Iraqi Journal of Statistical Science (10) 2006
P.P. [38-53]

The Active Movie (DirectShow Programming)

Aseel Waleed Al-Niamey[*] Jamaal Salah Aldeen[**] Hanan Hamed Ali[***]

ABSTRACT

The idea of this project depends on programming the functions of DirectX under the environment of the windows, which will give a very large possibility to facilitate and speed up the films, games, animations, and manage the messages.

In general, the graphics and the pictures need a lot of computation power, and the Windows suffered from the low speed as a result of dealing with the buffers of the memory of data to be displayed instead of dealing with the data itself, for this reasons DirectX being the solution for this problem, because DirectX directly deal with the buffers which contain data. For the above reasons it is necessary to develop programs that can work as an interface between DirectX and the windows.

The work focuses on programming the DirectShow function (formally called ActiveMovie), DirectShow enables the applications play files and streams from various sources, including local files and remote files on network. DirectShow has native compressors and decompresses for some file formats.

To achieve the throughput necessary for streaming video and audio, DirectShow uses DirectDraw and DirectSound to render data efficiently to the system's sound and graphic cards. Synchronization is achieved by encapsulating the multimedia data in Time-Stamped media samples.

This work shows the idea of displaying the image in a speed-up and sorted way, which will be considered as a basic form for the animation, display films and playing games by cutting a part from the memory and put the image in it, then change the pointer for it, without needing to transform the image itself. Also DirectShow, shows the displaying of the sounds which is necessary for displaying the films. The Visual C++ language is used to make this work.

[*]Ass. Lecturer/Computer Science Department/College of Computers and Mathematics Science
[**]Lecturer/ Computer Science Department/College of Computers and Mathematics Science
[***] Ass. Lecturer/ Computer Science Department/College of Computers and Mathematics Science

Received: 13/ 9 /2005 _____Accepted: 13/ 6 / 2006

دالة العرض المباشر

الملخص

تعتمد فكرة البحث على برمجة الدوال التي يوفرها Microsoft DirectX تحت بيئة نظام ويندوز والتي توفر إمكانيات كبيرة جداً في تسهيل وتسريع عرض الأفلام والألعاب وتحريك الصور وإرسال الرسائل.

تتطلب الرسومات والصور والأفلام بشكل عام الكثير من الجهد الحسابي لعملها والتي تتطلب السرعة العالية، وحيث ان نظام ويندوز يعاني من البطء نتيجة التعامل مع مقاطع من الذاكرة التي تحوي البيانات التي سوف تعرض وليس مع البيانات مباشرةُ، فكان DirectX هو الحل لهذه المشكلة لانه يتعامل مباشرة مع العتاد والمقاطع التي تحوي البيانات. لهذه الأسباب كان من الضروري عمل برامج تستطيع ربط DirectX مع نظام ويندوز.

يرتكز البحث على برمجة دالة العرض المباشر Direct Show (عادةً تسمى بـ ActiveMovie). الـ DitectShow تمكن التطبيقات من تشغيل الملفات المختلفة سواءاً الموجودة بشكل محلي او بشكل بعيد (كالملفات الموجودة عبر الشبكات) ومن مختلف المصادر. وله القابلية أيضا على كبس او فك الكبس لأنواع مختلفة من الملفات بشكل طبيعي. ولكي يستطيع برنامج العرض المباشر (DirectShow) إنجاز مهامه الضرورية فانه يقوم باستخدام كل من برنامج الرسم (DirectDraw) وبرنامج الصوت (DirectSound) لكي يستطيع تسليم البيانات بشكل كفوء الى كل من أنظمة الصوت وبطاقة الرسم. ويتم التزامن فيه بواسطة تقييد البيانات للأوساط المتعددة بوقت النموذج المحدد للوسط Time-Stamped.

يوضح البحث فكرة عرض الصور بشكل سريع ومتتابع وهو الأساس لعملية تحريك الصور وعرض الأفلام وكذلك الألعاب عن طريق حجز مقاطع من الذاكرة ووضع الصورة ثم تغيير المؤشر الخاص بالعرض وليس بنقل الصورة ذاتها. كما يقدم أيضا عرضاً للصوت الذي يكون ضرورياً في عرض الأفلام.

وكان لابد من توظيف لغة برمجية كفوءة لتأدية مثل هذا العمل فكانت لغة سي المرئية (Visual C++) هي الأنسب.

