原 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
class CMainFrame : public CFrameWnd, public CDropTarget
{
    ...

    // TODO: wrap these graph objects into a class to make it look cleaner
    //各种DirectShow接口
    //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。
    CComPtr<IGraphBuilder2> m_pGB;
    CComQIPtr<IMediaControl> m_pMC;
    CComQIPtr<IMediaEventEx> m_pME;
    CComQIPtr<IVideoWindow> m_pVW;
    //这里也可以获得
    //分辨率，比特率，帧率
    //经过测试，貌似这里取不到值 = =
    CComQIPtr<IBasicVideo> m_pBV;
    //音量，均衡器等信息
    CComQIPtr<IBasicAudio> m_pBA;
    CComQIPtr<IMediaSeeking> m_pMS;
    CComQIPtr<IVideoFrameStep> m_pFS;
    //接收端质量信息：抖动，抖动，视音频同步情况等。。。
    CComQIPtr<IQualProp, &IID_IQualProp> m_pQP;
    //缓存信息
    CComQIPtr<IBufferInfo> m_pBI;
    CComQIPtr<IAMOpenProgress> m_pAMOP;
    CComPtr<IVMRMixerControl9> m_pVMRMC;
    CComPtr<IMFVideoDisplayControl> m_pMFVDC;
    CComPtr<IMFVideoProcessor> m_pMFVP;
    CComPtr<IVMRWindowlessControl9> m_pVMRWC;
    ...
    void SetVolumeBoost(UINT nAudioBoost);
    void SetBalance(int balance);

    // subtitles
    CCritSec m_csSubLock;

    CList<SubtitleInput> m_pSubStreams;
    POSITION m posFirstExtSub;
```

```
    ISubStream* m_pCurrentSubStream;

    SubtitleInput* GetSubtitleInput(int& i, bool bIsOffset = false);

    friend class CTextPassThruFilter;

    // windowing

    CRect m_lastWindowRect;
    CPoint m_lastMouseMove;

    void ShowControls(int nCS, bool fSave = false);
    void SetUIPreset(int iCaptionMenuMode, UINT nCS);

    void SetDefaultWindowRect(int iMonitor = 0);
    void SetDefaultFullscreenState();
    void RestoreDefaultWindowRect();
    void ZoomVideoWindow(bool snap = true, double scale = ZOOM_DEFAULT_LEVEL);
    double GetZoomAutoFitScale(bool bLargerOnly = false) const;

    void SetAlwaysOnTop(int iOnTop);

    // dynamic menus
    // 动态菜单
    void SetupOpenCDSubMenu();
    void SetupFiltersSubMenu();
    void SetupAudioSwitcherSubMenu();
    void SetupSubtitlesSubMenu();
    ...

    CMenu m_popupmain, m_popup;
    CMenu m_opencds;
    CMenu m_filters, m_subtitles, m_audios;
    CMenu m_language;
    ...

    // chapters (file mode)
    CComPtr<IDSMChapterBag> m_pCB;
    void SetupChapters();

    // chapters (dvd mode)
    void SetupDVDChapters();

    void SetupIViAudReg();

    void AddTextPassThruFilter();

    int m_nLoops;
    UINT m_nLastSkipDirection;

    bool m_fCustomGraph;
    ...

public:
    void StartWebServer(int nPort);
    void StopWebServer();

    CString GetStatusMessage() const;
    int GetPlaybackMode() const { return m_iPlaybackMode; }
    void SetPlaybackMode(int iNewStatus);
    bool IsMuted() { return m_wndToolBar.GetVolume() == -10000; }
    int GetVolume() { return m_wndToolBar.m_volctrl.GetPos(); }

public:
    CMainFrame();
    DECLARE_DYNAMIC(CMainFrame)

    // Attributes
public:
    bool m_fFullScreen;
    bool m_fFirstFSAfterLaunchOnFS;
    bool m_fStartInD3DFullscreen;
    bool m_fHideCursor;
    CMenu m_navaudio, m_navsubtitle;

    CComPtr<IBaseFilter> m_pRefClock; // Adjustable reference clock. GothSync
    CComPtr<ISyncClock> m_pSyncClock;
    ...

