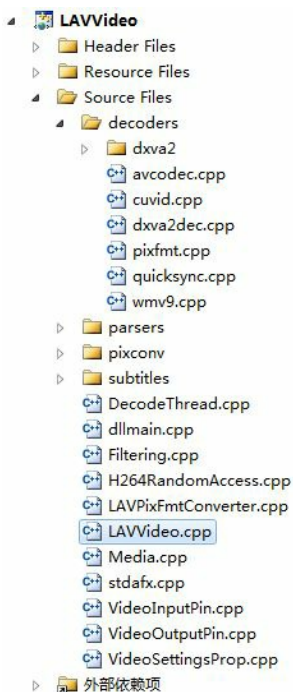


原 LAV Filter 源代码分析 3：LAV Video （1）

2013年10月27日 20:58:00 阅读数：6519

LAV Video 是使用很广泛的DirectShow Filter,它封装了FFMPEG中的libavcodec,支持十分广泛的视频格式的解码。在这里对其源代码进行详细的分析。

LAV Video 工程代码的结构如下图所示



直接看LAV Video最主要的类CLAVVideo吧，它的定义位于LAVVideo.h中。

LAVVideo.h

```
[cpp]
1.  /*  雷霄骅
2.  *  中国传媒大学/数字电视技术
3.  *  leixiaohua1020@126.com
4.  *
5.  */
6.  /*
7.  *      Copyright (C) 2010-2013 Hendrik Leppkes
8.  *      http://www.1f0.de
9.  *
10. *  This program is free software; you can redistribute it and/or modify
11. *  it under the terms of the GNU General Public License as published by
12. *  the Free Software Foundation; either version 2 of the License, or
13. *  (at your option) any later version.
14. *
15. *  This program is distributed in the hope that it will be useful,
16. *  but WITHOUT ANY WARRANTY; without even the implied warranty of
17. *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18. *  GNU General Public License for more details.
19. *
20. *  You should have received a copy of the GNU General Public License along
21. *  with this program; if not, write to the Free Software Foundation, Inc.,
22. *  51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
23. */
24.
25. #pragma once
26.
27. #include "decoders/ILAVDecoder.h"
28. #include "DecodeThread.h"
29. #include "ILAVPinInfo.h"
30.
31. #include "LAVPixFmtConverter.h"
32. #include "LAVVideoSettings.h"
33. #include "H264RandomAccess.h"
34. #include "FloatingAverage.h"
35.
36. #include "ISpecifyPropertyPages2.h"
37. #include "SynchronizedQueue.h"
38.
39. #include "subtitles/LAVSubtitleConsumer.h"
40. #include "subtitles/LAVVideoSubtitleInputPin.h"
```

```

41.
42. #include "BaseTrayIcon.h"
43.
44. #define LAVC_VIDEO_REGISTRY_KEY L"Software\\LAV\\Video"
45. #define LAVC_VIDEO_REGISTRY_KEY_FORMATS L"Software\\LAV\\Video\\Formats"
46. #define LAVC_VIDEO_REGISTRY_KEY_OUTPUT L"Software\\LAV\\Video\\Output"
47. #define LAVC_VIDEO_REGISTRY_KEY_HWACCEL L"Software\\LAV\\Video\\HWAccel"
48.
49. #define LAVC_VIDEO_LOG_FILE L"LAVVideo.txt"
50.
51. #define DEBUG_FRAME_TIMINGS 0
52. #define DEBUG_PIXELCONV_TIMINGS 0
53.
54. #define LAV_MT_FILTER_QUEUE_SIZE 4
55.
56. typedef struct {
57.     REFERENCE_TIME rtStart;
58.     REFERENCE_TIME rtStop;
59. } TimingCache;
60. //解码核类型
61. //Transform Filter
62. [uuid("EE30215D-164F-4A92-A4EB-9D4C13390F9F")]
63. class CLAVVideo : public CTransformFilter, public ISpecifyPropertyPages2, public ILAVVideoSettings, public ILAVVideoStatus, public ILAVVideoCallback
64. {
65. public:
66.     CLAVVideo(LPUNKNOWN pUnk, HRESULT* phr);
67.     ~CLAVVideo();
68.
69.     static void CALLBACK StaticInit(BOOL bLoading, const CLSID *clsid);
70.
71.     // IUnknown
72.     // 查找接口必须实现
73.     DECLARE_IUNKNOWN;
