

# Windows Mobile 多媒體應用程式開發

王建興

[qing@cs.nthu.edu.tw](mailto:qing@cs.nthu.edu.tw)

<http://blog.qing.tw>

[http://twitter.com/qing\\_wang](http://twitter.com/qing_wang)

2009/6/18

# 講者簡介

---

## ❑ **Software Development Skills**

- Programming languages: 80x86 assembly, C/C++, Java, C#, ObjC
- Multimedia Programming
- J2EE development and Web programming: EJB, JSP/Servlet
- Network programming: TCP/IP, socket programming
- Object Oriented Design/Programming
- Design Patterns and Software Architecture
- Distributed Network Management System
- Peer-to-Peer Networking

## ❑ **Book Translation**

- Thinking in Java 4nd Edition, in Traditional Chinese
- Thinking in Java 2nd Edition, in Traditional Chinese
- Essential C++, in Traditional Chinese

## ❑ **Honor**

- The champion of the Trend Micro Programming Contest 2004

# Agenda

---

- DirectShow簡介
- 如何開發視訊、音訊的撥放
- 如何利用攝影機捕捉視訊
- 如何利用麥克風捕捉音訊
- 多媒體檔案的儲存
- 結語

# 多媒體應用程式對行動裝置的重要性

---

- 行動裝置儼然成為人們生活不可或缺的重要個人裝置
  - 容易攜帶，計算力強，能連接Internet，有攝影機及麥克風，甚至有GPS/AGPS定位能力
  - 3G網路的普及，頻寬足夠支持多媒體應用
  - 儲存空間充足
- 由於上述提到的各種條件，使得多媒體應用程式在行動裝置上的可能性大增

# 一些簡單的行動多媒體應用

---

- 行動部落格
  - 影片，圖片，聲音
- 網路影音的欣賞
  - YouTube, 無名小站, 土豆網
- 影音分享及傳輸
- 網路廣播，網路電視
- 以住家為中心的行動影音設備
  - 電視，影片
  - 居家監控，保全
- 贓車查驗系統

# 何謂 DirectShow ？

---

- DirectShow 是 DirectX 家族中的一支
- 除了 DirectShow 之外，DirectX 尚包括了
  - DirectPlay
  - DirectDraw
  - Direct3D
  - DirectMusic
- DirectShow 的主要用途在於多媒體資訊的呈現
  - 視訊
  - 音訊

# DirectShow的應用範圍

---

- 多媒體訊息的輸入
  - 麥克風
  - 攝影機
  - 檔案
  - 網路串流
- 多媒體訊息的剖析
- 多媒體訊息的編碼，解碼
- 多媒體訊息的撥放
  - 音訊視訊呈現
- 多媒體訊息的儲存