1- Introduction

DirectX are considered as an architecture developed by Microsoft specifically for multimedia. DirectShow is currently included with Windows98, Windows Me, Windows XP, Windows 2000 and the Internet Explorer.

The Microsoft DirectShow (SDK) provides a hardware independent method for software developers to access services for the rendering, storing and manipulation of video and audio streams originating from local or network based sources. DirectShow allows for compressed and uncompressed input, and a variety of formats (MPEG, AVI, and WAV). Hardware sources should use either VFW (Video for Windows) or WDM (Windows Driver Model) drivers. The DirectShow system is itself very modular, components (which we refer to as filters) are linked together into a "filter graph" (see figure-1). Filter graph is used to modify the video stream [12].

There are three basic kinds of filters: Sources (e.g. device drivers or files), Transform Filters and Renderers.

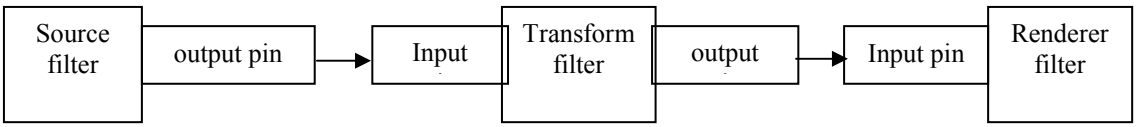


Figure 1: Example of filter graph

Applications control the activities of the filter graph by communicating with the filter graph manager (see figure-2). This can be achieved through the use of ActiveMovie controls or by directly calling a set of COM (Component Object Mod) interfaces. The basic classes of the C/C++ library create the COM interfaces needed for the filters and provide the basic filter frames. The DirectShow SDK is based on various Microsoft services. Microsoft DirectX services are used for video rendering and manipulation if possible. The knowledge required to work with DirectShow depends on the type of filter itself [5].

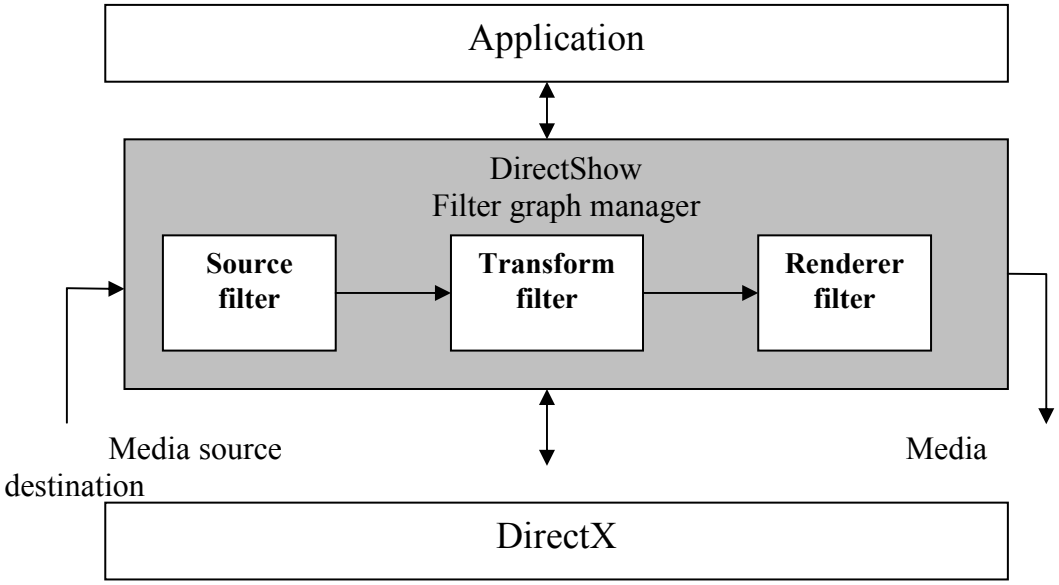


Figure 2: Filter Graph Manager

1.1 An overview on DirectX

Microsoft DirectX is a set of low-level application programming interfaces (APIs) for creating games and other high-performance multimedia applications. It includes support for two-dimensional (2-D) and three-dimensional (3-D) graphics, sound effects and music, input devices, and networked applications such as multi-player games. Microsoft DirectX 9.0 (the newest version) is a major release primarily for graphics. It includes new tools, new features for graphics and Microsoft DirectShow, and enhancements for Microsoft DirectInput and Microsoft DirectPlay [4].

