原 XBMC源代码分析 4：视频播放器（dvdplayer）-解码器（以ffmpeg为例）

2014年01月07日 00:03:53　阅读数：9679

XBMC分析系列文章：

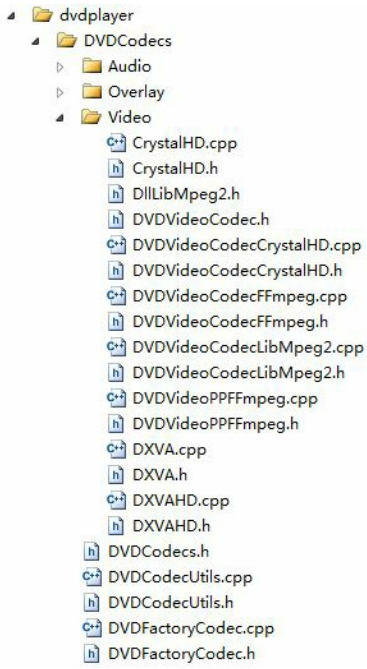
XBMC源代码分析 1：整体结构以及编译方法

XBMC源代码分析 2：Addons（皮肤Skin）

XBMC源代码分析 3：核心部分（core）-综述

本文我们分析XBMC中视频播放器（dvdplayer）中的解码器部分。由于解码器种类很多，不可能一一分析，因此以ffmpeg解码器为例进行分析。

XBMC解码器部分文件目录如下图所示：



解码器分为音频解码器和视频解码器。在这里我们看一下视频解码器中的FFMPEG解码器。对应DVDVideoCodecFFmpeg.h和DVDVideoCodecFFmpeg.cpp。

DVDVideoCodecFFmpeg.h源代码如下所示：

```cpp
/*
 * 雷霄骅
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 *
 */

#include "DVDVideoCodec.h"
#include "DVDResource.h"
#include "DllAvCodec.h"
#include "DllAvFormat.h"
#include "DllAvUtil.h"
#include "DllSwScale.h"
#include "DllAvFilter.h"
#include "DllPostProc.h"

class CCriticalSection;
//封装的FFMPEG视频解码器
class CDVDVideoCodecFFmpeg : public CDVDVideoCodec
{
public:
  class IHardwareDecoder : public IDVDResourceCounted<IHardwareDecoder>
  {
    public:
              IHardwareDecoder() {}
    virtual ~IHardwareDecoder() {};
    virtual bool Open      (AVCodecContext* avctx, const enum PixelFormat, unsigned int surfaces) = 0;
    virtual int  Decode    (AVCodecContext* avctx, AVFrame* frame) = 0;
    virtual bool GetPicture(AVCodecContext* avctx, AVFrame* frame, DVDVideoPicture* picture) = 0;
    virtual int  Check     (AVCodecContext* avctx) = 0;
```

```
    virtual int  Check     (AVCodecContext* avctx) = 0;
    virtual void Reset     () {}
    virtual unsigned GetAllowedReferences() { return 0; }
    virtual const std::string Name() = 0;
    virtual CCriticalSection* Section() { return NULL; }
  };

  CDVDVideoCodecFFmpeg();
  virtual ~CDVDVideoCodecFFmpeg();
  virtual bool Open(CDVDStreamInfo &hints, CDVDCodecOptions &options);//打开
  virtual void Dispose();//关闭
  virtual int Decode(uint8_t* pData, int iSize, double dts, double pts);//解码
  virtual void Reset();
  bool GetPictureCommon(DVDVideoPicture* pDvdVideoPicture);
  virtual bool GetPicture(DVDVideoPicture* pDvdVideoPicture);
  virtual void SetDropState(bool bDrop);
  virtual unsigned int SetFilters(unsigned int filters);
  virtual const char* GetName() { return m_name.c_str(); }; // m_name is never changed after open
  virtual unsigned GetConvergeCount();
  virtual unsigned GetAllowedReferences();

  bool               IsHardwareAllowed()                    { return !m_bSoftware; }
  IHardwareDecoder * GetHardware()                          { return m_pHardware; };
  void               SetHardware(IHardwareDecoder* hardware)
  {
    SAFE_RELEASE(m_pHardware);
    m_pHardware = hardware;
    UpdateName();
  }

protected:
  static enum PixelFormat GetFormat(struct AVCodecContext * avctx, const PixelFormat * fmt);

  int  FilterOpen(const CStdString& filters, bool scale);
  void FilterClose();
  int  FilterProcess(AVFrame* frame);

  void UpdateName()
  {
    if(m_pCodecContext->codec->name)
      m_name = CStdString("ff-") + m_pCodecContext->codec->name;
    else
      m_name = "ffmpeg";

    if(m_pHardware)
      m_name += "-" + m_pHardware->Name();
  }

  AVFrame* m_pFrame;
  AVCodecContext* m_pCodecContext;

  CStdString       m_filters;
  CStdString       m_filters_next;
  AVFilterGraph*   m_pFilterGraph;
  AVFilterContext* m_pFilterIn;
  AVFilterContext* m_pFilterOut;
#if defined(LIBAVFILTER_AVFRAME_BASED)
  AVFrame*         m_pFilterFrame;
#else
  AVFilterBufferRef* m_pBufferRef;
#endif

  int m_iPictureWidth;
  int m_iPictureHeight;

  int m_iScreenWidth;
  int m_iScreenHeight;
  int m_iOrientation;// orientation of the video in degress counter clockwise

  unsigned int m_uSurfacesCount;
  //封装Dll的各种类
  DllAvCodec m_dllAvCodec;
  DllAvUtil  m_dllAvUtil;
  DllSwScale m_dllSwScale;
  DllAvFilter m_dllAvFilter;
  DllPostProc m_dllPostProc;

  std::string m_name;
  bool              m_bSoftware;
  bool  m_isHi10p;
  IHardwareDecoder *m_pHardware;
  int m_iLastKeyframe;
  double m_dts;
  bool   m_started;
  std::vector<PixelFormat> m_formats;
};
```

