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

2013年10月28日 23:52:56 阅读数：5853

===

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

===

上一篇文章分析了Media Player Classic - HC（mpc-hc）的源代码中的核心类 CMainFrame： Media Player Classic - HC 源代码分析 2：核心类 （CMainFrame） （1）

主要介绍了CMainFrame类中的以下几个函数（“->”代表调用关系）：

OpenMedia() -> OpenMediaPrivate()-> OpenFile()

本文补充介绍CMainFrame类中的其他一些函数。

再回顾一下打开文件功能主要所在的函数OpenMediaPrivate()：

[cpp]

```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;

    m_fUpdateInfoBar = false;
    BeginWaitCursor();

    try {
        CComPtr<IVMRMixerBitmap9>    pVMB;
        CComPtr<IMFVideoMixerBitmap> pMFVMB;
        CComPtr<IMadVRTextOsd>       pMVTO;
        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }

        OpenCreateGraphObject(pOMD);

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

        SetupIViAudReg();

        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }
        //按类型的不同打开不同的文件
        if (pFileData) {
            //文件
            OpenFile(pFileData);
        } else if (pDVDData) {
            //DVD
            OpenDVD(pDVDData);
        } else if (pDeviceData) {
            if (s.iDefaultCaptureDevice == 1) {
                HRESULT hr = OpenBDAGraph();
                if (FAILED(hr)) {
                    throw (UINT)IDS_CAPTURE_ERROR_DEVICE;
                }
            } else {
                OpenCapture(pDeviceData);
            }
        } else {
            throw (UINT)IDS_INVALID_PARAMS_ERROR;
        }

        m_pCAP2 = nullptr;
        m_pCAP = nullptr;
        //查找接口
        m_pGB->FindInterface(__uuidof(ISubPicAllocatorPresenter), (void**)&m_pCAP, TRUE);
        m_pGB->FindInterface(__uuidof(ISubPicAllocatorPresenter2), (void**)&m_pCAP2, TRUE);
```

```cpp
        m_pGB->FindInterface(__uuidof(IVMRWindowlessControl9), (void**)&m_pVMRWC, FALSE); // might have IVMRMixerBitmap9, but not IVMRWindowlessControl9
        m_pGB->FindInterface(__uuidof(IVMRMixerControl9), (void**)&m_pVMRMC, TRUE);
        m_pGB->FindInterface(__uuidof(IVMRMixerBitmap9), (void**)&pVMB, TRUE);
        m_pGB->FindInterface(__uuidof(IMFVideoMixerBitmap), (void**)&pMFVMB, TRUE);
        pMVTO = m_pCAP;

        if (s.fShowOSD || s.fShowDebugInfo) { // Force OSD on when the debug switch is used
            if (pVMB) {
                m_OSD.Start(m_pVideoWnd, pVMB, IsD3DFullScreenMode());
            } else if (pMFVMB) {
                m_OSD.Start(m_pVideoWnd, pMFVMB, IsD3DFullScreenMode());
            } else if (pMVTO) {
                m_OSD.Start(m_pVideoWnd, pMVTO);
            }
        }
        //VMR9
        SetupVMR9ColorControl();

        // === EVR !
        m_pGB->FindInterface(__uuidof(IMFVideoDisplayControl), (void**)&m_pMFVDC,  TRUE);
        m_pGB->FindInterface(__uuidof(IMFVideoProcessor), (void**)&m_pMFVP, TRUE);
        if (m_pMFVDC) {
            m_pMFVDC->SetVideoWindow(m_pVideoWnd->m_hWnd);
        }

        //SetupEVRColorControl();
        //does not work at this location
        //need to choose the correct mode (IMFVideoProcessor::SetVideoProcessorMode)

        BeginEnumFilters(m_pGB, pEF, pBF) {
            if (m_pLN21 = pBF) {
                m_pLN21->SetServiceState(s.fClosedCaptions ? AM_L21_CCSTATE_On : AM_L21_CCSTATE_Off);
                break;
            }
        }
        EndEnumFilters;

        if (m_fOpeningAborted) {
            throw (UINT)IDS_AG_ABORTED;
        }
        //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
                && 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);
```

```cpp

        // 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();
}
```