74.     STDMETHODIMP NonDelegatingQueryInterface(REFIID riid, void** ppv);
75.
76.     // ISpecifyPropertyPages2
77.     // 属性页
78.     // 获取或者创建
79.     STDMETHODIMP GetPages(CAUUID *pPages);
80.     STDMETHODIMP CreatePage(const GUID& guid, IPropertyPage** ppPage);
81.
82.     // ILAVVideoSettings
83.
84.     STDMETHODIMP SetRuntimeConfig(BOOL bRuntimeConfig);
85.     STDMETHODIMP SetFormatConfiguration(LAVVideoCodec vCodec, BOOL bEnabled);
86.     STDMETHODIMP_(BOOL) GetFormatConfiguration(LAVVideoCodec vCodec);
87.     STDMETHODIMP SetNumThreads(DWORD dwNum);
88.     STDMETHODIMP_(DWORD) GetNumThreads();
89.     STDMETHODIMP SetStreamAR(DWORD bStreamAR);
90.     STDMETHODIMP_(DWORD) GetStreamAR();
91.     STDMETHODIMP SetPixelFormat(LAVOutPixFmts pixFmt, BOOL bEnabled);
92.     STDMETHODIMP_(BOOL) GetPixelFormat(LAVOutPixFmts pixFmt);
93.     STDMETHODIMP SetRGBOutputRange(DWORD dwRange);
94.     STDMETHODIMP_(DWORD) GetRGBOutputRange();
95.
96.     STDMETHODIMP SetDeintFieldOrder(LAVDeintFieldOrder fieldOrder);
97.     STDMETHODIMP_(LAVDeintFieldOrder) GetDeintFieldOrder();
98.     STDMETHODIMP SetDeintForce(BOOL bForce);
99.     STDMETHODIMP_(BOOL) GetDeintForce();
100.     STDMETHODIMP SetDeintAggressive(BOOL bAggressive);
101.     STDMETHODIMP_(BOOL) GetDeintAggressive();
102.
103.     STDMETHODIMP_(DWORD) CheckHWAccelSupport(LAVHWAccel hwAccel);
104.     STDMETHODIMP SetHWAccel(LAVHWAccel hwAccel);
105.     STDMETHODIMP_(LAVHWAccel) GetHWAccel();
106.     STDMETHODIMP SetHWAccelCodec(LAVVideoHWCodec hwAccelCodec, BOOL bEnabled);
107.     STDMETHODIMP_(BOOL) GetHWAccelCodec(LAVVideoHWCodec hwAccelCodec);
108.     STDMETHODIMP SetHWAccelDeintMode(LAVHWDeintModes deintMode);
109.     STDMETHODIMP_(LAVHWDeintModes) GetHWAccelDeintMode();
110.     STDMETHODIMP SetHWAccelDeintOutput(LAVDeintOutput deintOutput);
111.     STDMETHODIMP_(LAVDeintOutput) GetHWAccelDeintOutput();
112.     STDMETHODIMP SetHWAccelDeintHQ(BOOL bHQ);
113.     STDMETHODIMP_(BOOL) GetHWAccelDeintHQ();
114.     STDMETHODIMP SetSWDeintMode(LAVSWDeintModes deintMode);
115.     STDMETHODIMP_(LAVSWDeintModes) GetSWDeintMode();
116.     STDMETHODIMP SetSWDeintOutput(LAVDeintOutput deintOutput);
117.     STDMETHODIMP_(LAVDeintOutput) GetSWDeintOutput();
118.
119.     STDMETHODIMP SetDeintTreatAsProgressive(BOOL bEnabled);
120.     STDMETHODIMP_(BOOL) GetDeintTreatAsProgressive();
121.
122.     STDMETHODIMP SetDitherMode(LAVDitherMode ditherMode);
123.     STDMETHODIMP_(LAVDitherMode) GetDitherMode();
124.
125.     STDMETHODIMP SetUseMSWMV9Decoder(BOOL bEnabled);
126.     STDMETHODIMP_(BOOL) GetUseMSWMV9Decoder();
127.
128.     STDMETHODIMP SetDVDVideoSupport(BOOL bEnabled);
129.     STDMETHODIMP_(BOOL) GetDVDVideoSupport();
130.
131.     STDMETHODIMP SetHWAccelDeintHQ(BOOL bHQ);