# 多媒體影音的一些基本的名詞

---

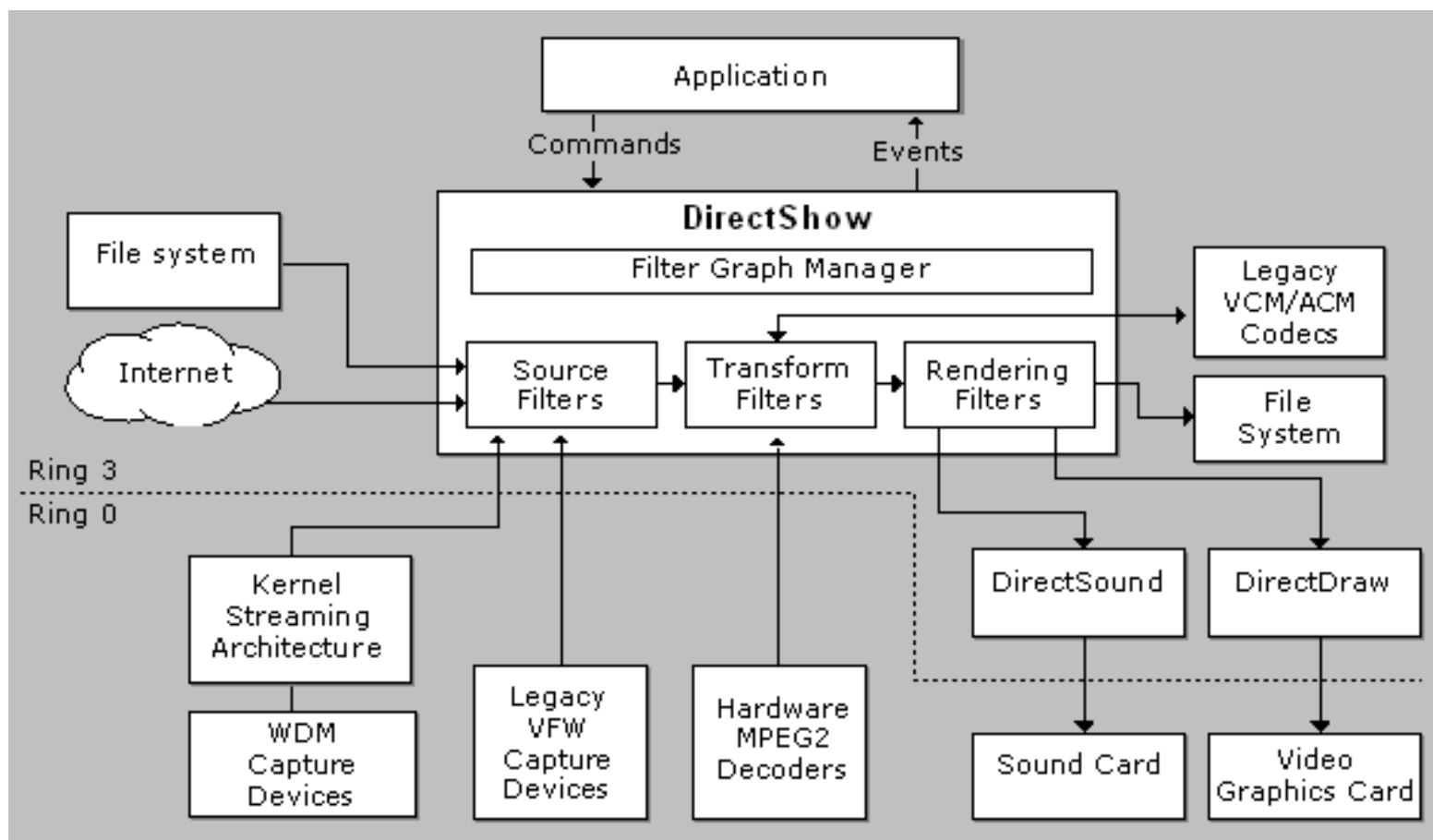
- streaming protocol
  - mms
- mux/demux
  - avi
- codec
  - mp3
  - wma
  - wmv

# 於WM上使用DirectShow，你需要會

---

- C/C++/VB.Net/C#
  - 在本講題中我們將會使用C#示範
- Windows COM元件的使用
  - DirectShow的機制全倚賴COM元件