来看一看OpenMediaPrivate()函数的细节：

1.开始的时候有这么一句

```cpp
CAppSettings& s = AfxGetAppSettings();
```

在这里涉及到一个类CAppSettings，存储的是mpc-hc用到的各种设置信息。源代码如下：

```cpp
//应用程序中的各种参数
class CAppSettings
{
    bool fInitialized;

    class CRecentFileAndURLList : public CRecentFileList
    {
    public:
        CRecentFileAndURLList(UINT nStart, LPCTSTR lpszSection,
                              LPCTSTR lpszEntryFormat, int nSize,
                              int nMaxDispLen = AFX_ABBREV_FILENAME_LEN);

        virtual void Add(LPCTSTR lpszPathName); // we have to override CRecentFileList::Add because the original version can't handle URLs
    };

public:
    bool fShaderEditorWasOpened;

    // cmdline params
    UINT nCLSwitches;
    CAtlList<CString>   slFiles, slDubs, slSubs, slFilters;

    // Initial position (used by command line flags)
    REFERENCE_TIME      rtShift;
    REFERENCE_TIME      rtStart;
    ULONG               lDVDTitle;
    ULONG               lDVDChapter;
    DVD_HMSF_TIMECODE   DVDPosition;

    CSize sizeFixedWindow;
    bool HasFixedWindowSize() const { return sizeFixedWindow.cx > 0 || sizeFixedWindow.cy > 0; }
    //int           iFixedWidth, iFixedHeight;
    int             iMonitor;

    CString         ParseFileName(CString const& param);
    void            ParseCommandLine(CAtlList<CString>& cmdln);

    // Added a Debug display to the screen (/debug option)
    bool            fShowDebugInfo;
    int             iAdminOption;


    //播放器 Player
    bool            fAllowMultipleInst;
    bool            fTrayIcon;
    bool            fShowOSD;
    bool            fLimitWindowProportions;
    bool            fSnapToDesktopEdges;
    bool            fHideCDROMsSubMenu;
    DWORD           dwPriority;
    int             iTitleBarTextStyle;
    bool            fTitleBarTextTitle;
    bool            fKeepHistory;
    CRecentFileAndURLList MRU;
    CRecentFileAndURLList MRUDub;
    CFilePositionList filePositions;
    CDVDPositionList  dvdPositions;
    bool            fRememberDVDPos;
    bool            fRememberFilePos;
    bool            bRememberPlaylistItems;
    bool            fRememberWindowPos;
    CRect           rcLastWindowPos;
    bool            fRememberWindowSize;
    bool            fSavePnSZoom;
    double          dZoomX;
    double          dZoomY;

    // Formats
    CMediaFormats   m_Formats;
    bool            fAssociatedWithIcons;

    // Keys
    CList<wmcmd>    wmcmds;
    HACCEL          hAccel;
    bool            fWinLirc;
    CString         strWinLircAddr;
    CWinLircClient  WinLircClient;
    bool            fUIce;
    CString         strUIceAddr;
    CUIceClient     UIceClient;
    bool            fGlobalMedia;

    //图标 Logo
    UINT            nLogoId;
    bool            fLogoExternal;
    CString         strLogoFileName;

    //web界面？ Web Inteface
    BOOL            fEnableWebServer;
    int             nWebServerPort;
    int             nCmdlnWebServerPort;
    bool            fWebServerUseCompression;
    bool            fWebServerLocalhostOnly;
    bool            fWebServerPrintDebugInfo;
    CString         strWebRoot, strWebDefIndex;
    CString         strWebServerCGI;

    //播放时候 Playback
    int             nVolume;
    bool            fMute;
    int             nBalance;
    int             nLoops;
    bool            fLoopForever;
    bool            fRewind;
    bool            fRememberZoomLevel;
    int             nAutoFitFactor;
    int             iZoomLevel;
    CStringW        strAudiosLanguageOrder;
    CStringW        strSubtitlesLanguageOrder;
    bool            fEnableWorkerThreadForOpening;
    bool            fReportFailedPins;
    bool            fAutoloadAudio;
    bool            fAutoloadSubtitles;
    bool            fBlockVSFilter;
    UINT            nVolumeStep;
    UINT            nSpeedStep;

    // DVD/OGM
    bool            fUseDVDPath;
    CString         strDVDPath;
    LCID            idMenuLang, idAudioLang, idSubtitlesLang;
    bool            fAutoSpeakerConf;
    bool            fClosedCaptions;

    //输出 Output
    CRenderersSettings m_RenderersSettings;
    int             iDSVideoRendererType;
    int             iRMVideoRendererType;
    int             iQTVideoRendererType;

    CStringW        strAudioRendererDisplayName;
    bool            fD3DFullscreen;

    //全屏 Fullscreen
    bool            fLaunchfullscreen;
    bool            fShowBarsWhenFullScreen;
    int             nShowBarsWhenFullScreenTimeOut;
    bool            fExitFullScreenAtTheEnd;
    CStringW        strFullScreenMonitor;
    AChFR           AutoChangeFullscrRes;
    bool            fRestoreResAfterExit;

    // Sync Renderer Settings

    // Capture (BDA configuration)
    int             iDefaultCaptureDevice;      // Default capture device (analog=0, 1=digital)
    CString         strAnalogVideo;
    CString         strAnalogAudio;
    int             iAnalogCountry;
    CString         strBDANetworkProvider;
    CString         strBDATuner;
    CString         strBDAReceiver;
    //CString           strBDAStandard;
    int             iBDAScanFreqStart;
    int             iBDAScanFreqEnd;
    int             iBDABandwidth;
    bool            fBDAUseOffset;
    int             iBDAOffset;
    bool            fBDAIgnoreEncryptedChannels;
    UINT            nDVBLastChannel;
    CAtlList<CDVBChannel> m_DVBChannels;
    DVB_RebuildFilterGraph nDVBRebuildFilterGraph;
    DVB_StopFilterGraph nDVBStopFilterGraph;

