은 자체적으로 내부 버퍼를 갖고있고 그것 역시 각 쓰레드에 의해서 비워져야 합니다.
일단 우리는 패킷 큐를 비우는 함수를 작성합니다. 그리고 오디오, 비디오 쓰레드에 avcodec의 내부 버퍼를 비우하는 명령을 하는 방법을 정의해야 합니다.
우리는 그것을 비운다음 큐에 특별한 패킷을 넣음으로써 할 수 있습니다. 그리고 그 특별한 패킷을 찾으면 그것들은 자동으로 버퍼를 비울것입니다. 다름 간단함 flush 함수를 보겠습니다.

view plain copy to clipboard print ?

```
01.  static void packet_queue_flush(PacketQueue *q) {
02.    AVPacketList *pkt, *pkt1;
03.
04.    SDL_LockMutex(q->mutex);
05.    for(pkt = q->first_pkt; pkt != NULL; pkt = pkt1) {
06.      pkt1 = pkt->next;
07.      av_free_packet(&pkt->pkt);
08.      av_freep(&pkt);
09.    }
10.    q->last_pkt = NULL;
11.    q->first_pkt = NULL;
12.    q->nb_packets = 0;
13.    q->size = 0;
```

```
  SDL_UnlockMutex(q->mutex);
}
```

AVPacketList , SDL_LockMutex(), av_free_packet(), av_freep(), SDL_UnlockMutex()
이제 큐는 비워졌습니다. 그리고 우리의 "flush packet"을 넣습니다.
그전에 일단 우리는 그게 무엇인지를 선언합니다.

```
view plain  copy to clipboard  print  ?
AVPacket flush_pkt;

main() {
  ...
  av_init_packet(&flush_pkt);
  flush_pkt.data = "FLUSH";
  ...
}
```

AVPacket, av_init_packet()
이체 큐에 패킷을 넣습니다.

```
view plain  copy to clipboard  print  ?
} else {
  if(is->audioStream >= 0) {
    packet_queue_flush(&is->audioq);
    packet_queue_put(&is->audioq, &flush_pkt);
  }
  if(is->videoStream >= 0) {
    packet_queue_flush(&is->videoq);
    packet_queue_put(&is->videoq, &flush_pkt);
  }
}
is->seek_req = 0;
```

(This code snippet also continues the code snippet above for decode_thread.) 우리는 packet_queue_put을 바꿔야 합니다. 그래서 special flush 패킷을 복사하지 않습니다.

```
view plain  copy to clipboard  print  ?
int packet_queue_put(PacketQueue *q, AVPacket *pkt) {

  AVPacketList *pkt1;
  if(pkt != &flush_pkt && av_dup_packet(pkt) < 0) {
    return -1;
  }
```

AVPacket, AVPacketList, av_dup_packet()
그리고 오디오쓰레드와 비디오쓰레드에서 packet_queue_get 한다음 바로 avcodec_flush_buffers로 와서 이것을 넣습니다.

```
view plain  copy to clipboard  print  ?
if(packet_queue_get(&is->audioq, pkt, 1) < 0) {
  return -1;
}
if(packet->data == flush_pkt.data) {
  avcodec_flush_buffers(is->audio_st->codec);
  continue;
}
```

avcodec_flush_buffers()
전체 코드 조각은 정확히 비디오 쓰레드와 같습니다. 오디오가 비디오로 대체되고 있네요. 다 되었습니다. 여러분의 플레이어를 컴파일 해보세요

```
gcc -o tutorial07 tutorial07.c -lavutil -lavformat -lavcodec -lz -lm`sdl-config --cflags --libs`
```

다음에는 약간의 수정만 해보도록 하겠습니다. ffmpeg가 지원하는 소프트웨어 스케일링으로 작은 샘플링을 체크해 보겠습니다.

千 Line으로 비디오플레이어 만들기 7 조회수 698

9 아이유 열애설 상대 누구? '주변사람이 만류할 정...
12 착한남자, 가장 잔인했던 1초 문채원의 키스가 두...

'멀티미디어' 카테고리의 다른 글

千 Line으로 비디오플레이어 만들기 8 (0) 2009/03/11
千 Line으로 비디오플레이어 만들기 7 (0) 2009/03/09
千 Line으로 비디오플레이어 만들기 6 (0) 2009/03/03
千 Line 으로 비디오플레이어 만들기 5 (0) 2009/02/27

Posted by Real_G

이전 1 ... 604 605 606 607 608 609 610 611 612 ... 1603 다음