

Media Player Classic - HC 源代码分析 2：核心类（CMainFrame）（1）

2013年10月28日 18:21:05 阅读数：8412

Media Player Classic - HC 源代码分析系列文章列表：

[Media Player Classic - HC 源代码分析 1：整体结构](#)

[Media Player Classic - HC 源代码分析 2：核心类（CMainFrame）（1）](#)

[Media Player Classic - HC 源代码分析 3：核心类（CMainFrame）（2）](#)

[Media Player Classic - HC 源代码分析 4：核心类（CMainFrame）（3）](#)

[Media Player Classic - HC 源代码分析 5：关于对话框（CAboutDlg）](#)

[Media Player Classic - HC 源代码分析 6：MediaInfo选项卡（CPPageFileMediaInfo）](#)

[Media Player Classic - HC 源代码分析 7：详细信息选项卡（CPPageFileInfoDetails）](#)



上一篇文章概括性的介绍了Media Player Classic - Home Cinema (mpc-hc)播放器的源代码：[Media Player Classic - HC 源代码分析 1：整体结构](#)现在可以开始看看具体的源代码了。

mpc-hc最核心的类名字叫CMainFrame，它的定义位于MainFrm.h文件中

CMainFrame定义非常的长，包含了视频播放器的方方面面，一共900多行，在这里应该快放不下了。因此我删掉了很多代码，只保留了部分代码。关键的函数上面都写上了注释。

```
[cpp]
1. class CMainFrame : public CFrameWnd, public CDropTarget
2. {
3.     ...
4.
5.     // TODO: wrap these graph objects into a class to make it look cleaner
6.     //各种DirectShow接口
7.     //CComPtr被称为智能指针，是ATL提供的一个模版类，能够从语法上自动完成COM的AddRef和Release。
8.     CComPtr<IGraphBuilder2> m_pGB;
9.     CComQIPtr<IMediaControl> m_pMC;
10.    CComQIPtr<IMediaEventEx> m_pME;
11.    CComQIPtr<IVideoWindow> m_pVW;
12.    //这里也可以获得
13.    //分辨率，比特率，帧率
14.    //经过测试，貌似这里取不到值 ==
15.    CComQIPtr<IBasicVideo> m_pBV;
16.    //音量，均衡器等信息
17.    CComQIPtr<IBasicAudio> m_pBA;
18.    CComQIPtr<IMediaSeeking> m_pMS;
19.    CComQIPtr<IVideoFrameStep> m_pFS;
20.    //接收端质量信息：抖动，抖动，视音频同步情况等。。。
21.    CComQIPtr<IQualProp, &IID_IQualProp> m_pQP;
22.    //缓存信息
23.    CComQIPtr<IBufferInfo> m_pBI;
24.    CComQIPtr<IAMOpenProgress> m_pAMOP;
25.    CComPtr<IVMRMixerControl9> m_pVMRMC;
26.    CComPtr<IMFVideoDisplayControl> m_pMFVDC;
27.    CComPtr<IMFVideoProcessor> m_pMFVP;
28.    CComPtr<IVMRWindowlessControl9> m_pVMRWC;
29.    ...
30.    void SetVolumeBoost(UINT nAudioBoost);
31.    void SetBalance(int balance);
32.
33.    // subtitles
34.    CCritSec m_csSubLock;
35.
36.    CList<SubtitleInput> m_pSubStreams;
37.    POSITION m_posFirstExtSub;
```

```

38.     ISubStream* m_pCurrentSubStream;
39.
40.     SubtitleInput* GetSubtitleInput(int& i, bool bIsOffset = false);
41.
42.     friend class CTextPassThruFilter;
43.
44.     // windowing
45.
46.     CRect m_lastWindowRect;
47.     CPoint m_lastMouseMove;
48.
49.     void ShowControls(int nCS, bool fSave = false);
50.     void SetUIPreset(int iCaptionMenuMode, UINT nCS);
51.
52.     void SetDefaultWindowRect(int iMonitor = 0);
53.     void SetDefaultFullscreenState();
54.     void RestoreDefaultWindowRect();
55.     void ZoomVideoWindow(bool snap = true, double scale = ZOOM_DEFAULT_LEVEL);
56.     double GetZoomAutoFitScale(bool bLargerOnly = false) const;
57.
58.     void SetAlwaysOnTop(int iOnTop);
59.
60.     // dynamic menus
61.     // 动态菜单
62.     void SetupOpenCDSubMenu();
63.     void SetupFiltersSubMenu();
64.     void SetupAudioSwitcherSubMenu();
65.     void SetupSubtitlesSubMenu();
66.     ...
67.
68.     CMenu m_popupmain, m_popup;
69.     CMenu m_opencds;
70.     CMenu m_filters, m_subtitles, m_audios;
71.     CMenu m_language;
72.     ...
73.
74.     // chapters (file mode)
75.     CComPtr<IDSMChapterBag> m_pCB;
76.     void SetupChapters();
77.
78.     // chapters (dvd mode)
79.     void SetupDVDChapters();
80.
81.     void SetupIViAudReg();
82.
83.     void AddTextPassThruFilter();
84.
85.     int m_nLoops;
86.     UINT m_nLastSkipDirection;
87.
88.     bool m_fCustomGraph;
89.     ...
90.
91. public:
92.     void StartWebServer(int nPort);
93.     void StopWebServer();
94.
95.     CString GetStatusMessage() const;
96.     int GetPlaybackMode() const { return m_iPlaybackMode; }
97.     void SetPlaybackMode(int iNewStatus);
98.     bool IsMuted() { return m_wndToolBar.GetVolume() == -10000; }
99.     int GetVolume() { return m_wndToolBar.m_volctrl.GetPos(); }
100.
101. public:
102.     CMainFrame();
103.     DECLARE_DYNAMIC(CMainFrame)
104.
105.     // Attributes
106. public:
107.     bool m_fFullscreen;
108.     bool m_fFirstFSAfterLaunchOnFS;
109.     bool m_fStartInD3DFullscreen;
110.     bool m_fHideCursor;
111.     CMenu m_navaudio, m_navsubtitle;
112.
113.     CComPtr<IBaseFilter> m_pRefClock; // Adjustable reference clock. GothSync
114.     CComPtr<ISyncClock> m_pSyncClock;
115.     ...
116.
117.     CControlBar* m_pLastBar;
118.
119. protected:
120.     MPC_LOADSTATE m_iMediaLoadState;
121.     bool m_bFirstPlay;
122.
123.     bool m_fAudioOnly;
124.     dispmode m_dmBeforeFullscreen;
125.     CString m_LastOpenFile, m_LastOpenBDPath;
126.     HMONITOR m_LastWindow_HM;
127.
128.     DVD_DOMAIN m_idVDDomain;