    // Internal Filters
    bool            SrcFilters[SRC_LAST + !SRC_LAST];
    bool            TraFilters[TRA_LAST + !TRA_LAST];

    //音频 Audio Switcher
    bool            fEnableAudioSwitcher;
    bool            fAudioNormalize;
    UINT            nAudioMaxNormFactor;
    bool            fAudioNormalizeRecover;
    UINT            nAudioBoost;
    bool            fDownSampleTo441;
    bool            fAudioTimeShift;
    int             iAudioTimeShift;
    bool            fCustomChannelMapping;
    int             nSpeakerChannels;
    DWORD           pSpeakerToChannelMap[AS_MAX_CHANNELS][AS_MAX_CHANNELS];

    // External Filters
    CAutoPtrList<FilterOverride> m_filters;

    //字幕 Subtitles
    bool            fOverridePlacement;
    int             nHorPos, nVerPos;
    int             nSubDelayInterval;

    // Default Style
    STSStyle        subdefstyle;

    // Misc
    bool            bPreferDefaultForcedSubtitles;
    bool            fPrioritizeExternalSubtitles;
    bool            fDisableInternalSubtitles;
    bool            bAllowOverridingExternalSplitterChoice;
    CString         strSubtitlePaths;
    CString         strISDb;

    // Tweaks
    int             nJumpDistS;
    int             nJumpDistM;
    int             nJumpDistL;
    bool            fFastSeek;
    bool            fShowChapters;
    bool            bNotifySkype;
    bool            fPreventMinimize;
    bool            fUseWin7TaskBar;
    bool            fLCDSupport;
    bool            fUseSearchInFolder;
    bool            fUseTimeTooltip;
    int             nTimeTooltipPosition;
    CString         strOSDFont;
    int             nOSDSize;

    //亮度色度饱和度 Miscellaneous
    int             iBrightness;
    int             iContrast;
    int             iHue;
    int             iSaturation;
    int             nUpdaterAutoCheck;
    int             nUpdaterDelay;

    // MENUS
    // View
    int             iCaptionMenuMode; // normal -> hidemenu -> frameonly -> borderless
    int             iCaptionMenuMode; // normal -> hidemenu -> frameonly -> borderless
    bool            fHideNavigation;
    UINT            nCS; // Control state for toolbars
    // Language
    LANGID          language;
    // Subtitles menu
    bool            fEnableSubtitles;
    bool            fUseDefaultSubtitlesStyle;
    // Video Frame
    int             iDefaultVideoSize;
    bool            fKeepAspectRatio;
    CSize           sizeAspectRatio;
    bool            fCompMonDeskARDiff;
    // Pan&Scan
    CString         strPnSPreset;
    CStringArray    m_pnspresets;
    // On top menu
    int             iOnTop;
    // After Playback
    bool            fExitAfterPlayback;
    bool            fNextInDirAfterPlayback;

    // WINDOWS
    // Add Favorite
    bool            bFavRememberPos;
    bool            bFavRelativeDrive;
    // Save Image...
    CString         strSnapShotPath, strSnapShotExt;
    // Save Thumbnails...
    int             iThumbRows, iThumbCols, iThumbWidth;
    // Shader Editor
    struct Shader {
        CString     label;
        CString     target;
        CString     srcdata;
    };
    CAtlList<Shader> m_shaders;
    // Shader Combiner
    bool            fToggleShader;
    bool            fToggleShaderScreenSpace;
    CString         strShaderList;
    CString         strShaderListScreenSpace;
    // Playlist (contex menu)
    bool            bShufflePlaylistItems;
    bool            bHidePlaylistFullScreen;

    // OTHER STATES
    CStringW        strLastOpenDir;
    UINT            nLastWindowType;
    UINT            nLastUsedPage;
    bool            fRemainingTime;
    bool            fLastFullScreen;

    bool            fIntRealMedia;
    //bool          fRealMediaRenderless;
    //float         dRealMediaQuickTimeFPS;
    //int           iVideoRendererType;
    //int           iQuickTimeRenderer;
    //bool          fMonitorAutoRefreshRate;
    bool            fEnableEDLEditor;

    HWND            hMasterWnd;

    bool            IsD3DFullscreen() const;
    CString         SelectedAudioRenderer() const;
    bool            IsISREnabled() const;

private:
    CString         SrcFiltersKeys[SRC_LAST + !SRC_LAST];
    CString         TraFiltersKeys[TRA_LAST + !TRA_LAST];

    __int64         ConvertTimeToMSec(const CString& time) const;
    void            ExtractDVDStartPos(CString& strParam);

    void            CreateCommands();

    void            SaveExternalFilters(CAutoPtrList<FilterOverride>& filters, LPCTSTR baseKey = IDS_R_EXTERNAL_FILTERS);
    void            LoadExternalFilters(CAutoPtrList<FilterOverride>& filters, LPCTSTR baseKey = IDS_R_EXTERNAL_FILTERS);
    void            ConvertOldExternalFiltersList();

    void            UpdateRenderersData(bool fSave);
    friend void     CRenderersSettings::UpdateData(bool bSave);

public:
    CAppSettings();
    virtual ~CAppSettings();

    void            SaveSettings();
    void            LoadSettings();
    void            SaveExternalFilters() { if (fInitialized) { SaveExternalFilters(m_filters); } };

    void            GetFav(favtype ft, CAtlList<CString>& sl) const;
    void            SetFav(favtype ft, CAtlList<CString>& sl);
    void            AddFav(favtype ft, CString s);

    CDVBChannel*    FindChannelByPref(int nPrefNumber);

    bool            GetAllowMultiInst() const;

    static bool     IsVSFilterInstalled();
    static bool     HasEVR();
};
```