1.2 DirectX major components

One of the main purposes of DirectX is to provide a standard way of accessing many different proprietary hardware devices. The major components of DirectX is as follows (see figure-3) [3]:

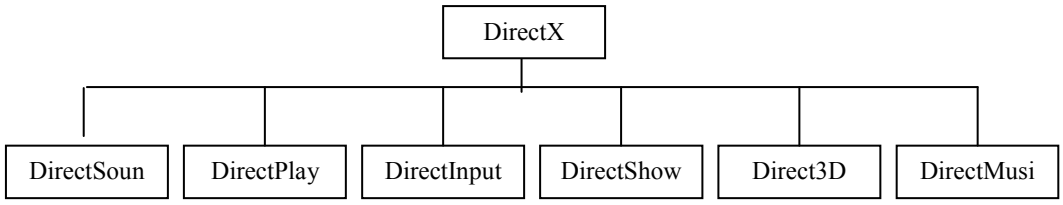


Figure 3: Major component of DirectX

2- DirectShow

DirectShow is one of DirectX components that divide the processing of multimedia tasks such as video playback into a set of steps known as filters. Filters have a number of input and output pins, which connect them together. The generic design of the connection mechanism means that filters can be connected in many different tasks and developers can add their own effects or other filters at any stage in the graph. DirectShow filter graphs are widely used in video playback as well as being used for video and audio recording and editing. Interactive tasks such as DVD navigation are also successfully based on DirectShow [7].

2.1 Component Object Model (COM)

Microsoft DirectShow is based on the Component Object Model (COM). If writing own filter, implement it as a COM object. The DirectShow base classes provide a framework from which to do this. Using the base classes is not required, but it can simplify the development process. COM defined the rules that a component must follow, putting those rules into effect is left for the developer. In DirectShow, all objects drive from a set of C++ base classes. The base class constructors and methods do most of the COM bookkeeping work such as keeping a consistent reference count. Most applications did not need to implement the COM objects, DirectShow provides the components needed [13].

2.2 Filters and Filter Graphs

The building block of Direct Show is a software called a filter. A filter is a software component that performs some operations on multimedia stream; for example, Direct Show filters can do the following:

- Read files.
- Get video from a video captures device.
- Decode various stream formats, such as MPEG-1 video.
- Pass data to the graphics or sound card.

In DirectShow an application performs any task by connecting chains of filters together, so that the output from one filter becomes the input for another. A set of connected filters is called a **filter graph** [8].
For example, (see figure-4) shows a filter graph for playing **AVI file**.

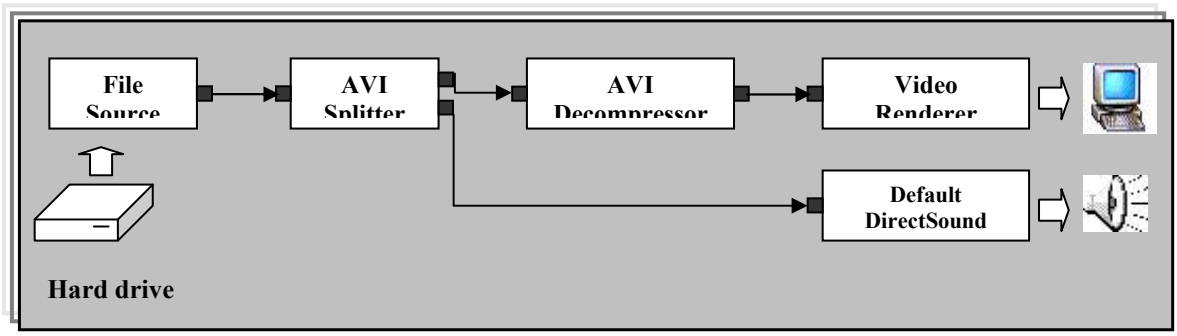


Figure 4 Filter graph for playing AVI

The file source filter reads the AVI file from the hard disk. The AVI splitter filter parses the file into two streams, a compressed video stream and an audio stream. The AVI decompressor filter decodes the video frames. The video renderer filter draws the frames to the display using DirectShow or GDI. The default DirectSound device filters play the audio stream, using DirectSound [14].