    CControlBar* m_pLastBar;

protected:
    MPC_LOADSTATE m_iMediaLoadState;
    bool m_bFirstPlay;

    bool m_fAudioOnly;
    dispmode m_dmBeforeFullscreen;
    CString m_LastOpenFile, m_LastOpenBDPath;
    HMONITOR m_LastWindow_HM;

    DVD_DOMAIN m_iDVDDomain;
```

```
    DWORD m_iDVDTitle;
    double m_dSpeedRate;
    double m_ZoomX, m_ZoomY, m_PosX, m_PosY;
    int m_AngleX, m_AngleY, m_AngleZ;

    //操作 Operations
    //打开一个媒体
    bool OpenMediaPrivate(CAutoPtr<OpenMediaData> pOMD);
    //关闭媒体
    void CloseMediaPrivate();
    void DoTunerScan(TunerScanData* pTSD);

    CWnd* GetModalParent();

    void OpenCreateGraphObject(OpenMediaData* pOMD);
    //打开文件
    void OpenFile(OpenFileData* pOFD);
    //打开DVD
    void OpenDVD(OpenDVDData* pODD);
    //打开摄像头
    void OpenCapture(OpenDeviceData* pODD);
    HRESULT OpenBDAGraph();
    void OpenCustomizeGraph();
    //设置视频窗口
    void OpenSetupVideo();
    //设置音量
    void OpenSetupAudio();
    void OpenSetupInfoBar();
    void UpdateChapterInInfoBar();
    //打开统计工具条
    void OpenSetupStatsBar();
    //打开状态工具条
    void OpenSetupStatusBar();
    // void OpenSetupToolBar();
    void OpenSetupCaptureBar();
    //设置窗口标题
    void OpenSetupWindowTitle(CString fn = _T(""));
    void AutoChangeMonitorMode();

    bool GraphEventComplete();

    friend class CGraphThread;
    CGraphThread* m_pGraphThread;
    bool m_bOpenedThruThread;

    CAtlArray<REFERENCE_TIME> m_kfs;

    bool m_fOpeningAborted;
    bool m_bWasSnapped;

public:
    void OpenCurPlaylistItem(REFERENCE_TIME rtStart = 0);
    void OpenMedia(CAutoPtr<OpenMediaData> pOMD);
    void PlayFavoriteFile(CString fav);
    void PlayFavoriteDVD(CString fav);
    bool ResetDevice();
    bool DisplayChange();
    void CloseMedia();
    void StartTunerScan(CAutoPtr<TunerScanData> pTSD);
    void StopTunerScan();
    HRESULT SetChannel(int nChannel);

    void AddCurDevToPlaylist();

    bool m_fTrayIcon;
    //设置系统托盘图标
    void ShowTrayIcon(bool fShow);
    void SetTrayTip(CString str);

    CSize GetVideoSize() const;
    void ToggleFullscreen(bool fToNearest, bool fSwitchScreenResWhenHasTo);
    void ToggleD3DFullscreen(bool fSwitchScreenResWhenHasTo);
    void MoveVideoWindow(bool fShowStats = false);
    void RepaintVideo();
    void HideVideoWindow(bool fHide);

    OAFilterState GetMediaState() const;
    REFERENCE_TIME GetPos() const;
    REFERENCE_TIME GetDur() const;
    void SeekTo(REFERENCE_TIME rt, bool fSeekToKeyFrame = false);
    //设置播放速率
    void SetPlayingRate(double rate);

    DWORD SetupAudioStreams();
    DWORD SetupSubtitleStreams();
    //字幕
    bool LoadSubtitle(CString fn, ISubStream** actualStream = nullptr, bool bAutoLoad = false);
    bool SetSubtitle(int i, bool bIsOffset = false, bool bDisplayMessage = false, bool bApplyDefStyle = false);
    void SetSubtitle(ISubStream* pSubStream, bool bApplyDefStyle = false);
    void ToggleSubtitleOnOff(bool bDisplayMessage = false);
    void ReplaceSubtitle(const ISubStream* pSubStreamOld, ISubStream* pSubStreamNew);
```

