

Re: Displaying WMV from a stream with the WM ASF Reader Filter in a filter graph

Source: <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2005-01/0274.html>

From: Bob Dellaca (*bobdlxyzy_at_xtra.co.nz*)

Date: 01/13/05

Date: Fri, 14 Jan 2005 11:30:30 +1300

Here is the source code of a sample unit to play an OGG file from a stream (here a TStream (TFileStream)).

It uses DS Pack 2.3.4 and the DS filters from oggDS0995.exe The filter to handle the input stream is derived from the Async Filter Sample in the DirectShow SDK (as translated in DS Pack). Setting MEDIASUBTYPE_ogg should result in the filter graph manager rendering

source filter -> ogg splitter -> ogg decoder -> sound renderer

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Menus, ActiveX, DirectShow9, BaseClass, UAsyncIO, UAsyncRdr;

type

TBCFileStream = class(TBCAsyncStream)

private

Flock: TBCCritSec;

FLength: LONGLONG;

FPosition: LONGLONG;

FStream: TStream;

public

constructor Create(Stream: TStream);

destructor Destroy; override;

function SetPointer(APos: LONGLONG): HRESULT; override;

function Read(ABuffer: PByte; ABytesToRead: DWord;

AAAlign: Boolean; out ABytesRead: DWord):

HRESULT; override;

function Size(out ASizeAvailable: LONGLONG): LONGLONG; override;

function Alignment: DWord; override;

```
    procedure Lock; override;
    procedure Unlock; override;
end;

TBCFileReader = class(TBCAsyncReader)
public
    function Register: HRESULT; override; stdcall;
    function UnRegister: HRESULT; override; stdcall;
    constructor Create(AStream: TBCFileStream; Amt: PAMMediaType;
        out hr: HRESULT);
end;
```

```
TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Openoggfile1: TMenuItem;
    OpenFileDialog1: TOpenDialog;
    Exit1: TMenuItem;
    procedure Openoggfile1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Exit1Click(Sender: TObject);
private
    Stream: TBCFileStream;
    Reader: TBCFileReader;
    pFilgraphManager: IMediaControl;
    pGraph: IGraphBuilder;
    pStream: TFileStream;
    procedure CloseIt;
public
end;
```

```
var
    Form1: TForm1;
```

implementation

```
{ $R *.dfm }
```

```
constructor TBCFileStream.Create(Stream: TStream);
begin
    FStream := Stream;
    FLength := FStream.Size;
    FPosition := 0;
    FLock := TBCCritSec.Create;
    inherited Create;
end;
```

```
destructor TBCFileStream.Destroy;
begin
    if Assigned(FLock) then
        FreeAndNil(FLock);
end;
```

```
    inherited Destroy;
end;

function TBCFileStream.SetPointer(APos: LONGLONG): HRESULT;
begin
    if (APos < 0) or (APos > FLength) then
        Result := S_FALSE else
    begin
        FPosition := APos;
        FStream.Position := APos;
        Result := S_OK;
    end;
end;

function TBCFileStream.Read(ABuffer: PByte; ABytesToRead: DWord;
    AAlign: Boolean;
    out ABytesRead: DWord): HRESULT;
var
    ReadLength: DWord;
begin
    FLock.Lock;
    try
        if (FPosition + ABytesToRead > FLength) then
            ReadLength := FLength - FPosition else
            ReadLength := ABytesToRead;
        FStream.Read(ABuffer^, ReadLength);
        Inc(FPosition, ReadLength);
        ABytesRead := ReadLength;
        Result := S_OK;
    finally
        FLock.Unlock;
    end;
end;

function TBCFileStream.Size(out ASizeAvailable: LONGLONG): LONGLONG;
begin
    ASizeAvailable := FLength;
    Result := FLength;
end;

function TBCFileStream.Alignment: DWord;
begin
    Result := 1;
end;

procedure TBCFileStream.Lock;
begin
    FLock.Lock;
end;
```

```
procedure TBCFileStream.Unlock;
begin
  FLock.UnLock;
end;

constructor TBCFileReader.Create(AStream: TBCFileStream;
                                Amt: PAMMediaType;
                                out hr: HRESULT);
begin
  inherited Create('File reader', nil, AStream, hr);
  CopyMemory(@Fmt, Amt, SizeOf(TAMMediaType));
end;

function TBCFileReader.Register: HRESULT;
begin
  Result := S_OK;
end;

function TBCFileReader.UnRegister: HRESULT;
begin
  Result := S_OK;
end;

procedure TForm1.Openoggfile1Click(Sender: TObject);
const
  MEDIASUBTYPE_ogg: TGUID = '{D2855FA9-61A7-4DB0-B979-71F297C17A04}';
var
  mt: TAMMediaType;
  pmt: PAMMediaType;
  hr: HRESULT;
  Pin: IPin;
begin
  if OpenFileDialog1.Execute then
  begin
    CloseIt;
    CoCreateInstance(CLSID_FilterGraph, nil, CLSCTX_INPROC_SERVER,
                    IID_IFilterGraph2, pGraph);
    pFiltergraphManager := pGraph as IMediaControl;
    pStream := TFileStream.Create(OpenDialog1.FileName, fmOpenRead);
    try
      Stream := TBCFileStream.Create(pStream);
      pmt := @mt;
      TBCMediaType(pmt).InitMediaType;
      mt.majortype := MEDIATYPE_Stream;
      mt.subtype := MEDIASUBTYPE_ogg;
      Reader := TBCFileReader.Create(Stream, @mt, hr);
      Reader._AddRef;
      pGraph.AddFilter(Reader, nil);
      Pin := Reader.GetPin(0);
      pGraph.Render(Pin);
    except

```

```
    on E:Exception do
    begin
        MessageDlg(E.Message, mtError, [mbOk], 0);
        pGraph := nil;
        CloseIt;
        Exit;
    end;
end;
pFilgraphManager.Run;
end;
end;
```

```
procedure TForm1.CloseIt;
begin
    if pGraph <> nil then
    begin
        pFilgraphManager.Stop;
        pGraph := nil;
    end;
    pFilgraphManager := nil;
    Reader := nil;
    FreeAndNil(Stream);
    FreeAndNil(pStream);
end;
```

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    CloseIt;
end;
```

```
procedure TForm1.Exit1Click(Sender: TObject);
begin
    Close;
end;
```

```
end.
```