# DirectShow 架構圖示



# DirectShow架構的優點

---

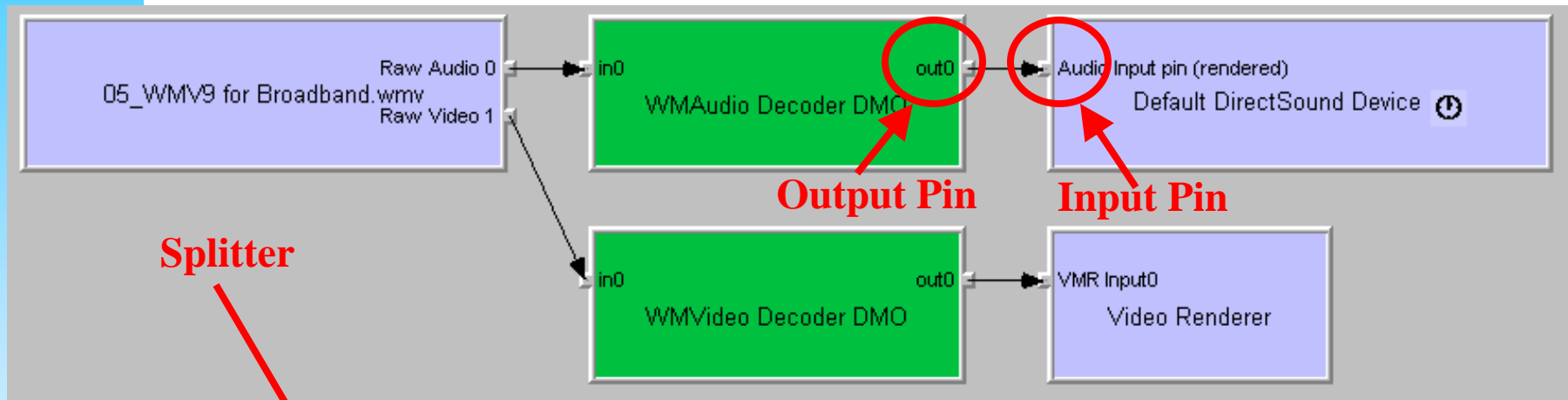
- 元件化，可重用性高
  - 各種用途的元件，都化身為標準的Filter，具備相同的介面，容易搭接各式Filter
- 極具彈性，透過對Filter的客製及組裝，可以達成各式的目的

# DirectShow核心元件

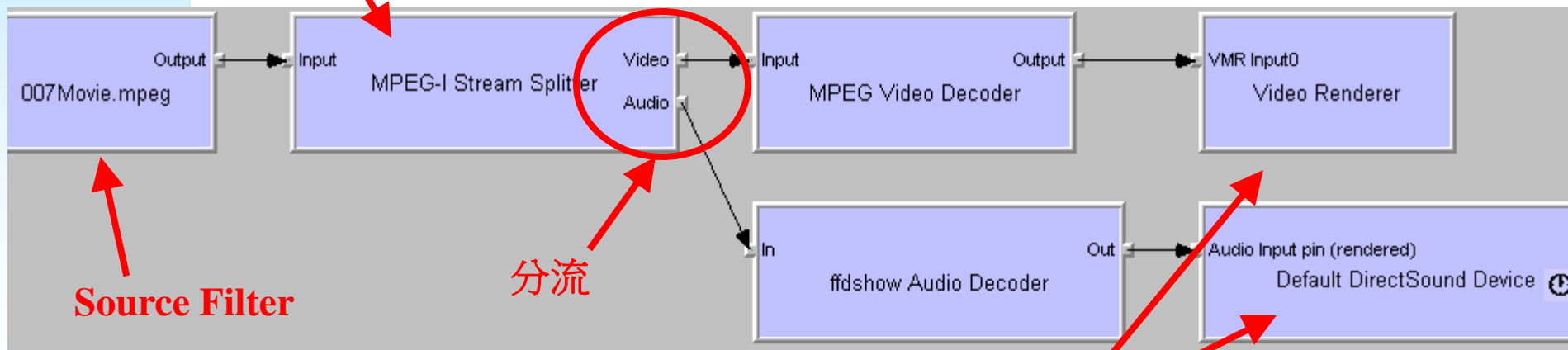
---

- IFilterGraph
  - 每個撥放結構都是透過一個IFilterGraph所描述
- IGraphBuilder
  - 建立、控制IFilterGraph的元件
- IBaseFilter
  - DirectShow中所有的Filter都實作的介面
- IPin
  - 代表多媒體訊息在IFilterGraph中流動的接腳
  - 每個IBaseFilter都有一個以上的IPin
- IMediaControl
  - 控制多媒體訊息撥放行為的元件
- IMediaEvent
  - 取得多媒體訊息撥放時的事件

# 何謂 Filter Graph ?



\*Render WMV9時會自動建立的Filter Graph



\*Render MPEG時會自動建立的Filter Graph

Renderer

# Filter的觀念如何對應應用

---

- streaming protocol -> Source Filter
  - mms
- demux -> Splitter Filter
  - avi
- codec -> Transform Filter
  - mp3
  - wma
  - wmv