```
    void InvalidateSubtitle(DWORD_PTR nSubtitleId = -1, REFERENCE_TIME rtInvalidate = -1);
    void ReloadSubtitle();
    HRESULT InsertTextPassThruFilter(IBaseFilter* pBF, IPin* pPin, IPin* pPinto);

    void SetAudioTrackIdx(int index);
    void SetSubtitleTrackIdx(int index);

    void AddFavorite(bool fDisplayMessage = false, bool fShowDialog = true);

    // shaders
    CAtlList<CString> m_shaderlabels;
    CAtlList<CString> m_shaderlabelsScreenSpace;
    void SetShaders();
    void UpdateShaders(CString label);

    // capturing
    bool m_fCapturing;
    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
    bool BuildToCapturePreviewPin(IBaseFilter* pVidCap, IPin** pVidCapPin, IPin** pVidPrevPin,
                                  IBaseFilter* pAudCap, IPin** pAudCapPin, IPin** pAudPrevPin);
    bool BuildGraphVideoAudio(int fVPreview, bool fVCapture, int fAPreview, bool fACapture);
    bool DoCapture(), StartCapture(), StopCapture();

    bool DoAfterPlaybackEvent();
    void ParseDirs(CAtlList<CString>& sl);
    bool SearchInDir(bool bDirForward, bool bLoop = false);

    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual BOOL PreTranslateMessage(MSG* pMsg);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo);
    virtual void RecalcLayout(BOOL bNotify = TRUE);

    // DVB capture
    void ShowCurrentChannelInfo(bool fShowOSD = true, bool fShowInfoBar = false);

    // Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
    // control bar embedded members
    CChildView m_wndView;

    UINT m_nCS;
    CPlayerSeekBar m_wndSeekBar;
    CPlayerToolBar m_wndToolBar;
    CPlayerInfoBar m_wndInfoBar;
    CPlayerInfoBar m_wndStatsBar;
    CPlayerStatusBar m_wndStatusBar;
    CList<CControlBar*> m_bars;

    CPlayerSubresyncBar m_wndSubresyncBar;
    CPlayerPlaylistBar m_wndPlaylistBar;
    CPlayerCaptureBar m_wndCaptureBar;
    CPlayerNavigationBar m_wndNavigationBar;
    CPlayerShaderEditorBar m_wndShaderEditorBar;
    CEditListEditor m_wndEditListEditor;
    CList<CSizingControlBar*> m_dockingbars;
    ...

    // Generated message map functions

    DECLARE_MESSAGE_MAP()

public:
    //打开的时候加载
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //关闭的时候加载
    afx_msg void OnDestroy();

    afx_msg LRESULT OnTaskBarRestart(WPARAM, LPARAM);
    afx_msg LRESULT OnNotifyIcon(WPARAM, LPARAM);
    afx_msg LRESULT OnTaskBarThumbnailsCreate(WPARAM, LPARAM);

    afx_msg LRESULT OnSkypeAttach(WPARAM wParam, LPARAM lParam);

    afx_msg void OnSetFocus(CWnd* pOldWnd);
    afx_msg void OnGetMinMaxInfo(MINMAXINFO* lpMMI);
    afx_msg void OnMove(int x, int y);
    afx_msg void OnMoving(UINT fwSide, LPRECT pRect);
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnSizing(UINT fwSide, LPRECT pRect);
    afx_msg void OnDisplayChange();

    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnActivateApp(BOOL bActive, DWORD dwThreadID);
```

```
    afx_msg LRESULT OnAppCommand(WPARAM wParam, LPARAM lParam);
    afx_msg void OnRawInput(UINT nInputcode, HRAWINPUT hRawInput);

    afx_msg LRESULT OnHotKey(WPARAM wParam, LPARAM lParam);

    afx_msg void OnTimer(UINT_PTR nIDEvent);

    afx_msg LRESULT OnGraphNotify(WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnResetDevice(WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnRepaintRenderLess(WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnResumeFromState(WPARAM wParam, LPARAM lParam);
    ...


