

原 ffdshow 源代码分析 8：视频解码器类（TvideoCodecDec）

2013年11月13日 00:06:50 阅读数：5587

=====

ffdshow源代码分析系列文章列表：

[ffdshow 源代码分析 1：整体结构](#)

[ffdshow 源代码分析 2：位图覆盖滤镜（对话框部分Dialog）](#)

[ffdshow 源代码分析 3：位图覆盖滤镜（设置部分Settings）](#)

[ffdshow 源代码分析 4：位图覆盖滤镜（滤镜部分Filter）](#)

[ffdshow 源代码分析 5：位图覆盖滤镜（总结）](#)

[ffdshow 源代码分析 6：对解码器的dll的封装（libavcodec）](#)

[ffdshow 源代码分析 7：libavcodec视频解码器类（TvideoCodecLibavcodec）](#)

[ffdshow 源代码分析 8：视频解码器类（TvideoCodecDec）](#)

[ffdshow 源代码分析 9：编解码器有关类的总结](#)

=====



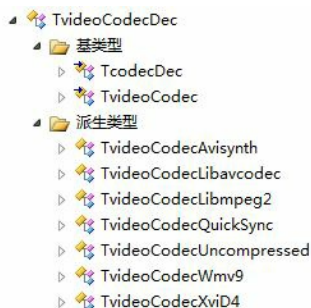
前面两篇文章介绍了ffdshow中libavcodec的封装类Tlibavcodec，以及libavcodec的解码器类TvideoCodecLibavcodec：

[ffdshow 源代码分析 6：对解码器的dll的封装（libavcodec）](#)

[ffdshow 源代码分析 7：解码器类（TvideoCodecLibavcodec）](#)

其中libavcodec的解码器类TvideoCodecLibavcodec通过调用Tlibavcodec中的方法实现了libavcodec的dll中方法的调用；而它继承了TvideoCodecDec，本文正是要分析它继承的这个类。

TvideoCodecDec是所有视频解码器共有的父类。可以看一下它的继承关系：



可见，除了TvideoCodecLibavcodec继承了TvideoCodecDec之外，还有好几个类继承了TvideoCodecDec，比如说：TvideoCodecLibmpeg2，TvideoCodecXviD4等等…。突然来了兴趣，我们可以看一下其他的解码器类的定义是什么样的。

TvideoCodecLibmpeg2定义如下：

```

1.  /*
2.  *雷霄骅
3.  *leixiaohua1020@126.com
4.  *中国传媒大学/数字电视技术
5.  */
6.  #ifndef _TVIDEOCODECLIBMPPEG2_H_
7.  #define _TVIDEOCODECLIBMPPEG2_H_
8.
9.  #include "TvideoCodec.h"
10. #include "libmpeg2/include/mpeg2.h"
11.
12. class Tdll;
13. struct Textradata;
14. class TccDecoder;
15. //libmpeg2解码器
16. class TvideoCodecLibmpeg2 : public TvideoCodecDec
17. {
18. private:
19.     Tdll *dll;
20.     uint32_t (*mpeg2_set_accel)(uint32_t accel);
21.     mpeg2dec_t* (*mpeg2_init)(void);
22.     const mpeg2_info_t* (*mpeg2_info)(mpeg2dec_t *mpeg2dec);
23.     mpeg2_state_t (*mpeg2_parse)(mpeg2dec_t *mpeg2dec);
24.     void (*mpeg2_buffer)(mpeg2dec_t *mpeg2dec, const uint8_t *start, const uint8_t *end);
25.     void (*mpeg2_close)(mpeg2dec_t *mpeg2dec);
26.     void (*mpeg2_reset)(mpeg2dec_t *mpeg2dec, int full_reset);
27.     void (*mpeg2_set_rtStart)(mpeg2dec_t *mpeg2dec, int64_t rtStart);
28.     int (*mpeg2_guess_aspect)(const mpeg2_sequence_t * sequence, unsigned int * pixel_width, unsigned int * pixel_height);
29.
30.     mpeg2dec_t *mpeg2dec;
31.     const mpeg2_info_t *info;
32.     bool wait4Iframe;
33.     int sequenceFlag;
34.     REFERENCE_TIME avgTimePerFrame;
35.     TffPict oldpict;
36.     Textradata *extradata;
37.     TccDecoder *ccDecoder;
38.     Tbuffer *buffer;
39.     uint32_t oldflags;
40.     bool m_fFilm;
41.     int SetDeinterlaceMethod(void);
42.
43.     void init(void);
44.     HRESULT decompressI(const unsigned char *src, size_t srcLen, IMediaSample *pIn);
45.
46. protected:
47.     virtual bool beginDecompress(TffPictBase &pict, FOURCC infcc, const CMediaType &mt, int sourceFlags);
48.
49. public:
50.     TvideoCodecLibmpeg2(IffdshowBase *Ideci, IdecVideoSink *Isink);
51.     virtual ~TvideoCodecLibmpeg2();
52.
53.     static const char_t *dllname;
54.     virtual int getType(void) const {
55.         return IDFF_MOVIE_LIBMPPEG2;
56.     }
57.     virtual int caps(void) const {
58.         return CAPS::VIS_QUANTS;
59.     }
60.
61.     virtual void end(void);
62.     virtual HRESULT decompress(const unsigned char *src, size_t srcLen, IMediaSample *pIn);
63.     virtual bool onSeek(REFERENCE_TIME segmentStart);
64.     virtual HRESULT BeginFlush();
65. };
66.
67. #endif