# IFilterGraph

---

- ❑ IFilterGraph為DirectShow應用程式中，用來表示整個撥放結構的介面
- ❑ DirectShow應用建立透過建立IFilterGraph物件，逐一建立所需的filter，依需要及目的設置並連接各個filter，啟動IFilterGraph的執行
- ❑ IFilterGraph所提供的methods，便是包括了
  - 加入filter至graph（AddFilter）
  - 將filter自graph中移除（RemoveFilter）
  - 連接兩個filter（ConnectDirect）
  - 將指定filter的pin連接自graph中切斷（Disconnect）

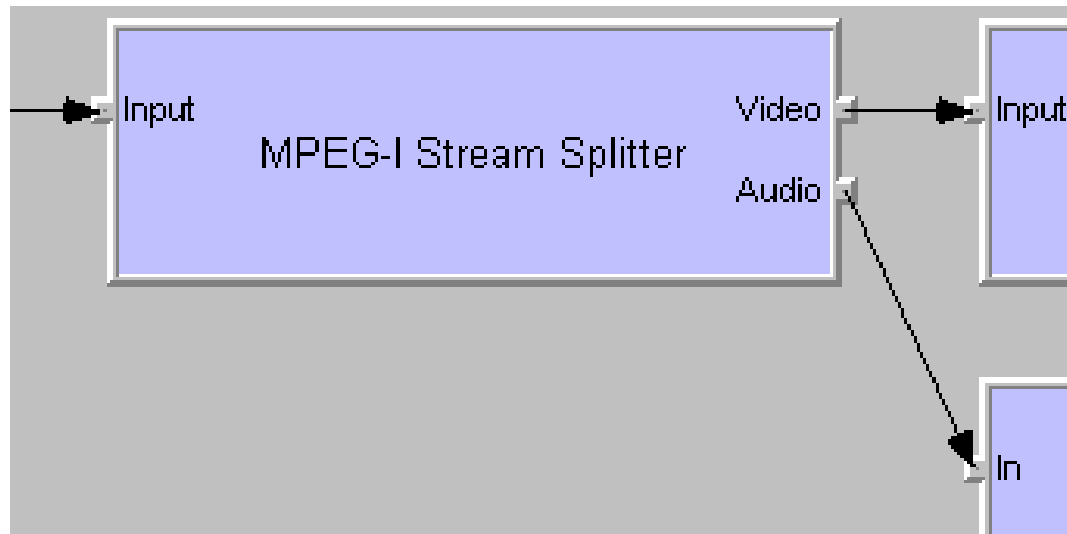
# IQueryBuilder

---

- IQueryBuilder繼承IFilterGraph，但提供更進階的功能，包括：
  - 連接兩個filter，倘若這兩個filter無法直接連接，則自動連接其間所需的 filter（Connect）
  - 自動自某個output pin接上播放所需的所有filter（Render）
  - 自動建立撥放某個檔案所需的所有filter（RenderFile）

# 何謂Filter？

- ❑ 一個filter具備一個以上的pin
- ❑ pin依多媒體訊息流動的方向可分為input pin或output pin
- ❑ filter的input pin，代表對此filter是訊息流入的pin
- ❑ filter的output pin，代表對此filter是訊息流出的pin



# IBaseFilter

---

- IBaseFilter是DirectShow filter的最主要介面，所有的filter都會實作這個介面
- 你主要會透過IBaseFilter介面達成下述的動作
  - 列舉出此filter上的所有pin（EnumPins）
  - 找出此filter上特定的pin（FindPin）
  - 取得filter的相關資訊（QueryFilterInfo）

# AM\_MEDIA\_TYPE

---

- AM\_MEDIA\_TYPE描述了在filter graph裡流動的media sample的類型
  - majorytype: 主要類型，例如
    - MEDIATYPE\_Video
    - MEDIATYPE\_Audio
  - subtype: 次要類型，例如
    - MEDIASUBTYPE\_RGB32
    - MEDIASUBTYPE\_PCM
  - formattype: 格式的類型
  - cbFormat: 格式資料的長度
  - pbFormat: 格式資料

