２００９年　卒業研究論文

マルチメディアプレイヤーについて

岡山理科大学

総合情報学部

情報科学科

澤見研究室

I04I021　片山祐輔

I05I095　山田大志

I06I040　野崎祥志

目次

1. はじめに

マルチメディアとは一般的に音楽や動画などの複数媒体をまとめて扱うためのものであり、利用者（ユーザー）の操作に対応して情報収集と情報処理が双方向性（Interactive）を持っているもののことである。そして、ハードウェアおよびソフトフェアの進歩、インターネットの普及と性能向上により、マルチメディアは身近なものになっておりＰＣを使って利用するのが一般的である。ＰＣ上でそれらのマルチメディアを利用するためには、再生するためのアプリケーションソフトウェアが必要となる。この再生するソフトウェアのことをマルチメディアプレイヤーと呼ぶ。

本研究では既存のマルチメディアプレイヤーとの仕組みについて詳しく調べ、それに基づく仕様に従ってマルチメディアプレイヤーを作成し簡単な比較評価を行った。

2. マルチメディアプレイヤー概要

マルチメディアプレイヤーとは PC などで文字、映像、動画、音声、音楽などをまとめて表示・再生するためのアプリケーションソフトウェアのことである。CD や DVD、最近では Blu-ray から音楽や動画など再生でき、現在ではインターネットで公開、配信されているものもマルチメディアプレイヤーを使用している。また音楽 CD から曲を PC に取り込んだり、保存したものを CD-R などにコピーする機能もある。しかしこのような行為は個人で楽しむ目的のみ法律で許されており、他人に貸与、譲渡したり販売したりすることはできない。

PC 上で音楽や動画を再生する上で、複数のファイルフォーマットに対応する必要がある。CD の音楽ファイルなど、データ圧縮されてないファイルもあるが、PC 上で扱う音楽や動画像はデータ圧縮されているものが多い。特に動画像においてはデータ無圧縮の場合、ファイルサイズが膨大なものとなる。これらをデータ圧縮する手法にも様々な種類があり、少なくとも一般的に使用されているものは再生できるようにしておく必要がある。また、既存のマルチメディアプレイヤーにはその他にも多くの機能がついていたり、後からプラグインにより拡張できるようにしたものがあり、インターネットを利用したものなどもある（例：ストリーミング配信など)。

代表的なマルチメディアプレイヤーには Microsoft 社の Windows Media Player（図 1）や Apple 社の Quick Time Player などがある。これには、先に触れたように、プラグイン拡張など多くの機能がある。

図 1　Windows Media Player

3. ファイルフォーマットについて

ファイルフォーマットとはファイル保存形式のとこである。ファイル形式はコンテナとコーデックで規定されており、コンテナには Avi 形式や Mp4 形式などがある。簡単な言葉にすると、動画の「入れ物」と言える。コンテナには音声や字幕情報なども入れることができる。

コーデックは音声や映像の符号化や復号化のアルゴリズムやそれを行うためのプログラムである。動画像は、データ圧縮されていない動画をコーデックによって圧縮してからコンテナへと格納するようになっている。コンテナにどのようなものを入れることができるかは、コンテナの種類によって決まる。例えば Avi 形式であれば、映像は WMV、DivX、H.264 などがあり、音声では WMA、MP3 など様々なコーデックに対応して格納することができる。

4.　再生方法について

動画を再生する方法には様々なものがあるか、ここでは DirectShow を使用した。DirectShow とはマルチメディア拡張 API 群である DirectX に含まれる、API のひとつである。DirectShow を使うための手順としてはまずフィルタグラフを作成し、次にフィルタグラフをレンダリングする。最後にビデオサイズなどを調整して動画像を出力するようになっている。