```

```

131.     STDMETHODIMP SetHWAccelResolutionFlags(DWORD dwKestFlags);
132.     STDMETHODIMP_(DWORD) GetHWAccelResolutionFlags();
133.
134.     STDMETHODIMP SetTrayIcon(BOOL bEnabled);
135.     STDMETHODIMP_(BOOL) GetTrayIcon();
136.
137.     STDMETHODIMP SetDeinterlacingMode(LAVDeintMode deintMode);
138.     STDMETHODIMP_(LAVDeintMode) GetDeinterlacingMode();
139.
140.     // ILAVVideoStatus
141.     STDMETHODIMP_(const WCHAR *) GetActiveDecoderName() { return m_Decoder.GetDecoderName(); }
142.
143.     // CTransformFilter
144.     // 核心的
145.     HRESULT CheckInputType(const CMediaType* mtIn);
146.     HRESULT CheckTransform(const CMediaType* mtIn, const CMediaType* mtOut);
147.     HRESULT DecideBufferSize(IMemAllocator * pAllocator, ALLOCATOR_PROPERTIES *pprop);
148.     HRESULT GetMediaType(int iPosition, CMediaType *pMediaType);
149.
150.     HRESULT SetMediaType(PIN_DIRECTION dir, const CMediaType *pmt);
151.     HRESULT EndOfStream();
152.     HRESULT BeginFlush();
153.     HRESULT EndFlush();
154.     HRESULT NewSegment(REFERENCE_TIME tStart, REFERENCE_TIME tStop, double dRate);
155.     //处理的核心
156.     //核心一般才有IMediaSample
157.     HRESULT Receive(IMediaSample *pIn);
158.
159.     HRESULT CheckConnect(PIN_DIRECTION dir, IPin *pPin);
160.     HRESULT BreakConnect(PIN_DIRECTION dir);
161.     HRESULT CompleteConnect(PIN_DIRECTION dir, IPin *pReceivePin);
162.
163.     int GetPinCount();
164.     CBasePin* GetPin(int n);
165.
166.     STDMETHODIMP JoinFilterGraph(IFilterGraph * pGraph, LPCWSTR pName);
167.
168.     // ILAVVideoCallback
169.     STDMETHODIMP AllocateFrame(LAVFrame **ppFrame);
170.     STDMETHODIMP ReleaseFrame(LAVFrame **ppFrame);
171.     STDMETHODIMP Deliver(LAVFrame *pFrame);
172.     STDMETHODIMP_(LPWSTR) GetFileExtension();
173.     STDMETHODIMP_(BOOL) FilterInGraph(PIN_DIRECTION dir, const GUID &clsid) { if (dir == PINDIR_INPUT) return FilterInGraphSafe(m_pInput, clsid); else return FilterInGraphSafe(m_pOutput, clsid); }
174.     STDMETHODIMP_(DWORD) GetDecodeFlags() { return m_dwDecodeFlags; }
175.     STDMETHODIMP_(CMediaType&) GetInputMediaType() { return m_pInput->CurrentMediaType(); }
176.     STDMETHODIMP GetLAVPinInfo(LAVPinInfo &info) { if (m_LAVPinInfoValid) { info = m_LAVPinInfo; return S_OK; } return E_FAIL; }
177.     STDMETHODIMP_(CBasePin*) GetOutputPin() { return m_pOutput; }
178.     STDMETHODIMP_(CMediaType&) GetOutputMediaType() { return m_pOutput->CurrentMediaType(); }
179.     STDMETHODIMP DVDStripPacket(BYTE* p, long& len) { static_cast<CDeCSSTransformInputPin*>(m_pInput)->StripPacket(p, len); return S_OK; }
180.     STDMETHODIMP_(LAVFrame*) GetFlushFrame();
181.     STDMETHODIMP ReleaseAllDXVAResources() { ReleaseLastSequenceFrame(); return S_OK; }
182.
183. public:
184.     // Pin Configuration
185.     const static AMOVIESETUP_MEDIATYPE sudPinTypesIn[];
186.     const static int sudPinTypesInCount;
187.     const static AMOVIESETUP_MEDIATYPE sudPinTypesOut[];
188.     const static int sudPinTypesOutCount;
189.
190. private:
191.     HRESULT LoadDefaults();
192.     HRESULT ReadSettings(HKEY rootKey);
193.     HRESULT LoadSettings();
194.     HRESULT SaveSettings();
195.
196.     HRESULT CreateTrayIcon();
197.
198.     HRESULT CreateDecoder(const CMediaType *pmt);
199.
200.     HRESULT GetDeliveryBuffer(IMediaSample** ppOut, int width, int height, AVRational ar, DXVA2_ExtendedFormat dxvaExtFormat, REFERENCE_TIME avgFrameDuration);
201.     HRESULT ReconnectOutput(int width, int height, AVRational ar, DXVA2_ExtendedFormat dxvaExtFlags, REFERENCE_TIME avgFrameDuration, BOOL bDXVA = FALSE);
202.
203.     HRESULT SetFrameFlags(IMediaSample* pMS, LAVFrame *pFrame);
204.
205.     HRESULT NegotiatePixelFormat(CMediaType &mt, int width, int height);
206.     BOOL IsInterlaced();
207.
208.
209.     HRESULT Filter(LAVFrame *pFrame);
210.     HRESULT DeliverToRenderer(LAVFrame *pFrame);
211.
212.     HRESULT PerformFlush();
213.     HRESULT ReleaseLastSequenceFrame();
214.
215.     HRESULT GetD3DBuffer(LAVFrame *pFrame);
216.     HRESULT RedrawStillImage();
217.     HRESULT SetInDVDMenu(BOOL menu) { m_bInDVDMenu = menu; return S_OK; }
218.