# 格式資料

---

- formattype之值決定pbFormat所指向的結構為何
  - FORMAT\_VideoInfo -> VIDEOINFOHEADER
    - 畫面的座標、傳輸的bitRate、每個frame所需的時間、bitmap的資訊、等等
  - FORMAT\_WaveFormatEx -> WAVEFORMATEX
    - channel數、每秒的sample數、每秒平均的byte數、每個sample的bits數、等等

# 何謂Pin

---

- ❑ pin是DirectShow中多媒體訊息藉以流經的元件
- ❑ pin是DirectShow中，兩個filter相接的介面
- ❑ 每個pin都有可以接受的media type
- ❑ 在filter graph中要接在一起的兩個filter，其中一個的output Pin可接受的media type必須和另一個filter的input pin可接受的media type相符

# IPin (1/2)

---

- 每個filter皆有一個以上的IPin
- IPin（接腳）扮演連接兩個filter的角色
- 所有input pin和output pin皆實作的介面
- 通常應用程式不會呼叫IPin的methods來改變它的狀態，但是會取得它的一些資訊，包括：
  - 取得目前pin所建立之連接的media type（ConnectionMediaType）
  - 取得和pin有關的一些資訊，像是名稱、方向（QueryPinInfo）
  - 取得pin的id（QueryId）

# IPin (2/2)

---

- (續) 通常應用程式不會呼叫IPin的methods來改變它的狀態，但是會取得它的一些資訊，包括：
  - 判斷該pin是否接受某一種media type (QueryAccept)
  - 列舉出該pin所偏好的media type (EnumMediaTypes)
  - 查詢該pin的方向究竟為input或output (QueryDirection)

# filter的連接

---

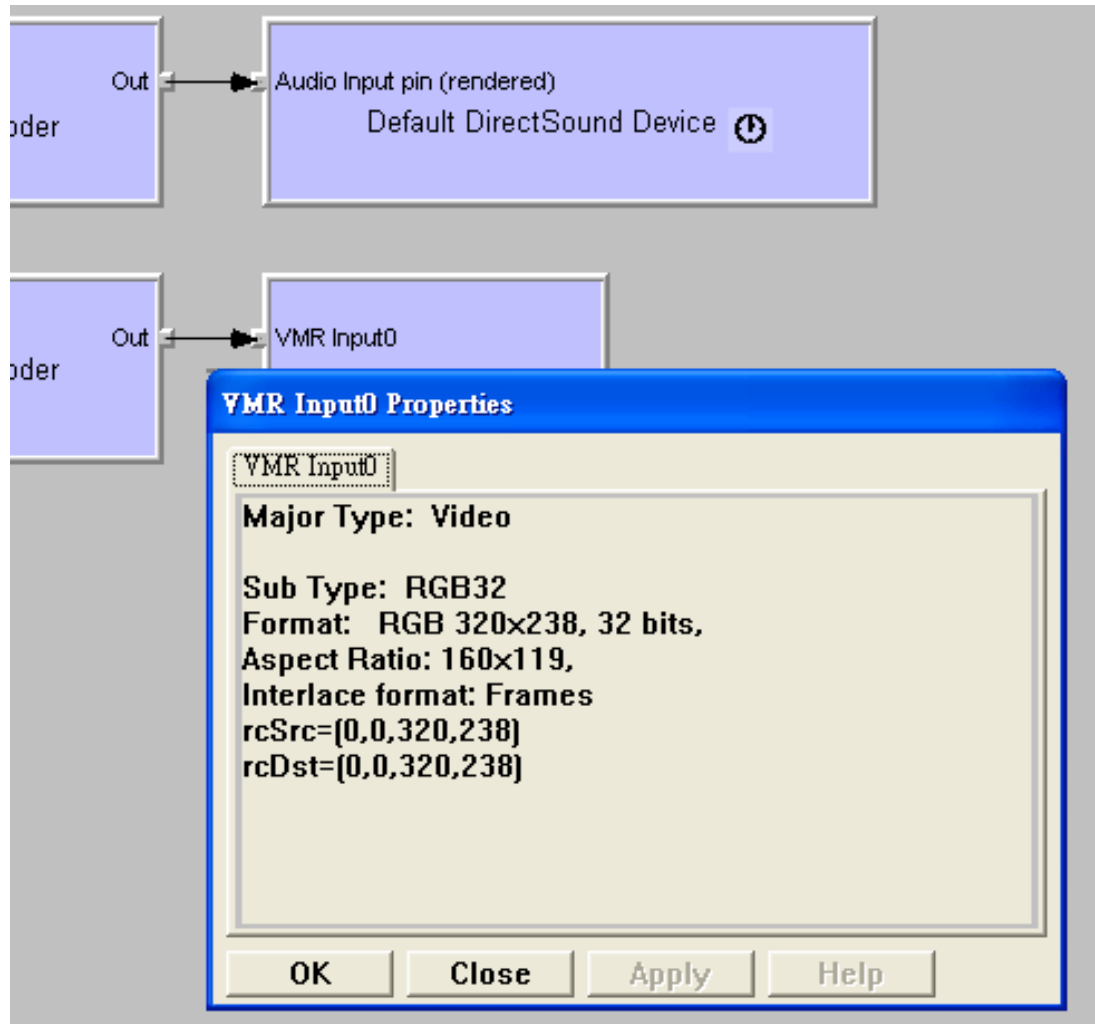
- ❑ 當兩個filter相連接時，其實意謂著是兩個filter上的pin相接
- ❑ 上游的filter的output pin連接下游filter的input pin
- ❑ 當我們指定連接兩個pin時，便會引發兩個pin的協商過程（negotiation），目的便是要協商出二者都能接受的media type