The application does not have to manage all of this data flow. Instead, the filters are controlled by high-level component called the *Filter Graph Manager*. The application makes high-level API calls such as "Run" (to move data through the graph) or "Stop" (to stop the flow of data). If they require more control over the stream operations, they can access the filters directly through COM interface. The filter graph manager also passes event notifications to the application.

The filter graph manager serves another purpose as well; it provides methods for the application to build the filter graph by connecting the filters together.

2.3 Filter Graph Manager

The filter graph manager is a COM object controls the filters in a filter graph. It performs many functions including the following [2]:

- Coordinating state changes among the filters:

State changes within filters must occur in a particular order. Therefore, the application does not issue state-change commands directly to the filters. Instead it gives a single command to the filter graph manager, which distributes the command to each of the filters. Seeking works in similar fashion, the application gives a seek command to the filter graph manager, which distributes it to the filters [10].

- Establishing a reference clock:

All of the filters in the graph use the same clock, called a *reference clock.* The reference clock ensures that all the streams are synchronized. The time at which a video frame or audio sample should be renderered is called the *presentation time.* The presentation time is measured relative to the reference clock. The filter graph manager chooses a reference clock, usually either the clock on the sound card or the system clock [9].

3- DirectShow programming

The application needed to display a dialog box on a start-up therefore the programmer should open a file, then the selected file automatically begins playing.

If the media contains video, the player should be able to resize its client area to the video's preferred size. If the media contains no video, only audio then the player will display a small default window.

3.1 Design steps:

At first any DirectShow application always performs the same basic steps or tasks:

1- Creates an instance of the Filter Graph Manager (see figure-5).
2- Uses the Filter Graph Manager to build a filter graph (see figure-6).
3- Runs the graph, which causes data to move through the filters (see figure-7).