    // menu item handlers

    afx_msg void OnFileOpenQuick();
    afx_msg void OnFileOpenmedia();
    afx_msg void OnUpdateFileOpen(CCmdUI* pCmdUI);
    afx_msg BOOL OnCopyData(CWnd* pWnd, COPYDATASTRUCT* pCopyDataStruct);
    afx_msg void OnFileOpendvd();
    afx_msg void OnFileOpendevice();
    afx_msg void OnFileOpenCD(UINT nID);
    afx_msg void OnFileReopen();
    afx_msg void OnFileRecycle();
    afx_msg void OnDropFiles(HDROP hDropInfo); // no menu item
    ...
    afx_msg void OnHelpHomepage();
    afx_msg void OnHelpCheckForUpdate();
    afx_msg void OnHelpToolbarImages();
    afx_msg void OnHelpDonate();
    //关闭的时候加载
    afx_msg void OnClose();

    afx_msg void OnLanguage(UINT nID);
    afx_msg void OnUpdateLanguage(CCmdUI* pCmdUI);

    CMPC_Lcd m_Lcd;

    // ==== Added by CASIMIR666
    CWnd*           m_pVideoWnd;            // Current Video (main display screen or 2nd)
    SIZE            m_fullWndSize;
    CFullscreenWnd* m_pFullscreenWnd;
    CVMROSD     m_OSD;
    bool        m_bRemainingTime;
    int         m_nCurSubtitle;
    long        m_lSubtitleShift;
    REFERENCE_TIME m_rtCurSubPos;
    CString     m_strTitle;
    bool        m_bToggleShader;
    bool        m_bToggleShaderScreenSpace;
    bool        m_bInOptions;
    bool        m_bStopTunerScan;
    bool        m_bLockedZoomVideoWindow;
    int         m_nLockedZoomVideoWindow;
    bool        m_fSetChannelActive;

    void        SetLoadState(MPC_LOADSTATE iState);
    void        SetPlayState(MPC_PLAYSTATE iState);
    bool        CreateFullScreenWindow();
    void        SetupEVRColorControl();
    void        SetupVMR9ColorControl();
    void        SetColorControl(DWORD flags, int& brightness, int& contrast, int& hue, int& saturation);
    void        SetClosedCaptions(bool enable);
    LPCTSTR     GetDVDAudioFormatName(const DVD_AudioAttributes& ATR) const;
    void        SetAudioDelay(REFERENCE_TIME rtShift);
    void        SetSubtitleDelay(int delay_ms);
    //void      AutoSelectTracks();
    bool        IsRealEngineCompatible(CString strFilename) const;
    void        SetTimersPlay();
    void        KillTimersStop();


    // MPC API functions
    void        ProcessAPICommand(COPYDATASTRUCT* pCDS);
    void        SendAPICommand(MPCAPI_COMMAND nCommand, LPCWSTR fmt, ...);
    void        SendNowPlayingToApi();
    void        SendSubtitleTracksToApi();
    void        SendAudioTracksToApi();
    void        SendPlaylistToApi();
    ...

protected:
    // GDI+
    ULONG_PTR m_gdiplusToken;
    virtual LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam);
    void WTSRegisterSessionNotification();
    void WTSUnRegisterSessionNotification();

    DWORD m_nMenuHideTick;
    UINT m_nSeekDirection;
public:
```

```
public:
    afx_msg UINT OnPowerBroadcast(UINT nPowerEvent, UINT nEventData);
    afx_msg void OnSessionChange(UINT nSessionState, UINT nId);

    void EnableShaders1(bool enable);
    void EnableShaders2(bool enable);

    CAtlList<CHdmvClipInfo::PlaylistItem> m_MPLSPlaylist;
    bool m_bIsBDPlay;
    bool OpenBD(CString Path);
};
```

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：