由代码可见，包含的参数信息很多。在mpc-hc中，任何需要获取设置信息的地方，都可以使用AfxGetAppSettings()获得CAppSettings的引用。

2.OpenSetupVideo()这个函数的作用是设置视频窗口，源代码如下：

[cpp]

```cpp
//设置视频窗口
void CMainFrame::OpenSetupVideo()
{
    //大部分都在确定：m_fAudioOnly是否为True
    m_fAudioOnly = true;
    //获得视频的宽和高，然后调整窗口大小
    if (m_pMFVDC) { // EVR
        m_fAudioOnly = false;
    } else if (m_pCAP) {
        CSize vs = m_pCAP->GetVideoSize();
        m_fAudioOnly = (vs.cx <= 0 || vs.cy <= 0);
    } else {
        {
            long w = 0, h = 0;

            if (CComQIPtr<IBasicVideo> pBV = m_pGB) {
                pBV->GetVideoSize(&w, &h);
            }

            if (w > 0 && h > 0) {
                m_fAudioOnly = false;
            }
        }
        //如果 m_fAudioOnly=true；再检查
        if (m_fAudioOnly) {
            BeginEnumFilters(m_pGB, pEF, pBF) {
                long w = 0, h = 0;

                if (CComQIPtr<IVideoWindow> pVW = pBF) {
                    long lVisible;
                    if (FAILED(pVW->get_Visible(&lVisible))) {
                        continue;
                    }

                    pVW->get_Width(&w);
                    pVW->get_Height(&h);
                }

                if (w > 0 && h > 0) {
                    m_fAudioOnly = false;
                    break;
                }
            }
            EndEnumFilters;
        }
    }

    if (m_fShockwaveGraph) {
        m_fAudioOnly = false;
    }

    if (m_pCAP) {
        SetShaders();
    }
    // else
    {
        // TESTME
        //设置所有者。。。
        m_pVW->put_Owner((OAHWND)m_pVideoWnd->m_hWnd);
        m_pVW->put_WindowStyle(WS_CHILD | WS_CLIPSIBLINGS | WS_CLIPCHILDREN);
        m_pVW->put_MessageDrain((OAHWND)m_hWnd);

        for (CWnd* pWnd = m_wndView.GetWindow(GW_CHILD); pWnd; pWnd = pWnd->GetNextWindow()) {
            pWnd->EnableWindow(FALSE);    // little trick to let WM_SETCURSOR thru
        }
    }
    //如果只有音频，则消灭视频窗口！
    if (m_fAudioOnly && IsD3DFullScreenMode()) {
        m_pFullscreenWnd->DestroyWindow();
    }
}
```