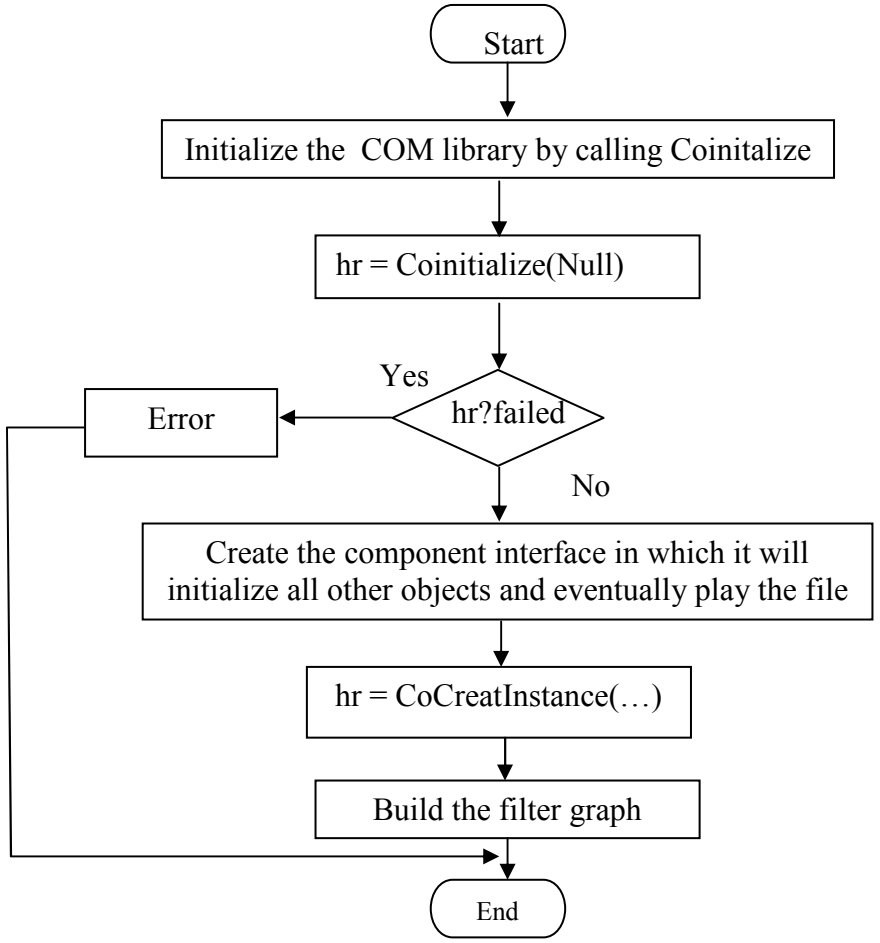


Figure 5: Create the Filter Graph Manager

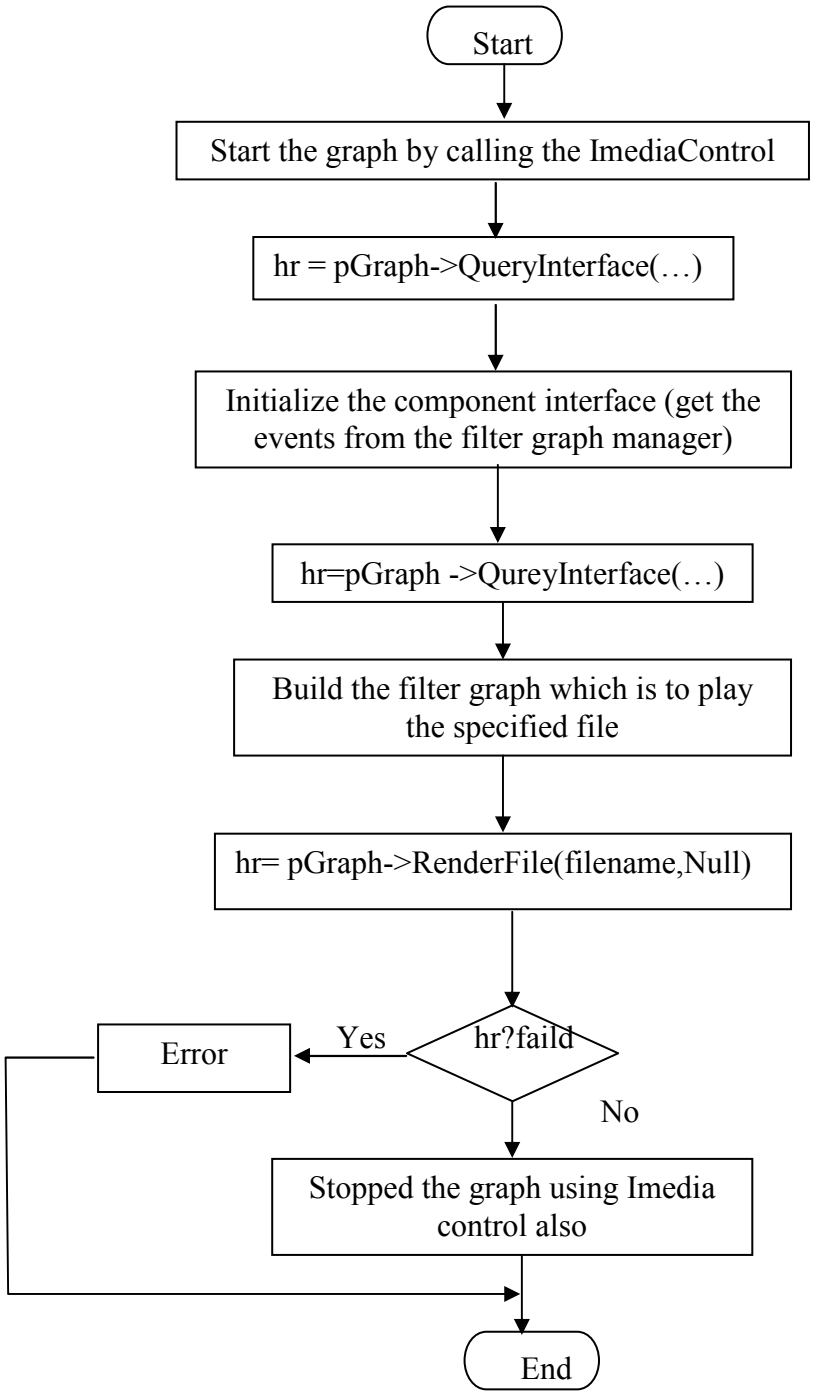


Figure 6: Filter Graph building

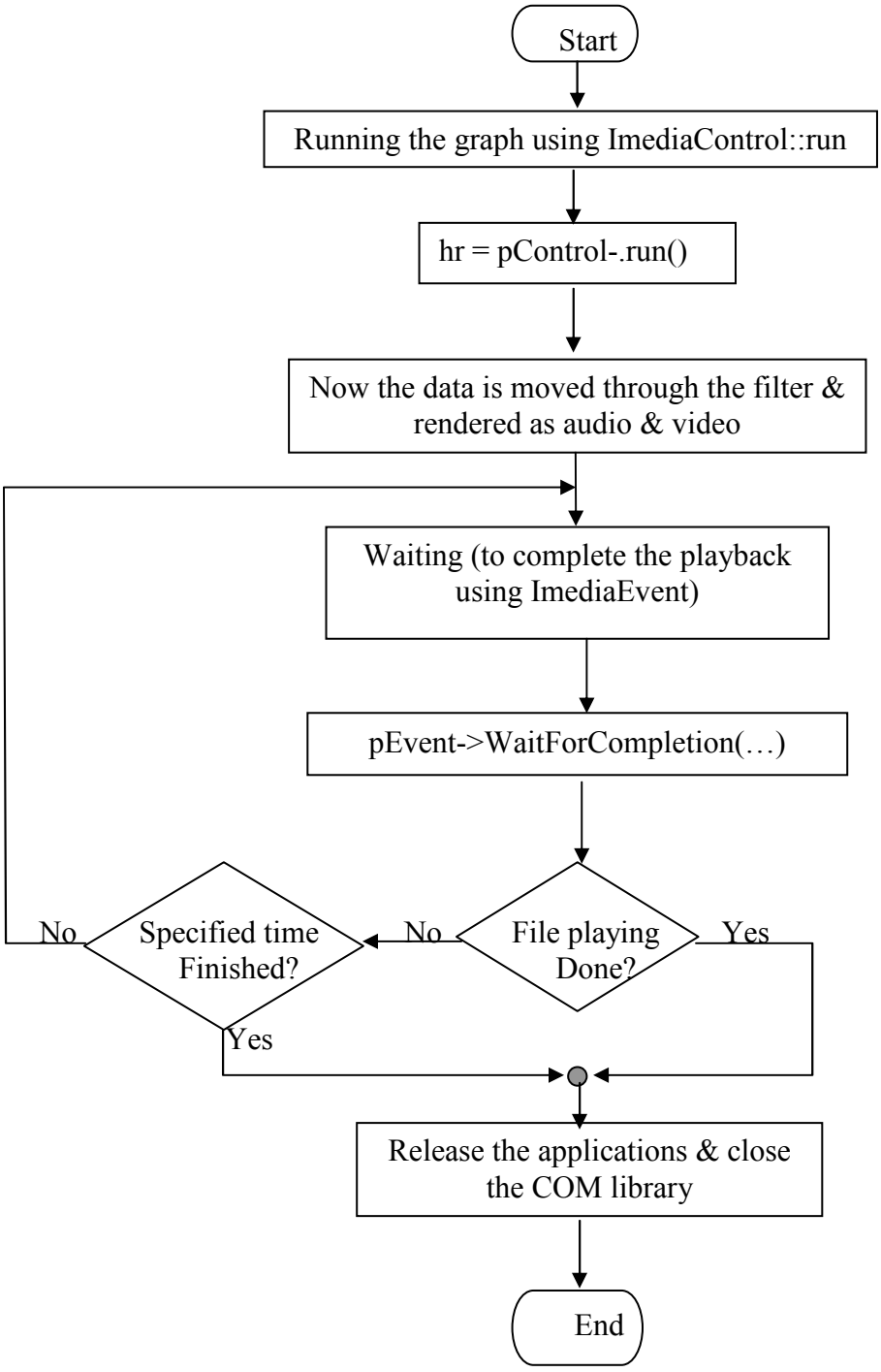


Figure 7: Running the graph

Some definitions of the program due to DirectShow:

HRESULT	Used to return the result of a function to know if it is accepted or rejected and also return the kind of the error if it is found.
hr	A variable of kind HRESULT.
::	The first two points mean output the current class, while the second two points means inside the current class.
IgraphBuilder	An interface that contains methods for building the filter.
*pGraph	Variable used as a pointer to IgraphBuilder.
ImediaControl	Contains methods for stopping & starting the graph
*pControl	Variable used as a pointer to ImediaControl.
ImediaEvent	Contains methods for getting events from the filter graph manager.
*pEvent	Variable used as a pointer to ImediaEvent.
CoInitialize	A function used to initialize the COM library.
CoCreatInstance	A function used to creat the filter graph manager.
pGragh ->RenderFile(file name,reserved)	Method to build a filter graph which can play the specified file

The first step is calling **CoInitialize** to initialize the COM library:

```
HRESULT hr = CoInitialize(Null);
 If (FAILED(hr))
 {
    // error-handling code.
 }
```

where HRESULT is a function that returns a value of accept or reject, hr is any variable of kind HRESULT.