```

```

129.     DWORD m_idVDTitle;
130.     double m_dSpeedRate;
131.     double m_ZoomX, m_ZoomY, m_PosX, m_PosY;
132.     int m_AngleX, m_AngleY, m_AngleZ;
133.
134.     //操作 Operations
135.     //打开一个媒体
136.     bool OpenMediaPrivate(CAutoPtr<OpenMediaData> pOMD);
137.     //关闭媒体
138.     void CloseMediaPrivate();
139.     void DoTunerScan(TunerScanData* pTSD);
140.
141.     CWnd* GetModalParent();
142.
143.     void OpenCreateGraphObject(OpenMediaData* pOMD);
144.     //打开文件
145.     void OpenFile(OpenFileData* pOFD);
146.     //打开DVD
147.     void OpenDVD(OpenDVDData* pODD);
148.     //打开摄像头
149.     void OpenCapture(OpenDeviceData* pODD);
150.     HRESULT OpenBDAGraph();
151.     void OpenCustomizeGraph();
152.     //设置视频窗口
153.     void OpenSetupVideo();
154.     //设置音量
155.     void OpenSetupAudio();
156.     void OpenSetupInfoBar();
157.     void UpdateChapterInInfoBar();
158.     //打开统计工具条
159.     void OpenSetupStatsBar();
160.     //打开状态工具条
161.     void OpenSetupStatusBar();
162.     // void OpenSetupToolBar();
163.     void OpenSetupCaptureBar();
164.     //设置窗口标题
165.     void OpenSetupWindowTitle(CString fn = _T(""));
166.     void AutoChangeMonitorMode();
167.
168.     bool GraphEventComplete();
169.
170.     friend class CGraphThread;
171.     CGraphThread* m_pGraphThread;
172.     bool m_bOpenedThruThread;
173.
174.     CAtlArray<REFERENCE_TIME> m_kfs;
175.
176.     bool m_fOpeningAborted;
177.     bool m_bWasSnapped;
178.
179. public:
180.     void OpenCurPlaylistItem(REFERENCE_TIME rtStart = 0);
181.     void OpenMedia(CAutoPtr<OpenMediaData> pOMD);
182.     void PlayFavoriteFile(CString fav);
183.     void PlayFavoriteDVD(CString fav);
184.     bool ResetDevice();
185.     bool DisplayChange();
186.     void CloseMedia();
187.     void StartTunerScan(CAutoPtr<TunerScanData> pTSD);
188.     void StopTunerScan();
189.     HRESULT SetChannel(int nChannel);
190.
191.     void AddCurDevToPlaylist();
192.
193.     bool m_fTrayIcon;
194.     //设置系统托盘图标
195.     void ShowTrayIcon(bool fShow);
196.     void SetTrayTip(CString str);
197.
198.     CSize GetVideoSize() const;
199.     void ToggleFullscreen(bool fToNearest, bool fSwitchScreenResWhenHasTo);
200.     void Toggled3DFullscreen(bool fSwitchScreenResWhenHasTo);
201.     void MoveVideoWindow(bool fShowStats = false);
202.     void RepaintVideo();
203.     void HideVideoWindow(bool fHide);
204.
205.     OAFilterState GetMediaState() const;
206.     REFERENCE_TIME GetPos() const;
207.     REFERENCE_TIME GetDur() const;
208.     void SeekTo(REFERENCE_TIME rt, bool fSeekToKeyFrame = false);
209.     //设置播放速率
210.     void SetPlayingRate(double rate);
211.
212.     DWORD SetupAudioStreams();
213.     DWORD SetupSubtitleStreams();
214.     //字幕
215.     bool LoadSubtitle(CString fn, ISubStream** actualStream = nullptr, bool bAutoLoad = false);
216.     bool SetSubtitle(int i, bool bIsOffset = false, bool bDisplayMessage = false, bool bApplyDefStyle = false);
217.     void SetSubtitle(ISubStream* pSubStream, bool bApplyDefStyle = false);
218.     void ToggleSubtitleOnOff(bool bDisplayMessage = false);
219.     void ReplaceSubtitle(const ISubStream* pSubStreamOld, ISubStream* pSubStreamNew);