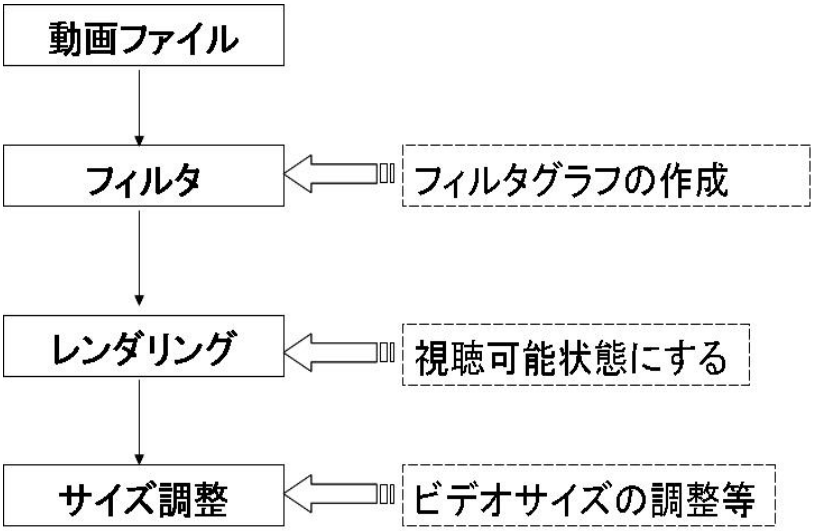

図 1　出力手順

4.1. フィルタグラフについて

DirectShow を用い音楽や動画像を再生するにあたって、まずフィルタグラフと呼ばれるものを作成する。フィルタは入力されたデータを処理し出力する。DirectShow ではこのフィルタを細かく分けている。フィルタ同士はピンと呼ばれるインターフェースでつながれ、つなぎ合わせてできたものをフィルタグラフと呼ぶ。DirectShow のフィルタは、主にソースフィルタ、変換フィルタ、レンダラの三つに分類される。ソースフィルタはデータを作成し、次のフィルタに送り込む。変換フィルタはデータを受け取って転送する。レンダラはデータの受け取りをして表示を行う。