# Filter的類型

---

- ❑ Source Filter (來源端)
- ❑ Mux/Muxer (多合一)
  - AVI Mux
- ❑ Splitter (一分多)
  - AVI Splitter
- ❑ Transform Filter
  - Decoder
  - Encoder
  - Color Space Converter
- ❑ Renderer Filter (末端)
  - Video Renderer
  - Audio Renderer
  - AsfWriter
  - NetworkSink

# 經協商後Pin的連線媒體類型



# Filter Graph的建立

---

- 手動建立
  - 自己產生所需的各種filter的instance
  - 依據需求把要連接在一起的filter兩兩接在一起
    - 把Filter A的output pin接到Filter B的input pin
- 智慧型連接需要的Filter，透過IGraphBuilder
  - AddSourceFilter
  - Connect()
  - Render()
  - RenderFile()

# 智慧型連接

- 每filter都有一個merit值，決定智慧型連接進行時的優先順序
- 自動連接機制會逐一嘗試各filter之pin的media type是否能相符

```
[-] Video Renderer:  
  ... DisplayName: @device:sw:{083863F1-70DE-11D0-BD40-00A0C911CF  
  ... Filename: C:\WINDOWS\system32\quartz.dll  
  ... Merit: 00800001  
  [-] pin 00:  
    ... bMany: 0  
    ... bOutput: 0  
    ... bRendered: 1  
    ... bZero: 0  
    ... ClsPinCategory: {00000000-0000-0000-0000-000000000000}  
    [+ type 00  
  ... Version: 2
```

# 應用 DirectShow 在 Windows Mobile 上需留意的限制

---

- ❑ DirectShow 於 Windows Mobile 幾乎與 PC 上無異
- ❑ Windows Mobile 上的內建 filter 較少
- ❑ 大部份的 filter 是由 Windows Mobile 或製造商所提供
- ❑ 由於運算能力的限制，編解碼的能力都大為受限