```

TvideoCodecXviD4定义如下：

```

1.  /*
2.  *雷霄骅
3.  *leixiaohua1020@126.com
4.  *中国传媒大学/数字电视技术
5.  */
6.  #ifndef _TVIDEOCODECXVID4_H_
7.  #define _TVIDEOCODECXVID4_H_
8.
9.  #include "TvideoCodec.h"
10.
11. class Tdll;
12. struct Textradata;
13. //xvid解码器
14. class TvideoCodecXviD4 : public TvideoCodecDec
15. {
16. private:
17.     void create(void);
18.     Tdll *dll;
19. public:
20.     TvideoCodecXviD4(IffdshowBase *Ideci, IdecVideoSink *IsinkD);
21.     virtual ~TvideoCodecXviD4();
22.     int (*xvid_global)(void *handle, int opt, void *param1, void *param2);
23.     int (*xvid_decore)(void *handle, int opt, void *param1, void *param2);
24.     int (*xvid_plugin_single)(void *handle, int opt, void *param1, void *param2);
25.     int (*xvid_plugin_lumimasking)(void *handle, int opt, void *param1, void *param2);
26.     static const char_t *dllname;
27. private:
28.     void *enchandle, *dechandle;
29.     int psnr;
30.     TffPict pict;
31.     Tbuffer pictbuf;
32.     static int me_hq(int rd3), me_(int me3);
33.     Textradata *extradata;
34.     REFERENCE_TIME rtStart, rtStop;
35. protected:
36.     virtual bool beginDecompress(TffPictBase &pict, FOURCC infcc, const CMediaType &mt, int sourceFlags);
37.     virtual HRESULT flushDec(void);
38. public:
39.     virtual int getType(void) const {
40.         return IDFF_MOVIE_XVID4;
41.     }
42.     virtual int caps(void) const {
43.         return CAPS::VIS_QUANTS;
44.     }
45.
46.     virtual HRESULT decompress(const unsigned char *src, size_t srcLen, IMediaSample *pIn);
47. };
48.
49. #endif

```

从以上这2种解码器类的定义，我们可以看出一些规律，比如说：

1. 都有Tdll *dll这个变量，用于加载视频解码器的dll
2. 都有beginDecompress()函数，用于初始化解码器
3. 都有decompress()函数，用于解码