该类中以下几个函数包含了解码器的几种功能：

virtual bool Open(CDVDStreamInfo &hints, CDVDCodecOptions &options);//打开

virtual void Dispose();//关闭
virtual int Decode(uint8_t* pData, int iSize, double dts, double pts);//解码
virtual void Reset();//复位

为了说明这一点，我们可以看一下视频解码器中的libmpeg2解码器，对应DVDVideoCodecLibMpeg2.h。可以看出这几个函数是一样的。

DVDVideoCodecLibMpeg2.h源代码如下：

```cpp
/*
 * 雷霄骅
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 *
 */
#include "DVDVideoCodec.h"
#include "DllLibMpeg2.h"

class CDVDVideoCodecLibMpeg2 : public CDVDVideoCodec
{
public:
  CDVDVideoCodecLibMpeg2();
  virtual ~CDVDVideoCodecLibMpeg2();
  virtual bool Open(CDVDStreamInfo &hints, CDVDCodecOptions &options);
  virtual void Dispose();
  virtual int Decode(uint8_t* pData, int iSize, double dts, double pts);
  virtual void Reset();
  virtual bool GetPicture(DVDVideoPicture* pDvdVideoPicture);
  virtual bool GetUserData(DVDVideoUserData* pDvdVideoUserData);

  virtual void SetDropState(bool bDrop);
  virtual const char* GetName() { return "libmpeg2"; }

protected:
  DVDVideoPicture* GetBuffer(unsigned int width, unsigned int height);
  inline void ReleaseBuffer(DVDVideoPicture* pPic);
  inline void DeleteBuffer(DVDVideoPicture* pPic);

  static int GuessAspect(const mpeg2_sequence_t *sequence, unsigned int *pixel_width, unsigned int *pixel_height);

  mpeg2dec_t* m_pHandle;
  const mpeg2_info_t* m_pInfo;
  DllLibMpeg2 m_dll;

  unsigned int m_irffpattern;
  bool m_bFilm; //Signals that we have film material
  bool m_bIs422;

  int m_hurry;
  double m_dts;
  double m_dts2;
  //The buffer of pictures we need
  DVDVideoPicture m_pVideoBuffer[3];
  DVDVideoPicture* m_pCurrentBuffer;
};
```