To keep things simple, we will ignore the return value, but we should always check the **HERESULT** value from any method call.
Next, we will call **CoCreatInstance** to create the filter graph manager:

```
IgraphBuilder *pGraph;
HERSULT hr=CocreatInstance (CLSID_FilterGraph, NULL,
                            CLSCTX_INPROC_SERVER,
IID_IgraphBuilder,
                            (void **) &pGraph);
```

As shown, the class identifier (CLSID) is CLSID_FilterGraph. The filter graph manager is provided by an in-process DLL, so the execution context is CLSCTX_INPROC_SERVER. DirectShow supports the free-threading model, so we can also call **ColnitializeEx** with the COINIT_MULTITHREADED flag.

The call to CoCreateInstance returns the **IgraphBuilder** interface, which mostly contains methods for building the filter graph. Another interfaces are needed:

- **ImediaControl** controls streaming. It contains methods for stopping and starting the graph.
- **ImediaEvent** has methods for getting events from the filter graph manager. In this example, the interface is used to wait for playback to complete.

Both of these interfaces are exposed by the filter graph manager. Use the returned IgraphBuilder pointer to query for them:

```
IMediaControl *pControl;
ImediaEvent *pEvent;
hr=pGraph->QueryInterface(IID_ImediaControl,(void**)&pControl);
hr=pGraph -> QueryInterface (IID_ImediaEvent, (void **) &pEvent);
```

Now we can build the filter graph. For file playback, this is done by a single method call:

```
hr = pGraph ->RenderFile (L"C:\\Example.avi", NULL);
```

The **IgraphBuilder::RenderFile** method builds a filter graph that can play the specified file. The first parameter is the file name, represented as a wide character (2-byte) string. The second parameter is reserved and must equal NULL.

This method can fail if the specified file does not exist, or the file format is not recognized. Assuming that the method succeeds, however, the filter graph is now ready for playback. To run the graph, we will call the **ImediaControl::Run** method:

```
hr= pControl ->Run( );
```

When the filter graph runs, data move through the filters and rendered as video and audio. Playback occurs on a separate thread. We can wait for playback to complete by calling the **IMediaEvent::WaitForCompletion** method:

```
long evCode = 0;
pEvent -> WaitForCompletion (INFINITE, &evCode);
```

This method blocks until the file is done playing, or until the specified time-out interval elapses. The value INFINITE means the application blocks indefinitely until the file is done playing.
When we finished our application, we should release the interface pointers and close the COM library:

```
pControl ->Relealse ( );
pEvent -> Relealse ( );
pGraph -> Relealse ( );
CoUninitialize ( );
```

These are the basics that we use in building our DirectShow application that we talk about its requirements earlier.

4- Testing and Execution

Before anything it is notable that for testing our program, DirectX-9 must be installed on the system.

When testing the quality of the sound we found that it is very good, especially by using the best code from ones available on the system. The following table lists the supported codecs, the bandwidth per second (Kbps), and the compression globally unique identifier (GUID) used to select them. The compression GUIDs are defined in Dvoice.h.

Codec	Bandwidth	GUID
Voxware VR12	Variable (1.2 Kbps,avg.)	DPVCTGUID_VR12
Voxware SC03	3.2 Kbps	DPVCTGUID_SC03
Voxware SC06	6.4 Kbps	DPVCTGUID_SC06
TrueSpeech	8 Kbps	DPVCTGUID_TRUE SPEECH
Global System for Mobile Communication (GSM)	13 Kbps	DPVCTGUID_GSM
Microsoft Adaptive Delta Pulse Code Modulation (MS-ADPCM)	32 Kbps	DPVCTGUID_ADPCM
Pulse Code Modulation (PCM)	64 Kbps	DPVCTGUID_NONE