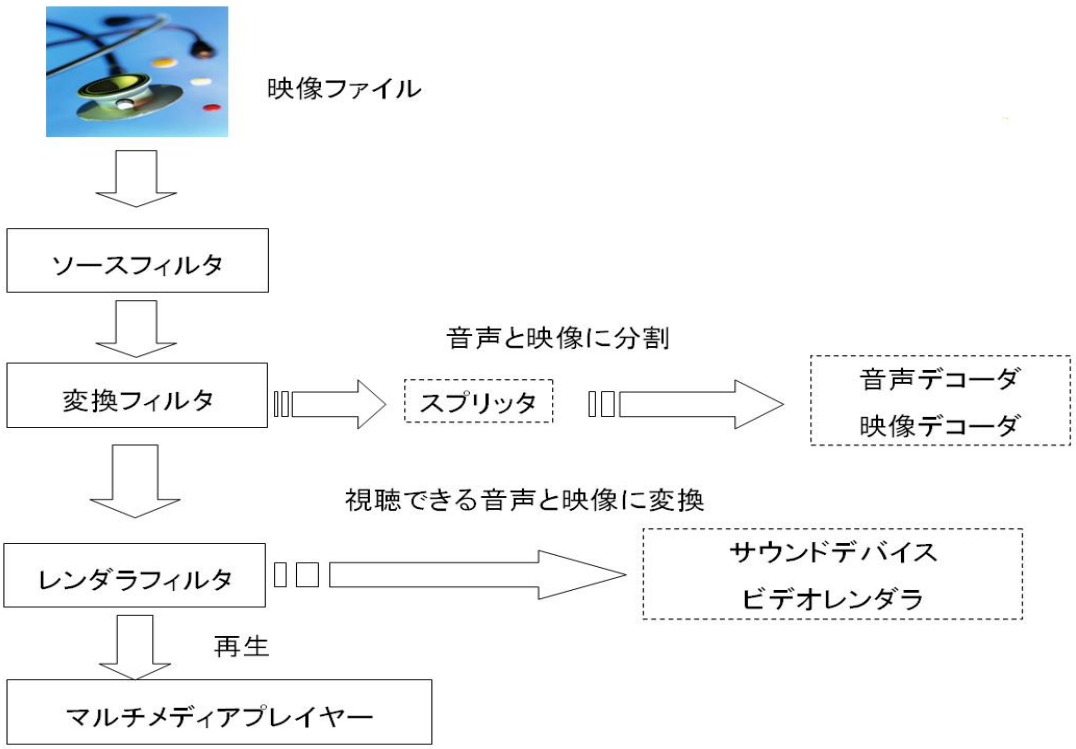

図２　フィルタグラフ内の構造

4.2. レンダリングについて

レンダリングとは、コンピュータにより各種フォーマットのデータを処理し、画像、音声、映像を作り出すことである。これらを行うソフトウェアなどをレンダラという。

4.3. フィルタグラフ内の処理

実際に動画を再生する場合、まずソースフィルタで HDD などの記憶装置からデータを読み込み変換フィルタへ流す（図２）。ソースフィルタはグラフの最初に位置することとなる。次に変換フィルタにより、映像・音声がそれぞれ別にデコード処理され、レンダリングフィルタへと送られる。最後にレンダリングフィルタによりデータを視聴可能な実際の映像、音声へと変換する。

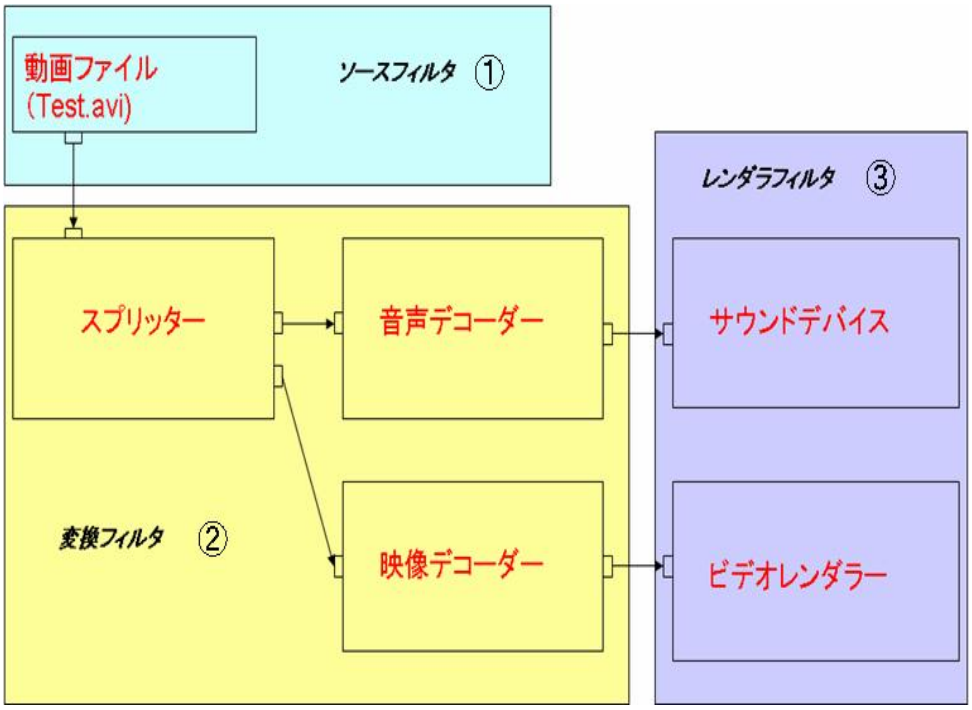

図３　フィルタグラフ内の処理

5. Directshow インターフェース

プレイヤーの主な機能として再生、停止および一時停止などがある。これらの操作をできるようにする方法のなかにソフトウェアインターフェースを使う方法がある。以下のソフトウェアインターフェースにより操作やビデオ出力などをしている。

（1）IGraphBuilder　(付録5、7参照)
　ビデオレンダリングや出力などの機能

（2）IMediaControl　(付録1、5、7参照)
　再生、停止、一時停止などの操作する機能

（3）IVideoWindow　(付録4、5、7参照)
　ビデオファイルの大きさなどの機能(フルスクリーン機能など)