```

```

219.     enum {CNTRL_EXIT, CNTRL_REDRAW};
220.     HRESULT ControlCmd(DWORD cmd) {
221.         return m_ControlThread->CallWorker(cmd);
222.     }
223.
224. private:
225.     friend class CVideoOutputPin;
226.     friend class CDecodeThread;
227.     friend class CLAVControlThread;
228.     friend class CLAVSubtitleProvider;
229.     friend class CLAVSubtitleConsumer;
230.     //解码线程
231.     CDecodeThread      m_Decoder;
232.     CAMThread          *m_ControlThread;
233.
234.     REFERENCE_TIME     m_rtPrevStart;
235.     REFERENCE_TIME     m_rtPrevStop;
236.
237.     BOOL               m_bForceInputAR;
238.     BOOL               m_bSendMediaType;
239.     BOOL               m_bFlushing;
240.
241.     HRESULT            m_hrDeliver;
242.
243.     CLAVPixFmtConverter m_PixFmtConverter;
244.     std::wstring       m_strExtension;
245.
246.     DWORD              m_bDXVAExtFormatSupport;
247.     DWORD              m_bMadVR;
248.     DWORD              m_bOverlayMixer;
249.     DWORD              m_dwDecodeFlags;
250.
251.     BOOL               m_bInDVDMenu;
252.
253.     AVFilterGraph      *m_pFilterGraph;
254.     AVFilterContext     *m_pFilterBufferSrc;
255.     AVFilterContext     *m_pFilterBufferSink;
256.
257.     LAVPixelFormat      m_filterPixFmt;
258.     int                 m_filterWidth;
259.     int                 m_filterHeight;
260.     LAVFrame            m_FilterPrevFrame;
261.
262.     BOOL               m_LAVPinInfoValid;
263.     LAVPinInfo         m_LAVPinInfo;
264.
265.     CLAVVideoSubtitleInputPin *m_pSubtitleInput;
266.     CLAVSubtitleConsumer *m_SubtitleConsumer;
267.
268.     LAVFrame           *m_pLastSequenceFrame;
269.
270.     AM_SimpleRateChange m_DVDRate;
271.
272.     BOOL               m_bRuntimeConfig;
273.     struct VideoSettings {
274.         BOOL TrayIcon;
275.         DWORD StreamAR;
276.         DWORD NumThreads;
277.         BOOL bFormats[Codec_VideoNB];
278.         BOOL bMSWMV9DMO;
279.         BOOL bPixFmts[LAVOutPixFmt_NB];
280.         DWORD RGBRange;
281.         DWORD HWAccel;
282.         BOOL bHWFormats[HWCodec_NB];
283.         DWORD HWAccelResFlags;
284.         DWORD HWDeintMode;
285.         DWORD HWDeintOutput;
286.         BOOL HWDeintHQ;
287.         DWORD DeintFieldOrder;
288.         LAVDeintMode DeintMode;
289.         DWORD SWDeintMode;
290.         DWORD SWDeintOutput;
291.         DWORD DitherMode;
292.         BOOL bDVDVideo;
293.     } m_settings;
294.
295.     CBaseTrayIcon *m_pTrayIcon;
296.
297. #ifdef DEBUG
298.     FloatingAverage<double> m_pixFmtTimingAvg;
299. #endif
300. };