```cpp
OnFileOpendevice()//打开一个设备（比如说摄像头）
OnFileOpendvd()//打开一个DVD
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：

```cpp
//打开媒体（非private）
void CMainFrame::OpenMedia(CAutoPtr<OpenMediaData> pOMD)
{
    // shortcut
    if (OpenDeviceData* p = dynamic_cast<OpenDeviceData*>(pOMD.m_p)) {
        if (m_iMediaLoadState == MLS_LOADED && m_pAMTuner
                && m_VidDispName == p->DisplayName[0] && m_AudDispName == p->DisplayName[1]) {
            m_wndCaptureBar.m_capdlg.SetVideoInput(p->vinput);
            m_wndCaptureBar.m_capdlg.SetVideoChannel(p->vchannel);
            m_wndCaptureBar.m_capdlg.SetAudioInput(p->ainput);
            SendNowPlayingToSkype();
            return;
        }
    }

    if (m_iMediaLoadState != MLS_CLOSED) {
        CloseMedia();
    }

    //m_iMediaLoadState = MLS_LOADING; // HACK: hides the logo

    const CAppSettings& s = AfxGetAppSettings();

    bool fUseThread = m_pGraphThread && s.fEnableWorkerThreadForOpening;

    if (OpenFileData* p = dynamic_cast<OpenFileData*>(pOMD.m_p)) {
        if (!p->fns.IsEmpty()) {
            engine_t e = s.m_Formats.GetEngine(p->fns.GetHead());
            if (e != DirectShow /*&& e != RealMedia && e != QuickTime*/) {
                fUseThread = false;
            }
        }
    } else if (OpenDeviceData* p = dynamic_cast<OpenDeviceData*>(pOMD.m_p)) {
        fUseThread = false;
    }

    // Create D3DFullscreen window if launched in fullscreen
    if (s.IsD3DFullscreen() && m_fStartInD3DFullscreen) {
        if (s.AutoChangeFullscrRes.bEnabled) {
            AutoChangeMonitorMode();
        }
        CreateFullScreenWindow();
        m_pVideoWnd = m_pFullscreenWnd;
        m_fStartInD3DFullscreen = false;
    } else {
        m_pVideoWnd = &m_wndView;
    }

    if (fUseThread) {
        m_pGraphThread->PostThreadMessage(CGraphThread::TM_OPEN, 0, (LPARAM)pOMD.Detach());
        m_bOpenedThruThread = true;
    } else {
        //打开媒体（private）
        OpenMediaPrivate(pOMD);
        m_bOpenedThruThread = false;
    }
}
```

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
//打开一个媒体（private）
bool CMainFrame::OpenMediaPrivate(CAutoPtr<OpenMediaData> pOMD)
{
    //获得设置信息
    CAppSettings& s = AfxGetAppSettings();

    if (m_iMediaLoadState != MLS_CLOSED) {
        ASSERT(0);
        return false;
    }
    //OpenFileData
    //OpenDVDData
    //OpenDeviceData
    //里面包含了文件或者DVD信息（名称等）
    OpenFileData* pFileData = dynamic_cast<OpenFileData*>(pOMD.m_p);
    OpenDVDData* pDVDData = dynamic_cast<OpenDVDData*>(pOMD.m_p);
    OpenDeviceData* pDeviceData = dynamic_cast<OpenDeviceData*>(pOMD.m_p);
    if (!pFileData && !pDVDData  && !pDeviceData) {
        ASSERT(0);
        return false;
    }

    // Clear DXVA state ...
    ClearDXVAState();

#ifdef _DEBUG
    // Debug trace code - Begin
    // Check for bad / buggy auto loading file code
    if (pFileData) {
        POSITION pos = pFileData->fns.GetHeadPosition();
        UINT index = 0;
        while (pos != nullptr) {
            CString path = pFileData->fns.GetNext(pos);
            TRACE(_T("--> CMainFrame::OpenMediaPrivate - pFileData->fns[%d]:\n"), index);
            TRACE(_T("\t%ws\n"), path.GetString()); // %ws - wide character string always
            index++;
        }
    }
    // Debug trace code - End
#endif

    CString mi_fn = _T("");

    if (pFileData) {
        if (pFileData->fns.IsEmpty()) {
            return false;
        }