（4）IBasicAudio　(付録2、5、7参照)
　音量の取得と音量調整の機能

（5）IMediaPosition　(付録1、2、3、5、7参照)
　ストリーム内のシークする機能

以上が Directshow で使用しているインターフェースである。

6. 作成したプレイヤーの機能

作成にはVisualC++として、VisualStadio2008を使用した。OSの動作確認はVista、Xpで確認した。今回作成したメディアプレイヤーの機能には、音楽、ビデオファイルの再生、音量の調整、シークバーによる再生位置の変更などがある。他にもドラッグ＆ドロップや再生リストを追加して扱いやすくした。フルスクリーン切り替えのためのキーボードショートカット機能も実装している。作成した機能を以下に列記する（図４）。

（１）動画、音楽の再生

オープンファイルダイアログまたはドラッグ＆ドロップによりファイルを選ぶ。選んだファイル名を取得し、DirectShowの各種フィルタを通して再生する。

（２）シークバー

動画を選んだときに動画の再生時間を取得し、シークバーの最大値とする。現在時間はタイマーの機能を使用し、一定時間ごとに再生している時間を取得してバーを動かす。シークバーを手動で動かした場合は動かした量を秒に換算しその時間へと再生場所を移動させる。

（３）音量の調整

シークバーと違い、音量は最大値と最小値は決まっているので、手動でバーを動かしたときにその割合を音量として表示する。

（４）ドラッグ＆ドロップ

ファイルをドラッグした場合、指定されたウィンドウの上に置かれた状態であるかどうかをまず判断する。その後ドロップされたファイル名を読み取りリストボックスへと送り再生する。指定されたウィンドウ枠内であればどこでも読み取ることができる。

（５）繰り返し再生

チェックが入っている場合はシークバーが最後まで行ったとき、時間が０へと戻って再生される。チェックが入ってない場合は停止する。

（６）再生リスト

ドラッグ＆ドロップでリストに追加される。現在再生されている動画のシークバーが最後まで行った場合、その動画の再生が終わったと判断され、再生リストにある次のファイルの再生へと移る。動画の再生中に再生リストにあるファイル名をダブルクリックすることで、再生している動画を停止し、選んだ動画を再生する。繰り返しにチェックが入っている場合は次に行かず繰り返す。

（7）フルスクリーン

再生中の動画を全画面で表示する。フルスクリーン中では再生されているウィンドウが別のものとなるので、フルスクリーンから元に戻す場合は、フルスクリーン時に入力された情報はフルスクリーンに変わる前のフォームへと送られるようになっている。切り替えはキーボードショートカット Ctrl＋Enter、解除は Esc キーにて行う。他にもメニュー→表示→フルスクリーン切り替えからでもフルスクリーンに切り替えるようになっている。この時も解除は Esc キーで行う。