好了，闲话不说，回归正题，来看一下这些解码器共有的父类：TvideoCodecDec

```

1. //具体 视频 解码器的父类, 存一些公共信息
2. class TvideoCodecDec : virtual public TvideoCodec, virtual public TcodecDec
3. {
4. protected:
5.     bool isdvdproc;
6.     comptrQ<IffdshowDecVideo> deciV;
7.     IdecVideoSink *sinkD;
8.     TvideoCodecDec(IffdshowBase *Ideci, IdecVideoSink *Isink);
9.     Rational guessMPEG2sar(const Trect &r, const Rational &sar2, const Rational &containerSar);
10.
11.     class TtelecineManager
12.     {
13.     private:
14.         TvideoCodecDec* parent;
15.         int segment_count;
16.         int pos_in_group;
17.         struct {
18.             int fieldtype;
19.             int repeat_pict;
20.             REFERENCE_TIME rtStart;
21.         } group[2]; // store information about 2 recent frames.
22.         REFERENCE_TIME group_rtStart;
23.         bool film;
24.         int cfg_softTelecine;
25.     public:
26.         TtelecineManager(TvideoCodecDec* Iparent);
27.         void get_timestamps(TffPict &pict);
28.         void get_fieldtype(TffPict &pict);
29.         void new_frame(int top_field_first, int repeat_pict, const REFERENCE_TIME &rtStart, const REFERENCE_TIME &rtStop);
30.         void onSeek(void);
31.     } telecineManager;
32.
33. public:
34.     static TvideoCodecDec* initDec(IffdshowBase *deci, IdecVideoSink *Isink, AVCodecID codecId, FOURCC fcc, const CMediaType &mt);
35.
36.     virtual ~TvideoCodecDec();
37.
38.     virtual int caps(void) const {
39.         return CAPS::NONE;
40.     }
41.     virtual bool testMediaType(FOURCC fcc, const CMediaType &mt) {
42.         return true;
43.     }
44.     virtual void forceOutputColorspace(const BITMAPINFOHEADER *hdr, int *ilace, TcspInfos &forcedCsps) {
45.         *ilace = 0; //cspInfos of forced output colorspace, empty when entering function
46.     }
47.     enum {SOURCE_REORDER = 1};
48.     virtual bool beginDecompress(TffPictBase &pict, FOURCC infcc, const CMediaType &mt, int sourceFlags) = 0;
49.     virtual HRESULT decompress(const unsigned char *src, size_t srcLen, IMediaSample *pIn) = 0;
50.     virtual bool onDiscontinuity(void) {
51.         return false;
52.     }
53.     virtual HRESULT onEndOfStream(void) {
54.         return S_OK;
55.     }
56.
57.     unsigned int quantsDx, quantsStride, quantsDy, quantBytes, quantType;
58.     //QP表
59.     void *quants;
60.     uint16_t *intra_matrix, *inter_matrix;
61.     //计算平均QP
62.     float calcMeanQuant(void);
63.     //画运动矢量
64.     virtual bool drawMV(unsigned char *dst, unsigned int dx, stride_t stride, unsigned int dy) const {
65.         return false;
66.     }
67.     virtual const char* get_current_idct(void) {
68.         return NULL;
69.     }
70.     virtual int useDXVA(void) {
71.         return 0;
72.     };
73.
74.     virtual void setOutputPin(IPin * /*pPin*/) {}
75. };

```

TvideoCodecDec这个类中, 还定义了一个类TtelecineManager。这种在类里面再定义一个类的方式还是不太多见的。TtelecineManager这个类的作用还没有研究, 先不管它。

可以看出, TvideoCodecDec类的定义并不复杂, 最主要的变量有如下几个, 这几个变量都是子类中会用到的:

comptrQ<IffdshowDecVideo>deciV: 重要性不言而喻, 回头介绍

IdecVideoSink *sinkD: 重要性不言而喻, 回头介绍

void *quants: QP表 (为什么要存在这里还没搞清)

TvideoCodecDec类定义了几个函数:

initDec()：初始化解码器（重要）

calcMeanQuant()：计算平均QP（为什么要在这里计算还没搞清）

TvideoCodecDec类还定义了一些纯虚函数，作为接口，这些函数的实现都在TvideoCodecDec的子类中完成【这几个函数是最重要的】：

```
beginDecompress();  
decompress();
```

TvideoCodecDec类中最重要的函数只有一个，就是initDec()，作用主要是初始化解码器。其他的很多函数大多只是定义了一个名称，并没有实现，因为都是打算在具体各种解码器类中再进行实现的。