```

```

220. void InvalidateSubtitle(DWORD_PTR nSubtitleId = -1, REFERENCE_TIME rtInvalidate = -1);
221. void ReloadSubtitle();
222. HRESULT InsertTextPassThruFilter(IBaseFilter* pBF, IPin* pPin, IPin* pPinto);
223.
224. void SetAudioTrackIdx(int index);
225. void SetSubtitleTrackIdx(int index);
226.
227. void AddFavorite(bool fDisplayMessage = false, bool fShowDialog = true);
228.
229. // shaders
230. CAtlList<CString> m_shaderLabels;
231. CAtlList<CString> m_shaderLabelsScreenSpace;
232. void SetShaders();
233. void UpdateShaders(CString label);
234.
235. // capturing
236. bool m_fCapturing;
237. HRESULT BuildCapture(IPin* pPin, IBaseFilter* pBF[3], const GUID& majortype, AM_MEDIA_TYPE* pmt); // pBF: 0 buff, 1 enc, 2 mux, p
mt is for 1 enc
238. bool BuildToCapturePreviewPin(IBaseFilter* pVidCap, IPin** pVidCapPin, IPin** pVidPrevPin,
239.                               IBaseFilter* pAudCap, IPin** pAudCapPin, IPin** pAudPrevPin);
240. bool BuildGraphVideoAudio(int fVPreview, bool fVCapture, int fAPreview, bool fACapture);
241. bool DoCapture(), StartCapture(), StopCapture();
242.
243. bool DoAfterPlaybackEvent();
244. void ParseDirs(CAtlList<CString>& sl);
245. bool SearchInDir(bool bDirForward, bool bLoop = false);
246.
247. virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
248. virtual BOOL PreTranslateMessage(MSG* pMsg);
249. virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo);
250. virtual void RecalcLayout(BOOL bNotify = TRUE);
251.
252. // DVB capture
253. void ShowCurrentChannelInfo(bool fShowOSD = true, bool fShowInfoBar = false);
254.
255. // Implementation
256. public:
257.     virtual ~CMainFrame();
258. #ifdef _DEBUG
259.     virtual void AssertValid() const;
260.     virtual void Dump(CDumpContext& dc) const;
261. #endif
262.
263. protected:
264.     // control bar embedded members
265.     CChildView m_wndView;
266.
267.     UINT m_nCS;
268.     CPlayerSeekBar m_wndSeekBar;
269.     CPlayerToolBar m_wndToolBar;
270.     CPlayerInfoBar m_wndInfoBar;
271.     CPlayerInfoBar m_wndStatsBar;
272.     CPlayerStatusBar m_wndStatusBar;
273.     CList<CControlBar*> mBars;
274.
275.     CPlayerSubresyncBar m_wndSubresyncBar;
276.     CPlayerPlaylistBar m_wndPlaylistBar;
277.     CPlayerCaptureBar m_wndCaptureBar;
278.     CPlayerNavigationBar m_wndNavigationBar;
279.     CPlayerShaderEditorBar m_wndShaderEditorBar;
280.     CEditListEditor m_wndEditListEditor;
281.     CList<CSizingControlBar*> m_dockingbars;
282.     ...
283.
284.     // Generated message map functions
285.
286.     DECLARE_MESSAGE_MAP()
287.
288. public:
289.     //打开的时候加载
290.     afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
291.     //关闭的时候加载
292.     afx_msg void OnDestroy();
293.
294.     afx_msg LRESULT OnTaskBarRestart(WPARAM, LPARAM);
295.     afx_msg LRESULT OnNotifyIcon(WPARAM, LPARAM);
296.     afx_msg LRESULT OnTaskBarThumbnailsCreate(WPARAM, LPARAM);
297.
298.     afx_msg LRESULT OnSkypeAttach(WPARAM wParam, LPARAM lParam);
299.
300.     afx_msg void OnSetFocus(CWnd* pOldWnd);
301.     afx_msg void OnGetMinMaxInfo(MINMAXINFO* lpMMI);
302.     afx_msg void OnMove(int x, int y);
303.     afx_msg void OnMoving(UINT fwSide, LPRECT pRect);
304.     afx_msg void OnSize(UINT nType, int cx, int cy);
305.     afx_msg void OnSizing(UINT fwSide, LPRECT pRect);
306.     afx_msg void OnDisplayChange();
307.
308.     afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
309.     afx_msg void OnActivateApp(BOOL bActive, DWORD dwThreadId);
310.     afx_msg LRESULT OnTaskBarThumbnailClick(WPARAM wParam, LPARAM lParam);