        CString fn = pFileData->fns.GetHead();

        int i = fn.Find(_T(":\\"));
        if (i > 0) {
            CString drive = fn.Left(i + 2);
            UINT type = GetDriveType(drive);
            CAtlList<CString> sl;
            if (type == DRIVE_REMOVABLE || type == DRIVE_CDROM && GetCDROMType(drive[0], sl) != CDROM_Audio) {
                int ret = IDRETRY;
                while (ret == IDRETRY) {
                    WIN32_FIND_DATA findFileData;
                    HANDLE h = FindFirstFile(fn, &findFileData);
                    if (h != INVALID_HANDLE_VALUE) {
                        FindClose(h);
                        ret = IDOK;
                    } else {
                        CString msg;
                        msg.Format(IDS_MAINFRM_114, fn);
                        ret = AfxMessageBox(msg, MB_RETRYCANCEL);
                    }
                }

                if (ret != IDOK) {
                    return false;
                }
            }
            mi_fn = fn;
        }
    }

    SetLoadState(MLS_LOADING);

    // FIXME: Don't show "Closed" initially
    PostMessage(WM_KICKIDLE);

    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 (pDVDData) {
113.            //DVD
114.            OpenDVD(pDVDData);
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 IVMRWindowlessCon
rol9
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
```

```
        OpenCustomizeGraph();

        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }
        //设置视频窗口
        OpenSetupVideo();

        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }
        //设置音量
        OpenSetupAudio();

        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }

        if (m_pCAP && (!m_fAudioOnly || m_fRealMediaGraph)) {

            if (s.fDisableInternalSubtitles) {
                m_pSubStreams.RemoveAll(); // Needs to be replaced with code that checks for forced subtitles.
            }

            m_posFirstExtSub = nullptr;
            POSITION pos = pOMD->subs.GetHeadPosition();
            while (pos) {
                LoadSubtitle(pOMD->subs.GetNext(pos), nullptr, true);
            }
        }

        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }
        //设置视频窗口标题
        OpenSetupWindowTitle(pOMD->title);

        if (s.fEnableEDLEditor) {
            m_wndEditListEditor.OpenFile(pOMD->title);
        }

        if (::GetCurrentThreadId() == AfxGetApp()->m_nThreadID) {
            OnFilePostOpenmedia();
        } else {
            PostMessage(WM_COMMAND, ID_FILE_POST_OPENMEDIA);
        }

        while (m_iMediaLoadState != MLS_LOADED
                && m_iMediaLoadState != MLS_CLOSING // FIXME
              ) {
            Sleep(50);
        }
        //设置音频流
        DWORD audstm = SetupAudioStreams();
        //设置字幕流
        DWORD substm = SetupSubtitleStreams();

        if (audstm) {
            OnPlayAudio(ID_AUDIO_SUBITEM_START + audstm);
        }
        if (substm) {
            SetSubtitle(substm - 1);
        }

        // PostMessage instead of SendMessage because the user might call CloseMedia and then we would deadlock

        PostMessage(WM_COMMAND, ID_PLAY_PAUSE);

        m_bFirstPlay = true;

        if (!(s.nCLSwitches & CLSW_OPEN) && (s.nLoops > 0)) {
            PostMessage(WM_COMMAND, ID_PLAY_PLAY);
        } else {
            // If we don't start playing immediately, we need to initialize
            // the seekbar and the time counter.
            OnTimer(TIMER_STREAMPOSPOLLER);
            OnTimer(TIMER_STREAMPOSPOLLER2);
        }

        s.nCLSwitches &= ~CLSW_OPEN;

        if (pFileData) {
            if (pFileData->rtStart > 0) {
                PostMessage(WM_RESUMEFROMSTATE, (WPARAM)PM_FILE, (LPARAM)(pFileData->rtStart / 10000));  // REFERENCE_TIME doesn't fit in LPARAM under a 32bit env.
            }
        } else if (pDVDData) {
            if (pDVDData->pDvdState) {
                PostMessage(WM_RESUMEFROMSTATE, (WPARAM)PM_DVD, (LPARAM)(CComPtr<IDvdState>(pDVDData->pDvdState).Detach()));    // must be released by the called message handler
            }
```