```

可见该类继承了CTransformFilter，其的功能真的是非常丰富的。在这里肯定无法对其进行一一分析，只能选择其中重点的函数进行一下分析。

该类中包含了解码线程类：CDecodeThread m_Decoder;，这里封装了解码功能。

同时该类中包含了函数Receive(IMediaSample *pIn);，是发挥解码功能的函数，其中pIn是输入的解码前的视频压缩编码数据。

下面来看看Receive()函数：

```
[cpp]
1. //处理的核心
2. //核心一般才有IMediaSample
3. HRESULT CLAVVideo::Receive(IMediaSample *pIn)
4. {
5.     CAutoLock cAutoLock(&m_csReceive);
6.     HRESULT hr = S_OK;
7.
8.     AM_SAMPLE2_PROPERTIES const *pProps = m_pInput->SampleProps();
9.     if(pProps->dwStreamId != AM_STREAM_MEDIA) {
10.         return m_pOutput->Deliver(pIn);
11.     }
12.
13.     AM_MEDIA_TYPE *pmt = NULL;
14.     //获取媒体类型等等
15.     if (SUCCEEDED(pIn->GetMediaType(&pmt)) && pmt) {
16.         CMediaType mt = *pmt;
17.         DeleteMediaType(pmt);
18.         if (mt != m_pInput->CurrentMediaType() || !(m_dwDecodeFlags & LAV_VIDEO_DEC_FLAG_DVD)) {
19.             DbgLog((LOG_TRACE, 10, L"::Receive(): Input sample contained media type, dynamic format change..."));
20.             m_Decoder.EndOfStream();
21.             hr = m_pInput->SetMediaType(&mt);
22.             if (FAILED(hr)) {
23.                 DbgLog((LOG_ERROR, 10, L"::Receive(): Setting new media type failed..."));
24.                 return hr;
25.             }
26.         }
27.     }
28.
29.     m_hrDeliver = S_OK;
30.
31.     // Skip over empty packets
32.     if (pIn->GetActualDataLength() == 0) {
33.         return S_OK;
34.     }
35.     //解码
36.     hr = m_Decoder.Decode(pIn);
37.     if (FAILED(hr))
38.         return hr;
39.
40.     if (FAILED(m_hrDeliver))
41.         return m_hrDeliver;
42.
43.     return S_OK;
44. }
```

由代码我们可以看出，实际发挥出解码功能的函数是hr = m_Decoder.Decode(pIn);。

下面我们来看看CDecodeThread类的Decode()方法：

```

1. //解码线程的解码函数
2. STDMETHODIMP CDecodeThread::Decode(IMediaSample *pSample)
3. {
4.     CAutoLock decoderLock(this);
5.
6.     if (!CAMThread::ThreadExists())
7.         return E_UNEXPECTED;
8.
9.     // Wait until the queue is empty
10.    while(HasSample())
11.        Sleep(1);
12.
13.    // Re-init the decoder, if requested
14.    // Doing this inside the worker thread alone causes problems
15.    // when switching from non-sync to sync, so ensure we're in sync.
16.    if (m_bDecoderNeedsReInit) {
17.        CAMThread::CallWorker(CMD_REINIT);
18.        while (!m_evEOSDone.Check()) {
19.            m_evSample.Wait();
20.            ProcessOutput();
21.        }
22.    }
23.
24.    m_evDeliver.Reset();
25.    m_evSample.Reset();
26.    m_evDecodeDone.Reset();
27.
28.    pSample->AddRef();
29.
30.    // Send data to worker thread, and wake it (if it was waiting)
31.    PutSample(pSample);
32.
33.    // If we don't have thread safe buffers, we need to synchronize
34.    // with the worker thread and deliver them when they are available
35.    // and then let it know that we did so
36.    if (m_bSyncToProcess) {
37.        while (!m_evDecodeDone.Check()) {
38.            m_evSample.Wait();
39.            ProcessOutput();
40.        }
41.    }
42.
43.    ProcessOutput();
44.
45.    return S_OK;
46. }