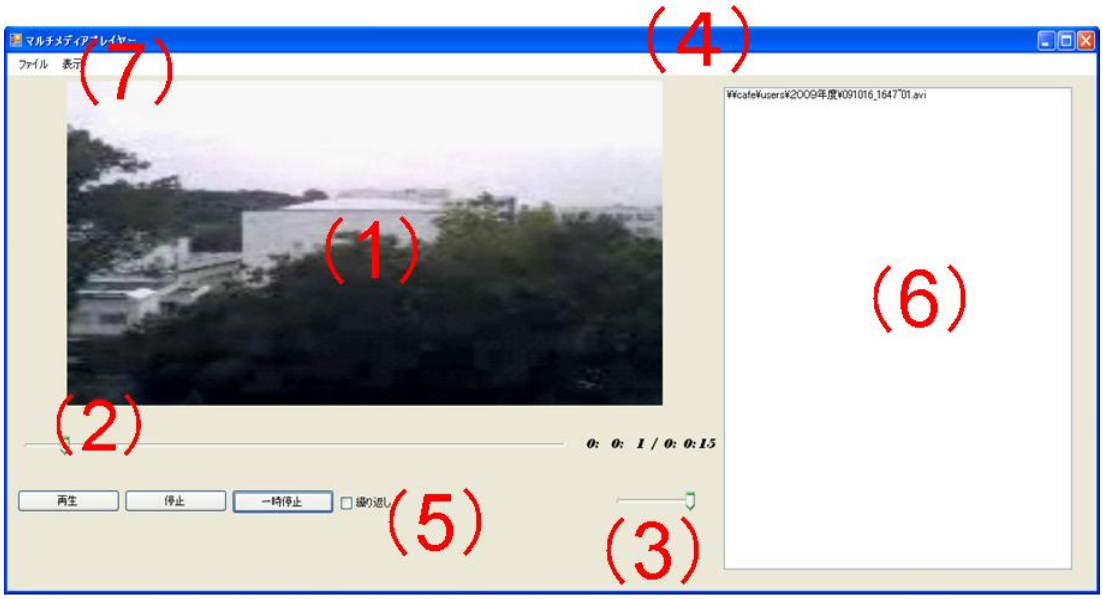

図４　作成した機能

7. コーデックの拡張について

対応しているコーデックは Mpeg - 1、MP3、Windows Media Audio、Windows Media Video、MIDI にコンテナフォーマットは Avi、Asf、Wav に対応している。最近になって良く使われている MP4 などは Directshow に対応しているコーデックをインストールすることにより Directshow の機能を用いて自動に再生できるようになっている。

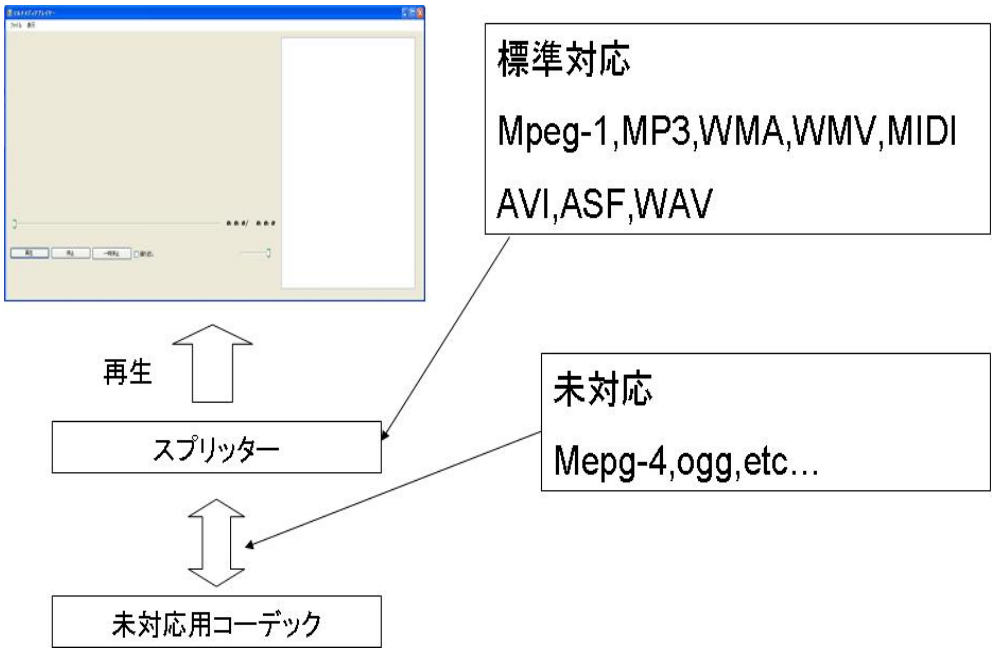

図５ コーデックの拡張

8. 一般的なプレイヤーとの比較

今回作成したプレイヤーを一般的に使われている Windows Media Player を基準にして、比較した。その結果、聴覚と視覚による主観評価では音質、画質ともに差はなく同等と言えることがわかった。再生機能以外の機能としては繰り返し再生、音量調節、ファイル選択、再生リスト、フルスクリーン表示切り替えなどがあるが必要な機能はついていると考えられる（図６）。ただし Windows Media Player のような視覚エフェクトやデータの取り込みなどの機能は実装していない。