```

```

310.     atx_msg LRESULT UnAppCommand(WPARAM wParam, LPARAM lParam);
311.     afx_msg void OnRawInput(UINT nInputcode, HRAWINPUT hRawInput);
312.
313.     afx_msg LRESULT OnHotKey(WPARAM wParam, LPARAM lParam);
314.
315.     afx_msg void OnTimer(UINT_PTR nIDEvent);
316.
317.     afx_msg LRESULT OnGraphNotify(WPARAM wParam, LPARAM lParam);
318.     afx_msg LRESULT OnResetDevice(WPARAM wParam, LPARAM lParam);
319.     afx_msg LRESULT OnRepaintRenderLess(WPARAM wParam, LPARAM lParam);
320.     afx_msg LRESULT OnResumeFromState(WPARAM wParam, LPARAM lParam);
321.     ...
322.
323.
324.     // menu item handlers
325.
326.     afx_msg void OnFileOpenQuick();
327.     afx_msg void OnFileOpenmedia();
328.     afx_msg void OnUpdateFileOpen(CCmdUI* pCmdUI);
329.     afx_msg BOOL OnCopyData(CWnd* pWnd, COPYDATASTRUCT* pCopyDataStruct);
330.     afx_msg void OnFileOpendvd();
331.     afx_msg void OnFileOpendevide();
332.     afx_msg void OnFileOpenCD(UINT nID);
333.     afx_msg void OnFileReopen();
334.     afx_msg void OnFileRecycle();
335.     afx_msg void OnDropFiles(HDROP hDropInfo); // no menu item
336.     ...
337.     afx_msg void OnHelpHomepage();
338.     afx_msg void OnHelpCheckForUpdate();
339.     afx_msg void OnHelpToolbarImages();
340.     afx_msg void OnHelpDonate();
341.     //关闭的时候加载
342.     afx_msg void OnClose();
343.
344.     afx_msg void OnLanguage(UINT nID);
345.     afx_msg void OnUpdateLanguage(CCmdUI* pCmdUI);
346.
347.     CMPC_Lcd m_Lcd;
348.
349.     // ==== Added by CASIMIR666
350.     CWnd* m_pVideoWnd; // Current Video (main display screen or 2nd)
351.     SIZE m_fullWndSize;
352.     CFullscreenWnd* m_pFullscreenWnd;
353.     CVMROSD m_OSD;
354.     bool m_bRemainingTime;
355.     int m_nCurSubtitle;
356.     long m_lSubtitleShift;
357.     REFERENCE_TIME m_rtCurSubPos;
358.     CString m_strTitle;
359.     bool m_bToggleShader;
360.     bool m_bToggleShaderScreenSpace;
361.     bool m_bInOptions;
362.     bool m_bStopTunerScan;
363.     bool m_bLockedZoomVideoWindow;
364.     int m_nLockedZoomVideoWindow;
365.     bool m_fSetChannelActive;
366.
367.     void SetLoadState(MPC_LOADSTATE iState);
368.     void SetPlayState(MPC_PLAYSTATE iState);
369.     bool CreateFullscreenWindow();
370.     void SetupEVRColorControl();
371.     void SetupVMR9ColorControl();
372.     void SetColorControl(DWORD flags, int& brightness, int& contrast, int& hue, int& saturation);
373.     void SetClosedCaptions(bool enable);
374.     LPCTSTR GetDVDAudioFormatName(const DVD_AudioAttributes& ATR) const;
375.     void SetAudioDelay(REFERENCE_TIME rtShift);
376.     void SetSubtitleDelay(int delay_ms);
377.     //void AutoSelectTracks();
378.     bool IsRealEngineCompatible(CString strFilename) const;
379.     void SetTimersPlay();
380.     void KillTimersStop();
381.
382.
383.     // MPC API functions
384.     void ProcessAPICommand(COPYDATASTRUCT* pCDS);
385.     void SendAPICommand(MPCAPI_COMMAND nCommand, LPCWSTR fmt, ...);
386.     void SendNowPlayingToApi();
387.     void SendSubtitleTracksToApi();
388.     void SendAudioTracksToApi();
389.     void SendPlaylistToApi();
390.     ...
391.
392. protected:
393.     // GDI+
394.     ULONG_PTR m_gdiplusToken;
395.     virtual LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam);
396.     void WTSRegisterSessionNotification();
397.     void WTSUnRegisterSessionNotification();
398.
399.     DWORD m_nMenuHideTick;
400.     UINT m_nSeekDirection;
401. public:

```

```

402.     afx_msg UINT OnPowerBroadcast(UINT nPowerEvent, UINT nEventData);
403.     afx_msg void OnSessionChange(UINT nSessionState, UINT nId);
404.
405.     void EnableShaders1(bool enable);
406.     void EnableShaders2(bool enable);
407.
408.     CAtlList<CHdMvClipInfo::PlaylistItem> m_MPLSPlaylist;
409.     bool m_bIsBDPlay;
410.     bool OpenBD(CString Path);
411. };

```

面对一个如此巨大的类，可能会让人感觉到无从下手。我开始研究的时候也不知道该从何学起（实际上找到CMainFrame这个类就花了我挺长时间的，开始的时候根本没找到哪个类才是mpc-hc的最核心的类）。经过一段时间的探索，我发现了打开一个媒体的函数OpenMedia()，这个函数应该是我们每次使用mpc-hc都一定会调用的函数。从这个函数开始学习源代码还是比较合适的。

在看OpenMedia()代码之前，先来看看有哪些函数调用它了。我们可以借助VC2010的“查看调用层次结构”功能来完成这个任务。发现有3个函数：

```

1. OnFileOpenDevice()//打开一个设备（比如说摄像头）
2. OnFileOpenDVD()//打开一个DVD
3. OpenCurPlaylistItem()//打开播放列表的一条记录（比如说一个文件）

```

这3个函数正好对应着mpc-hc的3个功能：打开设备（摄像头），打开DVD，打开文件。这3个函数在这里就不多讲了，以后有机会再进行分析。

下面我们来看看OpenMedia()函数：

```

1. //打开媒体 (非Private)
2. void CMainFrame::OpenMedia(CAutoPtr<OpenMediaData> pOMD)
3. {
4.     // shortcut
5.     if (OpenDeviceData* p = dynamic_cast<OpenDeviceData*>(pOMD.m_p)) {
6.         if (m_iMediaLoadState == MLS_LOADED && m_pAMTuner
7.             && m_VidDispName == p->DisplayName[0] && m_AudDispName == p->DisplayName[1]) {
8.             m_wndCaptureBar.m_capdlg.SetVideoInput(p->vinput);
9.             m_wndCaptureBar.m_capdlg.SetVideoChannel(p->vchannel);
10.            m_wndCaptureBar.m_capdlg.SetAudioInput(p->ainput);
11.            SendNowPlayingToSkype();
12.            return;
13.        }
14.    }
15.
16.    if (m_iMediaLoadState != MLS_CLOSED) {
17.        CloseMedia();
18.    }
19.
20.    //m_iMediaLoadState = MLS_LOADING; // HACK: hides the logo
21.
22.    const CAppSettings& s = AfxGetAppSettings();
23.
24.    bool fUseThread = m_pGraphThread && s.fEnableWorkerThreadForOpening;
25.
26.    if (OpenFileData* p = dynamic_cast<OpenFileData*>(pOMD.m_p)) {
27.        if (!p->fns.IsEmpty()) {
28.            engine_t e = s.m_Formats.GetEngine(p->fns.GetHead());
29.            if (e != DirectShow /*&& e != RealMedia && e != QuickTime*/) {
30.                fUseThread = false;
31.            }
32.        }
33.    } else if (OpenDeviceData* p = dynamic_cast<OpenDeviceData*>(pOMD.m_p)) {
34.        fUseThread = false;
35.    }
36.
37.    // Create D3DFullscreen window if launched in fullscreen
38.    if (s.IsD3DFullscreen() && m_fStartInD3DFullscreen) {
39.        if (s.AutoChangeFullscrRes.bEnabled) {
40.            AutoChangeMonitorMode();
41.        }
42.        CreateFullScreenWindow();
43.        m_pVideoWnd = m_pFullscreenWnd;
44.        m_fStartInD3DFullscreen = false;
45.    } else {
46.        m_pVideoWnd = &m_wndView;
47.    }
48.
49.    if (fUseThread) {
50.        m_pGraphThread->PostThreadMessage(CGThread::TM_OPEN, 0, (LPARAM)pOMD.Detach());
51.        m_bOpenedThruThread = true;
52.    } else {
53.        //打开媒体 (private)
54.        OpenMediaPrivate(pOMD);
55.        m_bOpenedThruThread = false;
56.    }
57. }