现在回到DVDVideoCodecFFmpeg.h。我们可以看一下上文所示的4个函数。

Open()

```cpp
//打开
bool CDVDVideoCodecFFmpeg::Open(CDVDStreamInfo &hints, CDVDCodecOptions &options)
{
  AVCodec* pCodec;

  if(!m_dllAvUtil.Load()
  || !m_dllAvCodec.Load()
  || !m_dllSwScale.Load()
  || !m_dllPostProc.Load()
  || !m_dllAvFilter.Load()
  ) return false;
  //注册解码器
  m_dllAvCodec.avcodec_register_all();
  m_dllAvFilter.avfilter_register_all();

  m_bSoftware     = hints.software;
  m_iOrientation  = hints.orientation;

  for(std::vector<ERenderFormat>::iterator it = options.m_formats.begin(); it != options.m_formats.end(); ++it)
  {
    m_formats.push_back((PixelFormat)CDVDCodecUtils::PixfmtFromEFormat(*it));
    if(*it == RENDER_FMT_YUV420P)
      m_formats.push_back(PIX_FMT_YUVJ420P);
  }
  m_formats.push_back(PIX_FMT_NONE); /* always add none to get a terminated list in ffmpeg world */

```

```
  pCodec = NULL;
  m_pCodecContext = NULL;

  if (hints.codec == AV_CODEC_ID_H264)
  {
    switch(hints.profile)
    {
      case FF_PROFILE_H264_HIGH_10:
      case FF_PROFILE_H264_HIGH_10_INTRA:
      case FF_PROFILE_H264_HIGH_422:
      case FF_PROFILE_H264_HIGH_422_INTRA:
      case FF_PROFILE_H264_HIGH_444_PREDICTIVE:
      case FF_PROFILE_H264_HIGH_444_INTRA:
      case FF_PROFILE_H264_CAVLC_444:
      // this is needed to not open the decoders
      m_bSoftware = true;
      // this we need to enable multithreading for hi10p via advancedsettings
      m_isHi10p = true;
      break;
    }
  }
  //查找解码器
  if(pCodec == NULL)
    pCodec = m_dllAvCodec.avcodec_find_decoder(hints.codec);

  if(pCodec == NULL)
  {
    CLog::Log(LOGDEBUG,"CDVDVideoCodecFFmpeg::Open() Unable to find codec %d", hints.codec);
    return false;
  }

  CLog::Log(LOGNOTICE,"CDVDVideoCodecFFmpeg::Open() Using codec: %s",pCodec->long_name ? pCodec->long_name : pCodec->name);

  if(m_pCodecContext == NULL)
    m_pCodecContext = m_dllAvCodec.avcodec_alloc_context3(pCodec);

  m_pCodecContext->opaque = (void*)this;
  m_pCodecContext->debug_mv = 0;
  m_pCodecContext->debug = 0;
  m_pCodecContext->workaround_bugs = FF_BUG_AUTODETECT;
  m_pCodecContext->get_format = GetFormat;
  m_pCodecContext->codec_tag = hints.codec_tag;
  /* Only allow slice threading, since frame threading is more
   * sensitive to changes in frame sizes, and it causes crashes
   * during HW accell - so we unset it in this case.
   *
   * When we detect Hi10p and user did not disable hi10pmultithreading
   * via advancedsettings.xml we keep the ffmpeg default thread type.
   * */
  if(m_isHi10p && !g_advancedSettings.m_videoDisableHi10pMultithreading)
  {
    CLog::Log(LOGDEBUG,"CDVDVideoCodecFFmpeg::Open() Keep default threading for Hi10p: %d",
                        m_pCodecContext->thread_type);
  }
  else if (CSettings::Get().GetBool("videoplayer.useframemtdec"))
  {
    CLog::Log(LOGDEBUG,"CDVDVideoCodecFFmpeg::Open() Keep default threading %d by videoplayer.useframemtdec",
                        m_pCodecContext->thread_type);
  }
  else
    m_pCodecContext->thread_type = FF_THREAD_SLICE;

#if defined(TARGET_DARWIN_IOS)
  // ffmpeg with enabled neon will crash and burn if this is enabled
  m_pCodecContext->flags &= CODEC_FLAG_EMU_EDGE;
#else
  if (pCodec->id != AV_CODEC_ID_H264 && pCodec->capabilities & CODEC_CAP_DR1
      && pCodec->id != AV_CODEC_ID_VP8
     )
    m_pCodecContext->flags |= CODEC_FLAG_EMU_EDGE;
#endif

  // if we don't do this, then some codecs seem to fail.
  m_pCodecContext->coded_height = hints.height;
  m_pCodecContext->coded_width = hints.width;
  m_pCodecContext->bits_per_coded_sample = hints.bitsperpixel;

  if( hints.extradata && hints.extrasize > 0 )
  {
    m_pCodecContext->extradata_size = hints.extrasize;
    m_pCodecContext->extradata = (uint8_t*)m_dllAvUtil.av_mallocz(hints.extrasize + FF_INPUT_BUFFER_PADDING_SIZE);
    memcpy(m_pCodecContext->extradata, hints.extradata, hints.extrasize);
  }

  // advanced setting override for skip loop filter (see avcodec.h for valid options)
  // TODO: allow per video setting?
  if (g_advancedSettings.m_iSkipLoopFilter != 0)
  {
    m_pCodecContext->skip_loop_filter = (AVDiscard)g_advancedSettings.m_iSkipLoopFilter;
  }