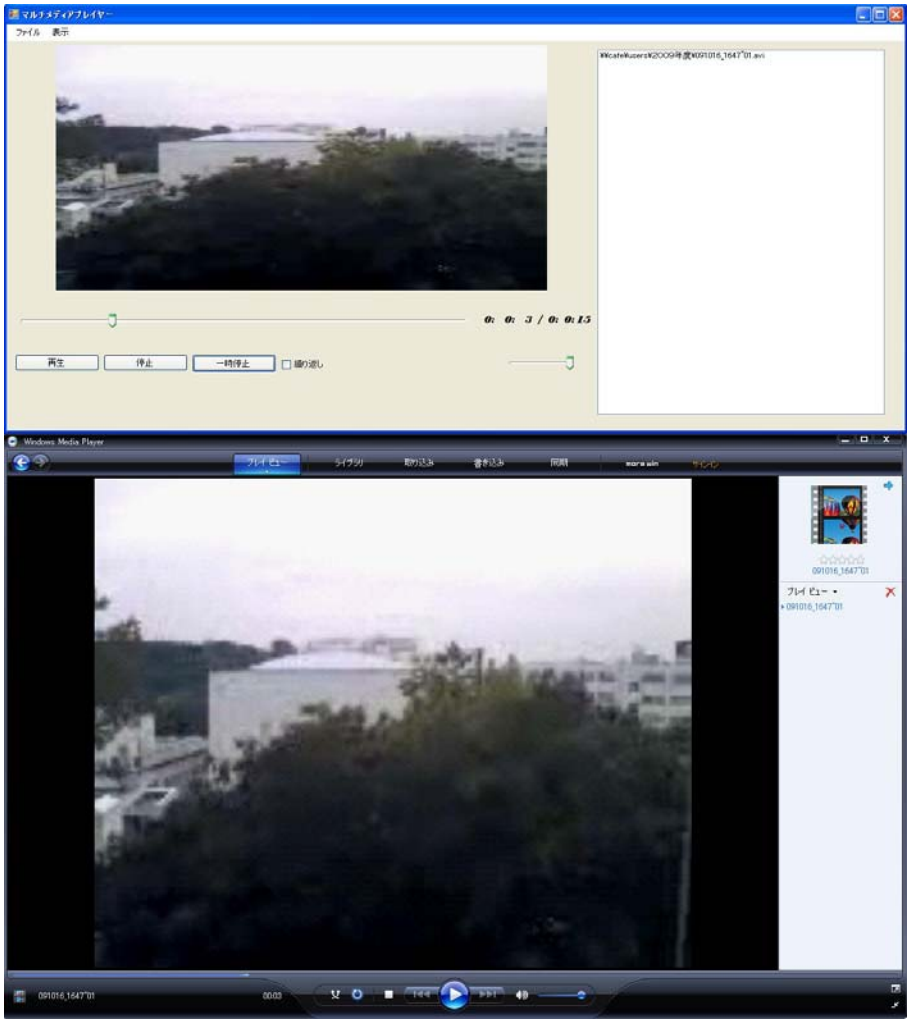

図６　作成したプレイヤーとの比較

9. まとめ

このマルチメディアプレイヤーの研究をとおしてプレイヤーの仕組みや構成などを学習できた。しかし、一般的なプレイヤーと比べると必要と思われる最小限機能しか作成できていないことがわかった。今後の課題として再生リストの保存などが挙げられる。