```

这里需要注意，OpenMedia()调用了函数OpenMediaPrivate()。文件的打开功能实际上是在OpenMediaPrivate()中完成的。

下面我们来看看OpenMediaPrivate()的代码，发现比OpenMedia()要复杂很多。

```
[cpp]
1. //打开一个媒体 (private)
2. bool CMainFrame::OpenMediaPrivate(CAutoPtr<OpenMediaData> pOMD)
3. {
4.     //获得设置信息
5.     CAppSettings& s = AfxGetAppSettings();
6.
7.     if (m_iMediaLoadState != MLS_CLOSED) {
8.         ASSERT(0);
9.         return false;
10.    }
11.    //OpenFileData
12.    //OpenDVDDData
13.    //OpenDeviceData
14.    //里面包含了文件或者DVD信息 (名称等)
15.    OpenFileData* pFileData = dynamic_cast<OpenFileData*>(pOMD.m_p);
16.    OpenDVDDData* pDVDDData = dynamic_cast<OpenDVDDData*>(pOMD.m_p);
17.    OpenDeviceData* pDeviceData = dynamic_cast<OpenDeviceData*>(pOMD.m_p);
18.    if (!pFileData && !pDVDDData && !pDeviceData) {
19.        ASSERT(0);
20.        return false;
21.    }
22.
23.    // Clear DXVA state ...
24.    ClearDXVAState();
25.
26.    #ifdef _DEBUG
27.        // Debug trace code - Begin
28.        // Check for bad / buggy auto loading file code
29.        if (pFileData) {
30.            POSITION pos = pFileData->fns.GetHeadPosition();
31.            UINT index = 0;
32.            while (pos != nullptr) {
33.                CString path = pFileData->fns.GetNext(pos);
34.                TRACE(_T("--> CMainFrame::OpenMediaPrivate - pFileData->fns[%d]:\n"), index);
35.                TRACE(_T("\t%s\n"), path.GetString()); // %ws - wide character string always
36.                index++;
37.            }
38.        }
39.        // Debug trace code - End
40.    #endif
41.
42.    CString mi_fn = _T("");
43.
44.    if (pFileData) {
45.        if (pFileData->fns.IsEmpty()) {
46.            return false;
47.        }
48.
49.        CString fn = pFileData->fns.GetHead();
50.
51.        int i = fn.Find(_T(":\\"));
52.        if (i > 0) {
53.            CString drive = fn.Left(i + 2);
54.            UINT type = GetDriveType(drive);
55.            CAtlList<CString> sl;
56.            if (type == DRIVE_REMOVABLE || type == DRIVE_CDROM && GetCDROMType(drive[0], sl) != CDROM_Audio) {
57.                int ret = IDRETRY;
58.                while (ret == IDRETRY) {
59.                    WIN32_FIND_DATA findFileData;
60.                    HANDLE h = FindFirstFile(fn, &findFileData);
61.                    if (h != INVALID_HANDLE_VALUE) {
62.                        FindClose(h);
63.                        ret = IDOK;
64.                    } else {
65.                        CString msg;
66.                        msg.Format(IDS_MAINFRM_114, fn);
67.                        ret = AfxMessageBox(msg, MB_RETRYCANCEL);
68.                    }
69.                }
70.
71.                if (ret != IDOK) {
72.                    return false;
73.                }
74.            }
75.            mi_fn = fn;
76.        }
77.    }
78.
79.    SetLoadState(MLS_LOADING);
80.
81.    // FIXME: Don't show "Closed" initially
82.    PostMessage(WM_KICKIDLE);
83.
84.    CString err;
```

```

85.
86.     m_fUpdateInfoBar = false;
87.     BeginWaitCursor();
88.
89.     try {
90.         CComPtr<IVMRMixerBitmap9> pVMB;
91.         CComPtr<IMFVideoMixerBitmap> pMFVMB;
92.         CComPtr<IMadVRTextOsd> pMVTO;
93.         if (m_fOpeningAborted) {
94.             throw (UINT)IDS_AG_ABORTED;
95.         }
96.
97.         OpenCreateGraphObject(pOMD);
98.
99.         if (m_fOpeningAborted) {
100.             throw (UINT)IDS_AG_ABORTED;
101.         }
102.
103.         SetupIViAudReg();
104.
105.         if (m_fOpeningAborted) {
106.             throw (UINT)IDS_AG_ABORTED;
107.         }
108.         //按类型的不同打开不同的文件
109.         if (pFileData) {
110.             //文件
111.             OpenFile(pFileData);
112.         } else if (pDVDDData) {
113.             //DVD
114.             OpenDVD(pDVDDData);
115.         } else if (pDeviceData) {
116.             if (s.iDefaultCaptureDevice == 1) {
117.                 HRESULT hr = OpenBDAGraph();
118.                 if (FAILED(hr)) {
119.                     throw (UINT)IDS_CAPTURE_ERROR_DEVICE;
120.                 }
121.             } else {
122.                 OpenCapture(pDeviceData);
123.             }
124.         } else {
125.             throw (UINT)IDS_INVALID_PARAMS_ERROR;
126.         }
127.
128.         m_pCAP2 = nullptr;
129.         m_pCAP = nullptr;
130.         //查找接口
131.         m_pGB->FindInterface(__uuidof(ISubPicAllocatorPresenter), (void**)&m_pCAP, TRUE);
132.         m_pGB->FindInterface(__uuidof(ISubPicAllocatorPresenter2), (void**)&m_pCAP2, TRUE);
133.         m_pGB->FindInterface(__uuidof(IVMRWindowlessControl9), (void**)&m_pVMRWC, FALSE); // might have IVMRMixerBitmap9, but not IVMRWindowlessControl9
134.         m_pGB->FindInterface(__uuidof(IVMRMixerControl9), (void**)&m_pVMRMC, TRUE);
135.         m_pGB->FindInterface(__uuidof(IVMRMixerBitmap9), (void**)&pVMB, TRUE);
136.         m_pGB->FindInterface(__uuidof(IMFVideoMixerBitmap), (void**)&pMFVMB, TRUE);
137.         pMVTO = m_pCAP;
138.
139.         if (s.fShowOSD || s.fShowDebugInfo) { // Force OSD on when the debug switch is used
140.             if (pVMB) {
141.                 m_OSD.Start(m_pVideoWnd, pVMB, IsD3DFullScreenMode());
142.             } else if (pMFVMB) {
143.                 m_OSD.Start(m_pVideoWnd, pMFVMB, IsD3DFullScreenMode());
144.             } else if (pMVTO) {
145.                 m_OSD.Start(m_pVideoWnd, pMVTO);
146.             }
147.         }
148.         //VMR9
149.         SetupVMR9ColorControl();
150.
151.         // === EVR !
152.         m_pGB->FindInterface(__uuidof(IMFVideoDisplayControl), (void**)&m_pMFVDC, TRUE);
153.         m_pGB->FindInterface(__uuidof(IMFVideoProcessor), (void**)&m_pMFVP, TRUE);
154.         if (m_pMFVDC) {
155.             m_pMFVDC->SetVideoWindow(m_pVideoWnd->m_hWnd);
156.         }
157.
158.         //SetupEVRColorControl();
159.         //does not work at this location
160.         //need to choose the correct mode (IMFVideoProcessor::SetVideoProcessorMode)
161.
162.         BeginEnumFilters(m_pGB, pEF, pBF) {
163.             if (m_pLN21 == pBF) {
164.                 m_pLN21->SetServiceState(s.fClosedCaptions ? AM_L21_CCSTATE_On : AM_L21_CCSTATE_Off);
165.                 break;
166.             }
167.         }
168.         EndEnumFilters;
169.
170.         if (m_fOpeningAborted) {
171.             throw (UINT)IDS_AG_ABORTED;
172.         }
173.         //打开自定义的Graph