```

```
  // set any special options
  for(std::vector<CDVDCodecOption>::iterator it = options.m_keys.begin(); it != options.m_keys.end(); ++it)
  {
    if (it->m_name == "surfaces")
      m_uSurfacesCount = std::atoi(it->m_value.c_str());
    else
      m_dllAvUtil.av_opt_set(m_pCodecContext, it->m_name.c_str(), it->m_value.c_str(), 0);
  }

  int num_threads = std::min(8 /*MAX_THREADS*/, g_cpuInfo.getCPUCount());
  if( num_threads > 1 && !hints.software && m_pHardware == NULL // thumbnail extraction fails when run threaded
  && ( pCodec->id == AV_CODEC_ID_H264
    || pCodec->id == AV_CODEC_ID_MPEG4 ))
    m_pCodecContext->thread_count = num_threads;
  //打开解码器
  if (m_dllAvCodec.avcodec_open2(m_pCodecContext, pCodec, NULL) < 0)
  {
    CLog::Log(LOGDEBUG,"CDVDVideoCodecFFmpeg::Open() Unable to open codec");
    return false;
  }
  //初始化AVFrame
  m_pFrame = m_dllAvCodec.avcodec_alloc_frame();
  if (!m_pFrame) return false;

#if defined(LIBAVFILTER_AVFRAME_BASED)
  m_pFilterFrame = m_dllAvUtil.av_frame_alloc();
  if (!m_pFilterFrame) return false;
#endif

  UpdateName();
  return true;
}
```

Dispose()

```cpp
//关闭
void CDVDVideoCodecFFmpeg::Dispose()
{
    //释放
  if (m_pFrame) m_dllAvUtil.av_free(m_pFrame);
  m_pFrame = NULL;

#if defined(LIBAVFILTER_AVFRAME_BASED)
  m_dllAvUtil.av_frame_free(&m_pFilterFrame);
#endif

  if (m_pCodecContext)
  {
      //关闭解码器
    if (m_pCodecContext->codec) m_dllAvCodec.avcodec_close(m_pCodecContext);
    if (m_pCodecContext->extradata)
    {
      m_dllAvUtil.av_free(m_pCodecContext->extradata);
      m_pCodecContext->extradata = NULL;
      m_pCodecContext->extradata_size = 0;
    }
    m_dllAvUtil.av_free(m_pCodecContext);
    m_pCodecContext = NULL;
  }
  SAFE_RELEASE(m_pHardware);

  FilterClose();

  m_dllAvCodec.Unload();
  m_dllAvUtil.Unload();
  m_dllAvFilter.Unload();
  m_dllPostProc.Unload();
}
```