参考文献

[1]MSDN

http://msdn.microsoft.com/ja-jp/default.asp

付録　プログラムの一部紹介

　ボタンのデザインはVisual Studio2008についているフォームの作成から行っているので付録には載せていません。

付録1. IMediacontrolによるボタン操作について

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                        if(length >0){
                                //再生ボタン
                                pMedia->Run();
                        }
                        }
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
                        if(length >0){
                                //停止ボタン
                                pMedia->Stop();
                                //停止メソッドを出したあと再生位置をに戻す
                                pPosi->put_CurrentPosition((REFTIME) 0);
                        }
                }
private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
                        if(length >0){
                                //一時停止ボタン
                                pMedia->Pause();
                        }
                }
```

2. IMediapositionによるシークバーとIBasicAudioによる音量調整バーについて

```
private: System::Void trackBar1_Scroll(System::Object^  sender, System::EventArgs^  e)
{
                        if(length >0){
                                REFTIME spos;
                                //sposで目盛りの位置の値を取得する
                                spos = trackBar1->Value ;
                                //目盛りの位置で再生を開始する
                                pPosi->put_CurrentPosition(spos);
                        }
```

```
}
private: System::Void trackBar2_Scroll(System::Object^  sender, System::EventArgs^  e)
{
                        if(length >0){
                                //音量トラックバーの位置の数値を取得
                                vol2 = trackBar2->Value;
                                //手に入れた数値の音量を出力する
                                pAudio->put_Volume(vol2);
                        }
}
```

3. シーク関連の部分

```
private: System::Void timer1_Tick(System::Object^  sender, System::EventArgs^  e) {
                        pPosi->get_CurrentPosition(&pos);
                        //シークバーの最大値を設定
                        trackBar1->Maximum = length;
                        trackBar1->Value = pos;
                        //IMediaPositionでのループ再生部分
                        if(length==pos){
                                if(checkBox1->Checked){  //チェックボックスのチェ
ック判定
                                        pPosi->put_CurrentPosition((REFTIME) 0);
                                }else{
                                num = num + 1;
                                        if(num<num2){           //再生リストの終了後
の処理
                                                再生リスト->SelectedIndex = num;
                                                String^ file = 再生リスト
->SelectedItem->ToString();
                                                start(file);
                                        }else{
                                                num = 0;
                                                再生リスト->SelectedIndex = num;
                                                String^ file = 再生リスト
->SelectedItem->ToString();
                                                start(file);
                                        }
```

```
                        }
                }
                //現在の再生時間の表示
                int poss;
                poss = pos; //double型をint型に変換
                hou2 = poss / 3600; //時間
                hour2 = poss % 3600; // 時間の余り
                min2 = hour2 / 60; //分
                sec2 = hour2 % 60; // 秒
                char    time2[100];
                String^ time3 = "0";
                sprintf_s(time2,"%2d: %2d: %2d /",hou2,min2,sec2);
                time3 = gcnew String(time2); // char[] を引数にしてstring を
new するとOK
                label2 -> Text = time3;
        }
```

4. IVideoWindow によるフルスクリーン切り替えについて

```
private: System::Void フルスクリーンToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
                //フルスクリーンにする
                if(length >0){
                        pVideo->put_FullScreenMode(OATRUE);
                }
        }
```

5. Directshow のインターフェースによるファイル選択時の処理ついて

ここではインターフェースの取得・レンダリング・総再生時間の取得などを行う部分である。

```
public: void start(String^ filename1){

                pin_ptr<const wchar_t> file1 = PtrToStringChars(filename1);

                // COMライブラリの初期化
                hRslt = CoInitialize( NULL );

                // フィルタグラフマネージャの生成
```

```
                        CoCreateInstance( CLSID_FilterGraph, NULL, CLSCTX_INPROC,
IID_IGraphBuilder, (LPVOID *)&pGrph );

                        // IVideoWindowインターフェースを取得する
                        hRslt = pGrph->QueryInterface(IID_IVideoWindow, (void
**)&pVideo);

                        //IMediaControlインターフェースを取得する
                        hRslt = pGrph->QueryInterface( IID_IMediaControl, (void
**)&pMedia );

                        //IBasicAudioインターフェースを取得する
                        hRslt = pGrph->QueryInterface( IID_IBasicAudio, (void
**)&pAudio );