```



```

174.     OpenCustomizeGraph();
175.
176.     if (m_fOpeningAborted) {
177.         throw (UINT)IDS_AG_ABORTED;
178.     }
179.     //设置视频窗口
180.     OpenSetupVideo();
181.
182.     if (m_fOpeningAborted) {
183.         throw (UINT)IDS_AG_ABORTED;
184.     }
185.     //设置音量
186.     OpenSetupAudio();
187.
188.     if (m_fOpeningAborted) {
189.         throw (UINT)IDS_AG_ABORTED;
190.     }
191.
192.     if (m_pCAP && (!m_fAudioOnly || m_fRealMediaGraph)) {
193.
194.         if (s.fDisableInternalSubtitles) {
195.             m_pSubStreams.RemoveAll(); // Needs to be replaced with code that checks for forced subtitles.
196.         }
197.
198.         m_posFirstExtSub = nullptr;
199.         POSITION pos = pOMD->subs.GetHeadPosition();
200.         while (pos) {
201.             LoadSubtitle(pOMD->subs.GetNext(pos), nullptr, true);
202.         }
203.     }
204.
205.     if (m_fOpeningAborted) {
206.         throw (UINT)IDS_AG_ABORTED;
207.     }
208.     //设置视频窗口标题
209.     OpenSetupWindowTitle(pOMD->title);
210.
211.     if (s.fEnableEDLEditor) {
212.         m_wndEditListEditor.OpenFile(pOMD->title);
213.     }
214.
215.     if (::GetCurrentThreadId() == AfxGetApp()->m_nThreadID) {
216.         OnFilePostOpenMedia();
217.     } else {
218.         PostMessage(WM_COMMAND, ID_FILE_POST_OPENMEDIA);
219.     }
220.
221.     while (m_iMediaLoadState != MLS_LOADED
222.           && m_iMediaLoadState != MLS_CLOSING // FIXME
223.           ) {
224.         Sleep(50);
225.     }
226.     //设置音频流
227.     DWORD audstm = SetupAudioStreams();
228.     //设置字幕流
229.     DWORD substm = SetupSubtitleStreams();
230.
231.     if (audstm) {
232.         OnPlayAudio(ID_AUDIO_SUBITEM_START + audstm);
233.     }
234.     if (substm) {
235.         SetSubtitle(substm - 1);
236.     }
237.
238.     // PostMessage instead of SendMessage because the user might call CloseMedia and then we would deadlock
239.
240.     PostMessage(WM_COMMAND, ID_PLAY_PAUSE);
241.
242.     m_bFirstPlay = true;
243.
244.     if (!(s.nCLSwitches & CLSW_OPEN) && (s.nLoops > 0)) {
245.         PostMessage(WM_COMMAND, ID_PLAY_PLAY);
246.     } else {
247.         // If we don't start playing immediately, we need to initialize
248.         // the seekbar and the time counter.
249.         OnTimer(TIMER_STREAMPOSPOLLER);
250.         OnTimer(TIMER_STREAMPOSPOLLER2);
251.     }
252.
253.     s.nCLSwitches &= ~CLSW_OPEN;
254.
255.     if (pFileData) {
256.         if (pFileData->rtStart > 0) {
257.             PostMessage(WM_RESUMEFROMSTATE, (WPARAM)PM_FILE, (LPARAM)(pFileData-
258. >rtStart / 10000)); // REFERENCE_TIME doesn't fit in LPARAM under a 32bit env.
259.         }
260.     } else if (pDVDDData) {
261.         if (pDVDDData->pDvdState) {
262.             PostMessage(WM_RESUMEFROMSTATE, (WPARAM)PM_DVD, (LPARAM)(CComPtr<IDvdState>(pDVDDData->pDvdState).Detach())); // m
ust be released by the called message handler
262.         }