```

这个方法乍一看感觉很抽象，好像没看见直接调用任何解码的函数。如果LAVVideo的封装的ffmpeg的libavcodec的话，应该是最终调用avcodec_decode_video2()才对啊。。。先来看看CDecodeThread这个类的定义吧！

DecodeThread.h

```

1. /* 雷霄骅
2.  * 中国传媒大学/数字电视技术
3.  * leixiaohua1020@126.com
4.  *
5.  */
6. /*
7.  * Copyright (C) 2010-2013 Hendrik Leppkes
8.  * http://www.1f0.de
9.  *
10. * This program is free software; you can redistribute it and/or modify
11. * it under the terms of the GNU General Public License as published by
12. * the Free Software Foundation; either version 2 of the License, or
13. * (at your option) any later version.
14. *
15. * This program is distributed in the hope that it will be useful,
16. * but WITHOUT ANY WARRANTY; without even the implied warranty of
17. * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
18. * GNU General Public License for more details.
19. *
20. * You should have received a copy of the GNU General Public License along
21. * with this program; if not, write to the Free Software Foundation, Inc.,
22. * 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
23. */
24.
25. #pragma once
26.
27. #include "decoders/ILAVDecoder.h"
28. #include "SynchronizedQueue.h"
29.
30. class CLAVVideo;
31.
32. class CDecodeThread : public ILAVVideoCallback, protected CAMThread, protected CCritSec
33. {
34. public:

```

```

35.     CDecodeThread(CLAVVideo *pLAVVideo);
36.     ~CDecodeThread();
37.
38.     // Parts of ILAVDecoder
39.     STDMETHODIMP_(const WCHAR*) GetDecoderName() { return m_pDecoder ? m_pDecoder->GetDecoderName() : NULL; }
40.     STDMETHODIMP_(long) GetBufferCount() { return m_pDecoder ? m_pDecoder->GetBufferCount() : 4; }
41.     STDMETHODIMP_(BOOL) IsInterlaced() { return m_pDecoder ? m_pDecoder->IsInterlaced() : TRUE; }
42.     STDMETHODIMP GetPixelFormat(LAVPixelFormat *pPix, int *pBpp) { ASSERT(m_pDecoder); return m_pDecoder-
>GetPixelFormat(pPix, pBpp); }
43.     STDMETHODIMP_(REFERENCE_TIME) GetFrameDuration() { ASSERT(m_pDecoder); return m_pDecoder->GetFrameDuration(); }
44.     STDMETHODIMP HasThreadSafeBuffers() { return m_pDecoder ? m_pDecoder->HasThreadSafeBuffers() : S_FALSE; }
45.
46.
47.     STDMETHODIMP CreateDecoder(const CMediaType *pmt, AVCodecID codec);
48.     STDMETHODIMP Close();
49.     //解码线程的解码函数
50.     STDMETHODIMP Decode(IMediaSample *pSample);
51.     STDMETHODIMP Flush();
52.     STDMETHODIMP EndOfStream();
53.
54.     STDMETHODIMP InitAllocator(IMemAllocator **ppAlloc);
55.     STDMETHODIMP PostConnect(IPin *pPin);
56.
57.     STDMETHODIMP_(BOOL) IsHWDecoderActive() { return m_bHWDecoder; }
58.
59.     // ILAVVideoCallback
60.     STDMETHODIMP AllocateFrame(LAVFrame **ppFrame);
61.     STDMETHODIMP ReleaseFrame(LAVFrame **ppFrame);
62.     STDMETHODIMP Deliver(LAVFrame *pFrame);
63.     STDMETHODIMP_(LPWSTR) GetFileExtension();
64.     STDMETHODIMP_(BOOL) FilterInGraph(PIN_DIRECTION dir, const GUID &clsid);
65.     STDMETHODIMP_(DWORD) GetDecodeFlags();
66.     STDMETHODIMP_(CMediaType&) GetInputMediaType();
67.     STDMETHODIMP GetLAVPinInfo(LAVPinInfo &info);
68.     STDMETHODIMP_(CBasePin*) GetOutputPin();
69.     STDMETHODIMP_(CMediaType&) GetOutputMediaType();
70.     STDMETHODIMP DVDStripPacket(BYTE* &p, long& len);
71.     STDMETHODIMP_(LAVFrame*) GetFlushFrame();
72.     STDMETHODIMP ReleaseAllDXVAResources();
73.
74. protected:
75.     //包含了对进程的各种操作, 重要
76.     DWORD ThreadProc();
77.
78. private:
79.     STDMETHODIMP CreateDecoderInternal(const CMediaType *pmt, AVCodecID codec);
80.     STDMETHODIMP PostConnectInternal(IPin *pPin);
81.
82.     STDMETHODIMP DecodeInternal(IMediaSample *pSample);
83.     STDMETHODIMP ClearQueues();
84.     STDMETHODIMP ProcessOutput();
85.
86.     bool HasSample();
87.     void PutSample(IMediaSample *pSample);
88.     IMediaSample* GetSample();
89.     void ReleaseSample();
90.
91.     bool CheckForEndOfSequence(IMediaSample *pSample);
92.
93. private:
94.     //各种对进程进行的操作
95.     enum {CMD_CREATE_DECODER, CMD_CLOSE_DECODER, CMD_FLUSH, CMD_EOS, CMD_EXIT, CMD_INIT_ALLOCATOR, CMD_POST_CONNECT, CMD_REINIT};
96.     //注意DecodeThread像是一个处于中间位置的东西
97.     //连接了Filter核心类CLAVVideo和解码器的接口ILAVDecoder
98.     CLAVVideo *m_pLAVVideo;
99.     ILAVDecoder *m_pDecoder;
100.
101.     AVCodecID m_Codec;
102.
103.     BOOL m_bHWDecoder;
104.     BOOL m_bHWDecoderFailed;
105.
106.     BOOL m_bSyncToProcess;
107.     BOOL m_bDecoderNeedsReInit;
108.     CAMEvent m_evInput;
109.     CAMEvent m_evDeliver;
110.     CAMEvent m_evSample;
111.     CAMEvent m_evDecodeDone;
112.     CAMEvent m_evEOSDone;
113.
114.     CCritSec m_ThreadCritSec;
115.     struct {
116.         const CMediaType *pmt;
117.         AVCodecID codec;
118.         IMemAllocator **allocator;
119.         IPin *pin;
120.     } m_ThreadCallContext;
121.     CSynchronizedQueue<LAVFrame *> m_Output;
122.
123.     CCritSec m_SampleCritSec;
124.     IMediaSample *m_NextSample;

```

```

125.
126.     IMediaSample *m_TempSample[2];
127.     IMediaSample *m_FailedSample;
128.
129.     std::wstring m_processName;
130. };

```

从名字上我们可以判断，这个类用于管理解码的线程。在这里我们关注该类里面的两个指针变量：

```

CLAVVideo  *m_pLAVVideo;
ILAVDecoder *m_pDecoder;

```

其中第一个指针变量就是这个工程中最核心的类CLAVVideo，而第二个指针变量则是解码器的接口。通过这个接口就可以调用具体解码器的相应方法了。（注：在源代码中发现，解码器不光包含libavcodec，也可以是wmv9等等，换句话说，是可以扩展其他种类的解码器的。不过就目前的情况来看，lavvideo似乎不如ffdshow支持的解码器种类多）

该类里面还有一个函数：

ThreadProc()