3. OpenSetupAudio()这个函数的作用是设置音频，源代码如下：

[cpp]

```cpp
//设置音量
void CMainFrame::OpenSetupAudio()
{
    //设置音量
    m_pBA->put_Volume(m_wndToolBar.Volume);

    // FIXME
    int balance = AfxGetAppSettings().nBalance;

    int sign = balance > 0 ? -1 : 1; // -1: invert sign for more right channel
    if (balance > -100 && balance < 100) {
        balance = sign * (int)(100 * 20 * log10(1 - abs(balance) / 100.0f));
    } else {
        balance = sign * (-10000);  // -10000: only left, 10000: only right
    }
    //设置均衡
    m_pBA->put_Balance(balance);
}
```

4.如果出现问题，则会调用CloseMediaPrivate()，关闭打开的媒体。

[cpp]

```cpp
//关闭
void CMainFrame::CloseMediaPrivate()
{
    SetLoadState(MLS_CLOSING); // why it before OnPlayStop()? // TODO: remake or add detailed comments
    OnPlayStop(); // SendMessage(WM_COMMAND, ID_PLAY_STOP);
    if (m_pMC) {
        m_pMC->Stop(); // needed for StreamBufferSource, because m_iMediaLoadState is always MLS_CLOSED // TODO: fix the opening for such media
    }
    SetPlaybackMode(PM_NONE);
    m_fLiveWM = false;
    m_fEndOfStream = false;
    m_rtDurationOverride = -1;
    m_kfs.RemoveAll();
    m_pCB.Release();

    {
        CAutoLock cAutoLock(&m_csSubLock);
        m_pSubStreams.RemoveAll();
    }
    m_pSubClock.Release();

    //if (m_pVW) m_pVW->put_Visible(OAFALSE);
    //if (m_pVW) m_pVW->put_MessageDrain((OAHWND)NULL), m_pVW->put_Owner((OAHWND)NULL);

    // IMPORTANT: IVMRSurfaceAllocatorNotify/IVMRSurfaceAllocatorNotify9 has to be released before the VMR/VMR9, otherwise it will cr
 in Release()
    //各种清空
    m_OSD.Stop();
    m_pCAP2.Release();
    m_pCAP.Release();
    m_pVMRWC.Release();
    m_pVMRMC.Release();
    m_pMFVP.Release();
    m_pMFVDC.Release();
    m_pLN21.Release();
    m_pSyncClock.Release();

    m_pAMXBar.Release();
    m_pAMDF.Release();
    m_pAMVCCap.Release();
    m_pAMVCPrev.Release();
    m_pAMVSCCap.Release();
    m_pAMVSCPrev.Release();
    m_pAMASC.Release();
    m_pVidCap.Release();
    m_pAudCap.Release();
    m_pAMTuner.Release();
    m_pCGB.Release();

    m_pDVDC.Release();
    m_pDVDI.Release();
    m_pAMOP.Release();
    m_pBI.Release();
    m_pQP.Release();
    m_pFS.Release();
    m_pMS.Release();
    m_pBA.Release();
    m_pBV.Release();
    m_pVW.Release();
    m_pME.Release();
    m_pMC.Release();

    if (m_pGB) {
        m_pGB->RemoveFromROT();
        m_pGB.Release();
    }

    m_pProv.Release();

    m_fRealMediaGraph = m_fShockwaveGraph = m_fQuicktimeGraph = false;

    m_VidDispName.Empty();
    m_AudDispName.Empty();

    m_closingmsg.LoadString(IDS_CONTROLS_CLOSED);

    AfxGetAppSettings().nCLSwitches &= CLSW_OPEN | CLSW_PLAY | CLSW_AFTERPLAYBACK_MASK | CLSW_NOFOCUS;
    //设置状态
    SetLoadState(MLS_CLOSED);
}
```

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/13297291

文章标签： mpc-hc 源代码 directshow 播放器 开源

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