                        //IIMediaPositionインターフェースを取得する
                        hRslt = pGrph->QueryInterface( IID_IMediaPosition, (void
**)&pPosi );

                        //IIMediaEventインターフェースを取得する
                         hRslt = pGrph->QueryInterface( IID_IMediaEvent, (void
**)&pEvent );

                        //IBasicVideoインターフェースを取得する
                        hRslt = pGrph->QueryInterface(IID_IBasicVideo, (void
**)&pBVideo);

                        //IMediaSeekingインターフェースを取得する
                        hRslt = pGrph->QueryInterface(IID_IMediaSeeking, (void
**)&pSeek);


                        // 再生するファイルを指定して、フィルタグラフを構築する
                        pGrph->RenderFile(file1,NULL);

                        pVideo->put_Owner(hwnd);
```

```
            pVideo->put_WindowStyle(WS_CHILD | WS_CLIPCHILDREN);
            pVideo->SetWindowPosition(70,30,700,350);

            pVideo->put_MessageDrain(hwnd);


            //音量の取得
            pAudio->get_Volume(&vol);

            //音量のトラックバーに取得した音量を送る
            trackBar2->Value = vol;

            //再生時間の取得
            pPosi->get_Duration(&length); //総時間の取得
            len = length; //double型をint型に変換
            hou = len / 3600; //時間
            hour = len % 3600; // 時間の余り
            min = hour / 60; //分
            sec = hour % 60; // 秒

            //総再生再生時間の表示
            char    time[100];
            String^ time1 = "0";
            sprintf_s(time,"%2d:%2d:%2d",hou,min,sec);
            time1 = gcnew String(time); // char[] を引数にしてstring を
new するとOK
            label1 -> Text = time1;

            timer1->Enabled = "True";

            // ストリームの再生（グラフの実行）
            pMedia->Run();

        }
```

6. ドラッグ＆ドロップについて

```
//ドラッグしたままカーソルがウィンドウ上に置かれたとき
private: System::Void Form1_DragEnter(System::Object^  sender,
System::Windows::Forms::DragEventArgs^  e) {
                        if(e->Data->GetDataPresent(DataFormats::FileDrop))
                                e->Effect = DragDropEffects::All;
                                  else
                                          e->Effect = DragDropEffects::None;
                }
//ファイルをドロップされたとき
private: System::Void Form1_DragDrop(System::Object^  sender,
System::Windows::Forms::DragEventArgs^  e){
                                          //ファイル名の取得
                                          array<String^>^ files=
static_cast<array<String^>^>(e->Data->GetData(DataFormats::FileDrop, true));
                                          String^ fname = files[0];
                                          //再生リストへ入れる
                                          再生リスト->Items->Add(fname);
                                          num2 =num2+1;
                                          //再生しているものがなかったら再生
                                          if(length == 0){
                                                  start(fname);
                                          }

                }
```

7. 再生リストのダブルクリックしたときの処理について

```
private: System::Void 再生リスト_DoubleClick(System::Object^  sender,
System::EventArgs^  e) {
                        if(length >0){  //現在再生されているものがあるかチェック
                                //現在再生されているものを終了する
                                pMedia->Stop();
                                pVideo->put_Visible(OAFALSE);
                                pVideo->put_Owner(NULL);

pEvent->SetNotifyWindow((OAHWND)hwnd,WM_USER+100,0);
```

```
                                pGrph->Release();
                                pGrph=NULL;
                                pVideo->Release();
                                pVideo=NULL;
                                pMedia->Release();
                                pMedia=NULL;
                                pAudio->Release();
                                pAudio=NULL;
                                pPosi->Release();
                                pPosi=NULL;
                                pEvent->Release();
                                pEvent=NULL;
                                pBVideo->Release();
                                pBVideo=NULL;
                                pSeek->Release();
                                pSeek=NULL;

                                String^ file = 再生リスト
->SelectedItem->ToString();
                                //選んだ再生リストの番号を更新
                                num = 再生リスト->SelectedIndex;
                                //再生
                                start(file);
                        }
                }
```