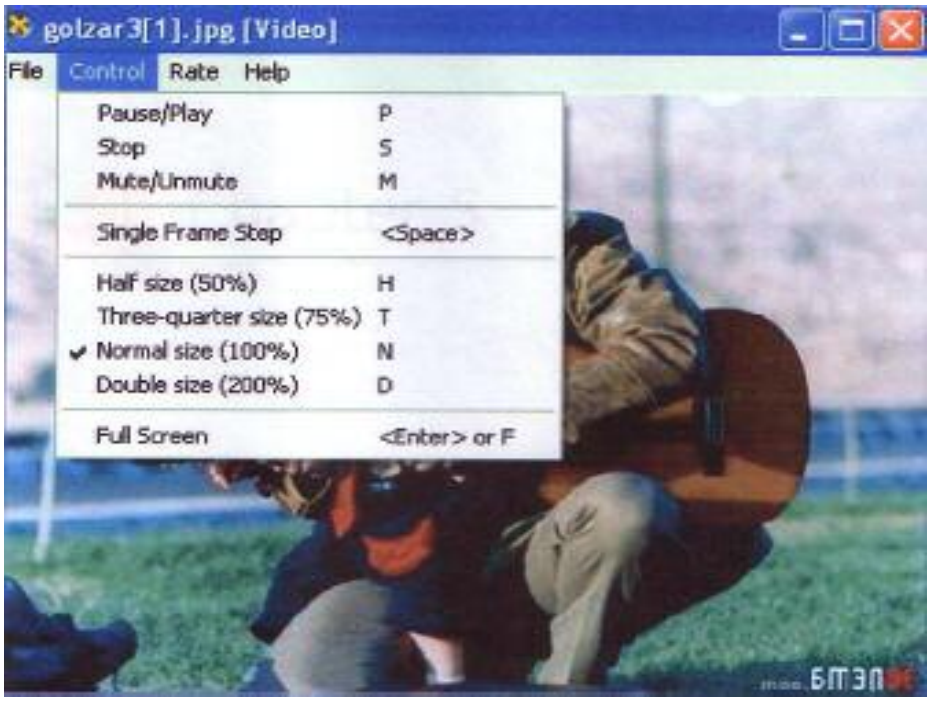
The first three codecs provide a high level of compression and have approximately the same resource demands. On 500 MHz Pentium III class computer, these codecs use approximately 1.5 percent of CPU capacity. The VR12 codec sounds tinny and robotic, but the SC03 and SC06 codecs provide reasonable fidelity. The PCM codec provides the highest sound quality and is essentially uncompressed 8 kHz 16-bit mono-format audio data.

Note that the GSM, ADPCM, and PCM codecs are included with the Microsoft Windows installation but might not have been installed by the user.

The speed of the program was very good, especially after we had processed a heavy graphics on our program.

By using this program the programmer can view the *.avi, *.qt, *.mov, *.mpeg video file, view image files and also play audio files.

In this program the programmer can increase or decrease the speed of playing video file, doubling the size screen, mute the sound of playing file by clicking on the **Control.**



Also we can change the rate of playing video files or audio files by clicking on the **Rate**.

5- Conclusion

We can use DirectX capabilities to build powerful and fast program especially about Drawing, 3D-animations, … etc.
The visual C++, MFC and API of the C++ itself can help for writing all the absent features explicitly.

Another point is about using DirectX_SDK for developing the work. SDK adds a wizard to visual C++ that can build the backbone of the program and then we can expand it.

DirectShow simplifies media playback, format conversion, and capture tasks. At the same time, it provides access to the underlying stream control architecture for applications that require custom solutions.
As an example of the types of applications we can write with DirectShow include the DVD players, Video editing applications, AVI to ASF converters, MP3 players and Digital video capture applications.

7- References

1- C++ Standards Committee. Boost library. http://www.boost.org/, 2004-01-15.website.
2- DirectX programming by Alireza A. Nezhad
3- Microsoft DirectX SDK, January-2000.
4- MSDN (Microsoft Developer Network)-January 2000.
5- Microsoft. DirectShow http://www.microsoft.com/directX/, 2004-01-15. Website.
6- Teach yourself VC++ in 21 days, by Nathan Gurewich & Ori Gurewich.
7-www.codeproject.com http://codeproject.sourceforge.net/, 2004-01-12. Website.
8- www.d-silence.com http://www.d-silence.com/, 2004-01-15. Website.
9- www.ews64.com.
10-www.gcdl.co.uk.
11-WWW.Microsoft.Com/DirectX.
12- www.montivision.com/products/directshow
13-www.programmersheaven.com.
14-WWW.Yaho.Com_Yaho Search *DirectX Books*_inside DirectX by Bradley Bargain and peter Donnelley.