```

```

263.         } else if (pDeviceData) {
264.             m_wndCaptureBar.m_capdlg.SetVideoInput(pDeviceData->vinput);
265.             m_wndCaptureBar.m_capdlg.SetVideoChannel(pDeviceData->vchannel);
266.             m_wndCaptureBar.m_capdlg.SetAudioInput(pDeviceData->ainput);
267.         }
268.     } catch (LPCTSTR msg) {
269.         err = msg;
270.     } catch (CString& msg) {
271.         err = msg;
272.     } catch (UINT msg) {
273.         err.LoadString(msg);
274.     }
275.
276. EndWaitCursor();
277.
278. if (!err.IsEmpty()) {
279.     //关闭
280.     CloseMediaPrivate();
281.     m_closingmsg = err;
282.
283.     if (err != ResStr(IDS_AG_ABORTED)) {
284.         if (pFileData) {
285.             m_wndPlaylistBar.SetCurValid(false);
286.
287.             if (m_wndPlaylistBar.IsAtEnd()) {
288.                 m_nLoops++;
289.             }
290.
291.             if (s.fLoopForever || m_nLoops < s.nLoops) {
292.                 bool hasValidFile = false;
293.
294.                 if (m_nLastSkipDirection == ID_NAVIGATE_SKIPBACK) {
295.                     hasValidFile = m_wndPlaylistBar.SetPrev();
296.                 } else {
297.                     hasValidFile = m_wndPlaylistBar.SetNext();
298.                 }
299.
300.                 if (hasValidFile) {
301.                     OpenCurPlaylistItem();
302.                 }
303.                 } else if (m_wndPlaylistBar.GetCount() > 1) {
304.                     DoAfterPlaybackEvent();
305.                 }
306.             } else {
307.                 OnNavigateSkip(ID_NAVIGATE_SKIPFORWARD);
308.             }
309.         }
310.     } else {
311.         m_wndPlaylistBar.SetCurValid(true);
312.
313.         // Apply command line audio shift
314.         if (s.rtShift != 0) {
315.             SetAudioDelay(s.rtShift);
316.             s.rtShift = 0;
317.         }
318.     }
319.
320.     m_nLastSkipDirection = 0;
321.
322.     if (s.AutoChangeFullscrRes.bEnabled && (m_fFullScreen || IsD3DFullScreenMode())) {
323.         AutoChangeMonitorMode();
324.     }
325.     if (m_fFullScreen && s.fRememberZoomLevel) {
326.         m_fFirstFSAfterLaunchOnFS = true;
327.     }
328.
329.     m_LastOpenFile = pOMD->title;
330.
331.     PostMessage(WM_KICKIDLE); // calls main thread to update things
332.
333.     if (!m_bIsBDPlay) {
334.         m_MPLSPlaylist.RemoveAll();
335.         m_LastOpenBDPath = _T("");
336.     }
337.     m_bIsBDPlay = false;
338.
339.     return err.IsEmpty();
340. }

```

这里需要注意，根据打开方式的不同，OpenMediaPrivate()调用了不同的函数。

如果输入的类型为文件，则调用OpenFile()

如果输入的类型为DVD，则调用OpenDVD()

如果输入的类型为设备（例如摄像头），则调用OpenCapture()