```
        } else if (pDeviceData) {
            m_wndCaptureBar.m_capdlg.SetVideoInput(pDeviceData->vinput);
            m_wndCaptureBar.m_capdlg.SetVideoChannel(pDeviceData->vchannel);
            m_wndCaptureBar.m_capdlg.SetAudioInput(pDeviceData->ainput);
        }
    } catch (LPCTSTR msg) {
        err = msg;
    } catch (CString& msg) {
        err = msg;
    } catch (UINT msg) {
        err.LoadString(msg);
    }

    EndWaitCursor();

    if (!err.IsEmpty()) {
        //关闭
        CloseMediaPrivate();
        m_closingmsg = err;

        if (err != ResStr(IDS_AG_ABORTED)) {
            if (pFileData) {
                m_wndPlaylistBar.SetCurValid(false);

                if (m_wndPlaylistBar.IsAtEnd()) {
                    m_nLoops++;
                }

                if (s.fLoopForever || m_nLoops < s.nLoops) {
                    bool hasValidFile = false;

                    if (m_nLastSkipDirection == ID_NAVIGATE_SKIPBACK) {
                        hasValidFile = m_wndPlaylistBar.SetPrev();
                    } else {
                        hasValidFile = m_wndPlaylistBar.SetNext();
                    }

                    if (hasValidFile) {
                        OpenCurPlaylistItem();
                    }
                } else if (m_wndPlaylistBar.GetCount() > 1) {
                    DoAfterPlaybackEvent();
                }
            } else {
                OnNavigateSkip(ID_NAVIGATE_SKIPFORWARD);
            }
        }
    } else {
        m_wndPlaylistBar.SetCurValid(true);

        // Apply command line audio shift
        if (s.rtShift != 0) {
            SetAudioDelay(s.rtShift);
            s.rtShift = 0;
        }
    }

    m_nLastSkipDirection = 0;

    if (s.AutoChangeFullscrRes.bEnabled && (m_fFullScreen || IsD3DFullScreenMode())) {
        AutoChangeMonitorMode();
    }
    if (m_fFullScreen && s.fRememberZoomLevel) {
        m_fFirstFSAfterLaunchOnFS = true;
    }

    m_LastOpenFile = pOMD->title;

    PostMessage(WM_KICKIDLE); // calls main thread to update things

    if (!m_bIsBDPlay) {
        m_MPLSPlaylist.RemoveAll();
        m_LastOpenBDPath = _T("");
    }
    m_bIsBDPlay = false;