该函数中包含了对线程的各种操作，其中包含调用了ILAVDecoder接口的各种方法：

```

[cpp]
1. //包含了对进程的各种操作
2. DWORD CDecodeThread::ThreadProc()
3. {
4.     HRESULT hr;
5.     DWORD cmd;
6.
7.     BOOL bEOS = FALSE;
8.     BOOL bReinit = FALSE;
9.
10.    SetThreadName(-1, "LAVVideo Decode Thread");
11.
12.    HANDLE hWaitEvents[2] = { GetRequestHandle(), m_evInput };
13.    //不停转圈，永不休止
14.    while(1) {
15.        if (!bEOS && !bReinit) {
16.            // Wait for either an input sample, or an request
17.            WaitForMultipleObjects(2, hWaitEvents, FALSE, INFINITE);
18.        }
19.        //根据操作命令的不同
20.        if (CheckRequest(&cmd)) {
21.            switch (cmd) {
22.                //创建解码器
23.                case CMD_CREATE_DECODER:
24.                {
25.                    CAutoLock lock(&m_ThreadCritSec);
26.                    //创建
27.                    hr = CreateDecoderInternal(m_ThreadCallContext.pmt, m_ThreadCallContext.codec);
28.                    Reply(hr);
29.
30.                    m_ThreadCallContext.pmt = NULL;
31.                }
32.                break;
33.                case CMD_CLOSE_DECODER:
34.                {
35.                    //关闭
36.                    ClearQueues();
37.                    SAFE_DELETE(m_pDecoder);
38.                    Reply(S_OK);
39.                }
40.                break;
41.                case CMD_FLUSH:
42.                {
43.                    //清楚
44.                    ClearQueues();
45.                    m_pDecoder->Flush();
46.                    Reply(S_OK);
47.                }
48.                break;
49.                case CMD_EOS:
50.                {
51.                    bEOS = TRUE;
52.                    m_evEOSDone.Reset();
53.                    Reply(S_OK);
54.                }
55.                break;
56.                case CMD_EXIT:
57.                {
58.                    //退出
59.                    Reply(S_OK);
60.                    return 0;
61.                }
62.                break;
63.                case CMD_INIT_ALLOCATOR:
64.                {

```



```

65.         CAutoLock lock(&m_ThreadCritSec);
66.         hr = m_pDecoder->InitAllocator(m_ThreadCallContext.allocator);
67.         Reply(hr);
68.
69.         m_ThreadCallContext.allocator = NULL;
70.     }
71.     break;
72. case CMD_POST_CONNECT:
73.     {
74.         CAutoLock lock(&m_ThreadCritSec);
75.         hr = PostConnectInternal(m_ThreadCallContext.pin);
76.         Reply(hr);
77.
78.         m_ThreadCallContext.pin = NULL;
79.     }
80.     break;
81. case CMD_REINIT:
82.     {
83.         //重启
84.         CMediaType &mt = m_pLAVVideo->GetInputMediaType();
85.         CreateDecoderInternal(&mt, m_Codec);
86.         m_TempSample[1] = m_NextSample;
87.         m_NextSample = m_FailedSample;
88.         m_FailedSample = NULL;
89.         bReinit = TRUE;
90.         m_evEOSDone.Reset();
91.         Reply(S_OK);
92.         m_bDecoderNeedsReInit = FALSE;
93.     }
94.     break;
95. default:
96.     ASSERT(0);
97. }
98. }
99.
100. if (m_bDecoderNeedsReInit) {
101.     m_evInput.Reset();
102.     continue;
103. }
104.
105. if (bReinit && !m_NextSample) {
106.     if (m_TempSample[0]) {
107.         m_NextSample = m_TempSample[0];
108.         m_TempSample[0] = NULL;
109.     } else if (m_TempSample[1]) {
110.         m_NextSample = m_TempSample[1];
111.         m_TempSample[1] = NULL;
112.     } else {
113.         bReinit = FALSE;
114.         m_evEOSDone.Set();
115.         m_evSample.Set();
116.         continue;
117.     }
118. }
119. //获得一份数据
120. IMediaSample *pSample = GetSample();
121. if (!pSample) {
122.     // Process the EOS now that the sample queue is empty
123.     if (bEOS) {
124.         bEOS = FALSE;
125.         m_pDecoder->EndOfStream();
126.         m_evEOSDone.Set();
127.         m_evSample.Set();
128.     }
129.     continue;
130. }
131. //解码
132. DecodeInternal(pSample);
133.
134. // Release the sample
135. //释放
136. SafeRelease(&pSample);
137.
138. // Indicates we're done decoding this sample
139. m_evDecodeDone.Set();
140.
141. // Set the Sample Event to unblock any waiting threads
142. m_evSample.Set();
143. }
144.
145. return 0;
146. }

```

先分析到这里了，至于ILAVDecoder接口方面的东西下篇文章再写。

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/13022201>

文章标签：LAVFilter 源代码 解码 ffmpeg directshow

个人分类：LAV Filter

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