在这里，我们假设输入的类型为文件（实际上这也是最普遍的情况）。

看看OpenFile()的源代码。

```
[cpp]
1. //打开文件
2. void CMainFrame::OpenFile(OpenFileData* pOFD)
3. {
4.     if (pOFD->fns.IsEmpty()) {
5.         throw (UINT)IDS_MAINFRM_81;
6.     }
7.     //获取设置
8.     CAppSettings& s = AfxGetAppSettings();
9.
10.    bool bMainFile = true;
11.
12.    POSITION pos = pOFD->fns.GetHeadPosition();
13.    while (pos) {
14.        CString fn = pOFD->fns.GetNext(pos);
15.
16.        fn.Trim();
17.        if (fn.IsEmpty() && !bMainFile) {
18.            break;
19.        }
20.        //使用DirectShow播放文件
21.        HRESULT hr = m_pGB->RenderFile(CStringW(fn), nullptr);
22.
23.        if (bMainFile) {
24.            // Don't try to save file position if source isn't seekable
25.            REFERENCE_TIME rtDur = 0;
26.            m_pMS->GetDuration(&rtDur);
27.
28.            m_bRememberFilePos = s.fKeepHistory && s.fRememberFilePos && rtDur > 0;
29.
30.            if (m_bRememberFilePos && !s.filePositions.AddEntry(fn)) {
31.                REFERENCE_TIME rtPos = s.filePositions.GetLatestEntry()->llPosition;
32.                if (m_pMS) {
33.                    m_pMS->SetPositions(&rtPos, AM_SEEKING_AbsolutePositioning, nullptr, AM_SEEKING_NoPositioning);
34.                }
35.            }
36.        }
37.        QueryPerformanceCounter(&m_liLastSaveTime);
38.
39.        if (FAILED(hr)) {
40.            if (bMainFile) {
41.                if (s.fReportFailedPins) {
42.                    CComQIPtr<IGraphBuilderDeadEnd> pGBDE = m_pGB;
43.                    if (pGBDE && pGBDE->GetCount()) {
44.                        CMediaTypesDlg(pGBDE, GetModalParent()).DoModal();
45.                    }
46.                }
47.
48.                UINT err;
49.
50.                switch (hr) {
51.                    case E_ABORT:
52.                    case RFS_E_ABORT:
53.                        err = IDS_MAINFRM_82;
54.                        break;
55.                    case E_FAIL:
56.                    case E_POINTER:
57.                    default:
58.                        err = IDS_MAINFRM_83;
59.                        break;
60.                    case E_INVALIDARG:
61.                        err = IDS_MAINFRM_84;
62.                        break;
63.                    case E_OUTOFMEMORY:
64.                        err = IDS_AG_OUT_OF_MEMORY;
65.                        break;
66.                    case VFW_E_CANNOT_CONNECT:
67.                        err = IDS_MAINFRM_86;
68.                        break;
69.                    case VFW_E_CANNOT_LOAD_SOURCE_FILTER:
70.                        err = IDS_MAINFRM_87;
71.                        break;
72.                    case VFW_E_CANNOT_RENDER:
73.                        err = IDS_MAINFRM_88;
74.                        break;
75.                    case VFW_E_INVALID_FILE_FORMAT:
76.                        err = IDS_MAINFRM_89;
77.                        break;
78.                    case VFW_E_NOT_FOUND:
79.                        err = IDS_MAINFRM_90;
80.                        break;
81.                    case VFW_E_UNKNOWN_FILE_TYPE:
82.                        err = IDS_MAINFRM_91;
83.                        break;
84.                    case VFW_E_UNSUPPORTED_STREAM:
85.                        err = IDS_MAINFRM_92;
```

```

86.         break;
87.     case RFS_E_NO_FILES:
88.         err = IDS_RFS_NO_FILES;
89.         break;
90.     case RFS_E_COMPRESSED:
91.         err = IDS_RFS_COMPRESSED;
92.         break;
93.     case RFS_E_ENCRYPTED:
94.         err = IDS_RFS_ENCRYPTED;
95.         break;
96.     case RFS_E_MISSING_VOLS:
97.         err = IDS_RFS_MISSING_VOLS;
98.         break;
99.     }
100.
101.     throw err;
102. }
103. }
104.
105. // We don't keep track of the standard input since that hardly makes any sense
106. if (s.fKeepHistory && fn != _T("pipe:0")) {
107.     CRecentFileList* pMRU = bMainFile ? &s.MRU : &s.MRUDub;
108.     pMRU->ReadList();
109.     pMRU->Add(fn);
110.     pMRU->WriteList();
111.     SHAddToRecentDocs(SHARD_PATH, fn);
112. }
113.
114. if (bMainFile) {
115.     pOFD->title = fn;
116. }
117.
118. bMainFile = false;
119.
120. if (m_fCustomGraph) {
121.     break;
122. }
123. }
124.
125. if (s.fReportFailedPins) {
126.     CComQIPtr<IGraphBuilderDeadEnd> pGBDE = m_pGB;
127.     if (pGBDE && pGBDE->GetCount()) {
128.         CMediaTypesDlg(pGBDE, GetModalParent()).DoModal();
129.     }
130. }
131.
132. if (!(m_pAMOP = m_pGB)) {
133.     BeginEnumFilters(m_pGB, pEF, pBF);
134.     if (m_pAMOP = pBF) {
135.         break;
136.     }
137.     EndEnumFilters;
138. }
139.
140. if (FindFilter(CLSID_MPCShoutcastSource, m_pGB)) {
141.     m_fUpdateInfoBar = true;
142. }
143.
144. SetupChapters();
145.
146. CComQIPtr<IKeyFrameInfo> pKFI;
147. BeginEnumFilters(m_pGB, pEF, pBF);
148. if (pKFI = pBF) {
149.     break;
150. }
151. EndEnumFilters;
152. UINT nKFs = 0;
153. if (pKFI && S_OK == pKFI->GetKeyFrameCount(nKFs) && nKFs > 0) {
154.     UINT k = nKFs;
155.     if (!m_kfs.SetCount(k) || S_OK != pKFI->GetKeyFrames(&TIME_FORMAT_MEDIA_TIME, m_kfs.GetData(), k) || k != nKFs) {
156.         m_kfs.RemoveAll();
157.     }
158. }
159. //设置播放模式
160. SetPlaybackMode(PM_FILE);
161. }

```

从OpenFile()函数的源代码我们可以看出，mpc-hc调用了DirectShow的函数，打开相应的文件。比如说：

```
HRESULT hr = m_pGB->RenderFile(CStringW(fn), nullptr);
```

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/13290345>

文章标签：[mpc-hc](#) [源代码](#) [directshow](#) [开源](#) [播放器](#)

个人分类：[MPC-HC](#)

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