Decode()

```cpp
//解码
int CDVDVideoCodecFFmpeg::Decode(uint8_t* pData, int iSize, double dts, double pts)
{
  int iGotPicture = 0, len = 0;

  if (!m_pCodecContext)
    return VC_ERROR;

  if(pData)
    m_iLastKeyframe++;

  shared_ptr<CSingleLock> lock;
  if(m_pHardware)
```

```
  {
    CCriticalSection* section = m_pHardware->Section();
    if(section)
      lock = shared_ptr<CSingleLock>(new CSingleLock(*section));

    int result;
    if(pData)
      result = m_pHardware->Check(m_pCodecContext);
    else
      result = m_pHardware->Decode(m_pCodecContext, NULL);

    if(result)
      return result;
  }

  if(m_pFilterGraph)
  {
    int result = 0;
    if(pData == NULL)
      result = FilterProcess(NULL);
    if(result)
      return result;
  }

  m_dts = dts;
  m_pCodecContext->reordered_opaque = pts_dtoi(pts);
  //初始化AVPacket
  AVPacket avpkt;
  m_dllAvCodec.av_init_packet(&avpkt);
  avpkt.data = pData;
  avpkt.size = iSize;
  /* We lie, but this flag is only used by pngdec.c.
   * Setting it correctly would allow CorePNG decoding. */
  avpkt.flags = AV_PKT_FLAG_KEY;
  //解码
  len = m_dllAvCodec.avcodec_decode_video2(m_pCodecContext, m_pFrame, &iGotPicture, &avpkt);

  if(m_iLastKeyframe < m_pCodecContext->has_b_frames + 2)
    m_iLastKeyframe = m_pCodecContext->has_b_frames + 2;

  if (len < 0)
  {
    CLog::Log(LOGERROR, "%s - avcodec_decode_video returned failure", __FUNCTION__);
    return VC_ERROR;
  }

  if (!iGotPicture)
    return VC_BUFFER;

  if(m_pFrame->key_frame)
  {
    m_started = true;
    m_iLastKeyframe = m_pCodecContext->has_b_frames + 2;
  }

  /* put a limit on convergence count to avoid huge mem usage on streams without keyframes */
  if(m_iLastKeyframe > 300)
    m_iLastKeyframe = 300;

  /* h264 doesn't always have keyframes + won't output before first keyframe anyway */
  if(m_pCodecContext->codec_id == AV_CODEC_ID_H264
  || m_pCodecContext->codec_id == AV_CODEC_ID_SVQ3)
    m_started = true;

  if(m_pHardware == NULL)
  {
    bool need_scale = std::find( m_formats.begin()
                               , m_formats.end()
                               , m_pCodecContext->pix_fmt) == m_formats.end();

    bool need_reopen  = false;
    if(!m_filters.Equals(m_filters_next))
      need_reopen = true;

    if(m_pFilterIn)
    {
      if(m_pFilterIn->outputs[0]->format != m_pCodecContext->pix_fmt
      || m_pFilterIn->outputs[0]->w      != m_pCodecContext->width
      || m_pFilterIn->outputs[0]->h      != m_pCodecContext->height)
        need_reopen = true;
    }

    // try to setup new filters
    if (need_reopen || (need_scale && m_pFilterGraph == NULL))
    {
      m_filters = m_filters_next;

      if(FilterOpen(m_filters, need_scale) < 0)
        FilterClose();
    }
  }
```

```
105.
106.   int result;
107.   if(m_pHardware)
108.     result = m_pHardware->Decode(m_pCodecContext, m_pFrame);
109.   else if(m_pFilterGraph)
110.     result = FilterProcess(m_pFrame);
111.   else
112.     result = VC_PICTURE | VC_BUFFER;
113.
114.   if(result & VC_FLUSHED)
115.     Reset();
116.
117.   return result;
118. }
```

Reset()

```
[cpp]
1.  //复位
2.  void CDVDVideoCodecFFmpeg::Reset()
3.  {
4.    m_started = false;
5.    m_iLastKeyframe = m_pCodecContext->has_b_frames;
6.    m_dllAvCodec.avcodec_flush_buffers(m_pCodecContext);
7.
8.    if (m_pHardware)
9.      m_pHardware->Reset();
10.
11.   m_filters = "";
12.   FilterClose();
13. }
```


文章标签： xbmc 源代码 ffmpeg 媒体中心 播放器

个人分类： XBMC FFMPEG

所属专栏： 开源多媒体项目源代码分析