看一下initDec()的代码：



```

1. TvideoCodecDec* TvideoCodecDec::initDec(IffdshowBase *deci, IdecVideoSink *sink, AVCodecID codecId, FOURCC fcc, const CMediaType &mt
2. )
3. {
4.     // DXVA mode is a preset setting
5.     switch (codecId) {
6.         case AV_CODEC_ID_H264:
7.             if (deci->getParam2(IDFF_filterMode) & IDFF_FILTERMODE_VIDEO_DXVA) {
8.                 if (deci->getParam2(IDFF_dec_DXVA_H264)) {
9.                     codecId = CODEC_ID_H264_DXVA;
10.                } else {
11.                    return NULL;
12.                }
13.            }
14.            break;
15.        case AV_CODEC_ID_VC1:
16.        case CODEC_ID_WMV9_LIB:
17.            if (deci->getParam2(IDFF_filterMode) & IDFF_FILTERMODE_VIDEO_DXVA) {
18.                if (deci->getParam2(IDFF_dec_DXVA_VC1)) {
19.                    codecId = CODEC_ID_VC1_DXVA;
20.                } else {
21.                    return NULL;
22.                }
23.            }
24.            break;
25.        default:
26.            break;
27.    }
28.
29.    TvideoCodecDec *movie = NULL;
30.
31.    if (is_quicksync_codec(codecId)) {
32.        movie = new TvideoCodecQuickSync(deci, sink, codecId);
33.    } else if (lavc_codec(codecId)) {
34.        movie = new TvideoCodecLibavcodec(deci, sink);
35.    } else if (raw_codec(codecId)) {
36.        movie = new TvideoCodecUncompressed(deci, sink);
37.    } else if (wmv9_codec(codecId)) {
38.        movie = new TvideoCodecWmv9(deci, sink);
39.    } else if (codecId == CODEC_ID_XVID4) {
40.        movie = new TvideoCodecXviD4(deci, sink);
41.    } else if (codecId == CODEC_ID_LIBMPEG2) {
42.        movie = new TvideoCodecLibmpeg2(deci, sink);
43.    } else if (codecId == CODEC_ID_AVISYNTH) {
44.        movie = new TvideoCodecAvisynth(deci, sink);
45.    } else if (codecId == CODEC_ID_H264_DXVA || codecId == CODEC_ID_VC1_DXVA) {
46.        movie = new TvideoCodecLibavcodecDxva(deci, sink, codecId);
47.    } else {
48.        return NULL;
49.    }
50.
51.    if (!movie) {
52.        return NULL;
53.    }
54.
55.    if (movie->ok && movie->testMediaType(fcc, mt)) {
56.        movie->codecId = codecId;
57.        return movie;
58.    } else if (is_quicksync_codec(codecId)) {
59.        // QuickSync decoder init failed, revert to internal decoder.
60.        switch (codecId) {
61.            case CODEC_ID_H264_QUICK_SYNC:
62.                codecId = AV_CODEC_ID_H264;
63.                break;
64.            case CODEC_ID_MPEG2_QUICK_SYNC:
65.                codecId = CODEC_ID_LIBMPEG2;
66.                break;
67.            case CODEC_ID_VC1_QUICK_SYNC:
68.                codecId = CODEC_ID_WMV9_LIB;
69.                break;
70.            default:
71.                ASSERT(FALSE); // this shouldn't happen!
72.        }
73.
74.        delete movie;
75.
76.        // Call this function again with the new codecId.
77.        return initDec(deci, sink, codecId, fcc, mt);
78.    } else {
79.        delete movie;
80.        return NULL;
81.    }
82. }

```

这个函数的功能还是比较好理解的，根据CodecID的不同，创建不同的解码器（从TvideoCodecLibavcodec，TvideoCodecXviD4，TvideoCodecLibmpeg2这些里面选择）。

虽然不知道用途是什么，但是我们可以顺便看一下计算平均QP的函数，就是把quants1指向的QP表里面的数据求了一个平均值：

```
[cpp]    
1. //计算平均QP  
2. float TvideoCodecDec::calcMeanQuant(void)  
3. {  
4.     if (!quants || !quantsDx || !quantsDy) {  
5.         return 0;  
6.     }  
7.     unsigned int sum = 0, num = quantsDx * quantsDy;  
8.     unsigned char *quants1 = (unsigned char*)quants;  
9.     for (unsigned int y = 0; y < quantsDy; y++)  
10.        for (unsigned int x = 0; x < quantsDx; x++) {  
11.            sum += quants1[(y * quantsStride + x) * quantBytes];  
12.        }  
13.     return float(sum) / num;  
14. }
```

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/15493743>

文章标签： ffdshow 视频 解码器 xvid libmpeg2

个人分类： [ffdshow](#)

所属专栏： [开源多媒体项目源代码分析](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com