    return err.IsEmpty();
}
```

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
//打开文件
void CMainFrame::OpenFile(OpenFileData* pOFD)
{
    if (pOFD->fns.IsEmpty()) {
        throw (UINT)IDS_MAINFRM_81;
    }
    //获取设置
    CAppSettings& s = AfxGetAppSettings();

    bool bMainFile = true;

    POSITION pos = pOFD->fns.GetHeadPosition();
    while (pos) {
        CString fn = pOFD->fns.GetNext(pos);

        fn.Trim();
        if (fn.IsEmpty() && !bMainFile) {
            break;
        }
        //使用DirectShow播放文件
        HRESULT hr = m_pGB->RenderFile(CStringW(fn), nullptr);

        if (bMainFile) {
            // Don't try to save file position if source isn't seekable
            REFERENCE_TIME rtDur = 0;
            m_pMS->GetDuration(&rtDur);

            m_bRememberFilePos = s.fKeepHistory && s.fRememberFilePos && rtDur > 0;

            if (m_bRememberFilePos && !s.filePositions.AddEntry(fn)) {
                REFERENCE_TIME rtPos = s.filePositions.GetLatestEntry()->llPosition;
                if (m_pMS) {
                    m_pMS->SetPositions(&rtPos, AM_SEEKING_AbsolutePositioning, nullptr, AM_SEEKING_NoPositioning);
                }
            }
        }
        QueryPerformanceCounter(&m_liLastSaveTime);

        if (FAILED(hr)) {
            if (bMainFile) {
                if (s.fReportFailedPins) {
                    CComQIPtr<IGraphBuilderDeadEnd> pGBDE = m_pGB;
                    if (pGBDE && pGBDE->GetCount()) {
                        CMediaTypesDlg(pGBDE, GetModalParent()).DoModal();
                    }
                }

                UINT err;

                switch (hr) {
                    case E_ABORT:
                    case RFS_E_ABORT:
                        err = IDS_MAINFRM_82;
                        break;
                    case E_FAIL:
                    case E_POINTER:
                    default:
                        err = IDS_MAINFRM_83;
                        break;
                    case E_INVALIDARG:
                        err = IDS_MAINFRM_84;
                        break;
                    case E_OUTOFMEMORY:
                        err = IDS_AG_OUT_OF_MEMORY;
                        break;
                    case VFW_E_CANNOT_CONNECT:
                        err = IDS_MAINFRM_86;
                        break;
                    case VFW_E_CANNOT_LOAD_SOURCE_FILTER:
                        err = IDS_MAINFRM_87;
                        break;
                    case VFW_E_CANNOT_RENDER:
                        err = IDS_MAINFRM_88;
                        break;
                    case VFW_E_INVALID_FILE_FORMAT:
                        err = IDS_MAINFRM_89;
                        break;
                    case VFW_E_NOT_FOUND:
                        err = IDS_MAINFRM_90;
                        break;
                    case VFW_E_UNKNOWN_FILE_TYPE:
                        err = IDS_MAINFRM_91;
                        break;
                    case VFW_E_UNSUPPORTED_STREAM:
                        err = IDS_MAINFRM_92;
```

```
                        break;
                    case RFS_E_NO_FILES:
                        err = IDS_RFS_NO_FILES;
                        break;
                    case RFS_E_COMPRESSED:
                        err = IDS_RFS_COMPRESSED;
                        break;
                    case RFS_E_ENCRYPTED:
                        err = IDS_RFS_ENCRYPTED;
                        break;
                    case RFS_E_MISSING_VOLS:
                        err = IDS_RFS_MISSING_VOLS;
                        break;
                }

                throw err;
            }
        }

        // We don't keep track of the standard input since that hardly makes any sense
        if (s.fKeepHistory && fn != _T("pipe:0")) {
            CRecentFileList* pMRU = bMainFile ? &s.MRU : &s.MRUDub;
            pMRU->ReadList();
            pMRU->Add(fn);
            pMRU->WriteList();
            SHAddToRecentDocs(SHARD_PATH, fn);
        }

        if (bMainFile) {
            pOFD->title = fn;
        }

        bMainFile = false;

        if (m_fCustomGraph) {
            break;
        }
    }

    if (s.fReportFailedPins) {
        CComQIPtr<IGraphBuilderDeadEnd> pGBDE = m_pGB;
        if (pGBDE && pGBDE->GetCount()) {
            CMediaTypesDlg(pGBDE, GetModalParent()).DoModal();
        }
    }

    if (!(m_pAMOP = m_pGB)) {
        BeginEnumFilters(m_pGB, pEF, pBF);
        if (m_pAMOP = pBF) {
            break;
        }
        EndEnumFilters;
    }

    if (FindFilter(CLSID_MPCShoutcastSource, m_pGB)) {
        m_fUpdateInfoBar = true;
    }

    SetupChapters();

    CComQIPtr<IKeyFrameInfo> pKFI;
    BeginEnumFilters(m_pGB, pEF, pBF);
    if (pKFI = pBF) {
        break;
    }
    EndEnumFilters;
    UINT nKFs = 0;
    if (pKFI && S_OK == pKFI->GetKeyFrameCount(nKFs) && nKFs > 0) {
        UINT k = nKFs;
        if (!m_kfs.SetCount(k) || S_OK != pKFI->GetKeyFrames(&TIME_FORMAT_MEDIA_TIME, m_kfs.GetData(), k) || k != nKFs) {
            m_kfs.RemoveAll();
        }
    }
    //设置播放模式
    SetPlaybackMode(PM_FILE);
}
```

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);

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