# 一般的PC上都安裝大量的filter

## [-] DirectShow Filters

- + RAM file Parser
- + 9x8Resize
- + AC3 Parser Filter
- + AC3File
- + AC3Filter
- + ACELP.net Sipro Lab Audio Decoder
- + ACM Wrapper
- + Allocator Fix
- + ASF ACM Handler
- + ASF DIB Handler
- + ASF DJPEG Handler
- + ASF embedded stuff Handler
- + ASF ICM Handler
- + ASF JPEG Handler
- + ASF URL Handler
- + ASX file Parser
- + ASX v.2 file Parser
- + Audio Source
- + AVI Decompressor
- + AVI Draw
- + MONOGRAM AMR Mux
- + MONOGRAM AMR Splitter
- + MONOGRAM Musepack Decoder
- + MONOGRAM Musepack Splitter
- + Morgan RTP Source Filter
- + MP3Decoder Filter
- + MP4 Source
- + MP4 Splitter
- + MPEG Audio Decoder
- + MPEG Layer-3 Decoder
- + Mpeg Source
- + Mpeg Splitter
- + MPEG Video Decoder
- + MPEG-2 Demultiplexer
- + MPEG-2 Sections and Tables
- + MPEG-2 Splitter
- + MPEG-2 Video Stream Analyzer
- + Mpeg4 Decoder DMO
- + MPEG4 Video Source
- + MPEG4 Video Splitter
- + Mpeg43 Decoder DMO

# Windows Mobile上會有的filter—範例 (1/2)

---

- ❑ HTC ADX Renderer
- ❑ HTC DX Renderer
- ❑ HTC Audio Decode Transform
- ❑ HTC Video Decode Transform
- ❑ HTC DDR
- ❑ MPEG-1 Layer3 Decoder DMO
- ❑ HTC Source Filter2
- ❑ Audio Renderer
- ❑ Video Renderer
- ❑ WMVideo9 Encoder DMO
- ❑ Buffering Filter
- ❑ ACM Wrapper

# Windows Mobile上會有的filter—範例 (2/2)

---

- AVI Decompressor
- AVI Splitter
- Wave Parser
- Color Space Converter
- VO PD Source Filter
- 3GPP Streaming Source Filter
- File Source( Async. )
- HTC Audio Renderer
- Audio Capture
- Video Capture
- Image Sink Filter
- ASF Writer
- Smart Tee

# 透過C#操控COM元件，必須要能夠

---

- ❑ 建立COM元件的instance
- ❑ 查詢COM元件的介面
- ❑ 透過COM介面呼叫method
- ❑ 資料結構的對應

# 在C#中建立COM元件

- Unmanaged code: CoCreateInstance()

```
CoCreateInstance(CLSID_FilterGraph, NULL,  
CLSCTX_INPROC, IID_IFilterGraph, (void**)  
&pFilterGraph);
```

```
[ComImport, Guid("e436ebb3-524f-11ce-9f53-  
0020af0ba770")]
```

```
public class FilterGraph
```

```
{
```

```
}
```

```
...
```

```
FilterGraph graph = new FilterGraph();
```

## 在C#中查詢COM元件的介面

- Unmanaged code: : comObject->QueryInterface()

```
IMediaControl *pMediaControl = NULL;  
pFilterGraph->QueryInterface(IID_IMediaControl,  
    (void **) &pMediaControl );
```

```
IMediaControl mediaControl = (IMediaControl)graph;
```

# 呼叫COM元件的method

---

- Unmanaged:

```
pMediaControl->Run();
```

- C#

```
mediaControl.Run();
```

## 資料結構的對映 (1/2)

```
[StructLayout(LayoutKind.Sequential)]  
public class AMMediaType  
{  
    public Guid majorType;  
    public Guid subType;  
    [MarshalAs(UnmanagedType.Bool)] public bool  
        fixedSizeSamples;  
    [MarshalAs(UnmanagedType.Bool)] public bool  
        temporalCompression;
```

## 資料結構的對映 (2/2)

---

```
public int sampleSize;  
public Guid formatType;  
public IntPtr unkPtr;  
public int formatSize;  
public IntPtr formatPtr;  
}
```

# DirectShow.NET & DirectShow.NET CF

---

- <http://directshownet.sourceforge.net/>
- LGPL
- 此類別庫能讓.NET應用程式使用DirectShow的功能
  - 因此不限C# — 所有的.NET程式語言皆能使用
- DirectShow中的COM元件、COM介面、以及資料結構，此類別庫皆已完整且廣泛的定義
  - 程式設計者毋需再自行耗費時間
- 不過.NET與.NET Compact Framework仍然一些差異
  - 我稍微抽出了最主要常用的部份，可適用於.NET Compact Framework
  - <http://www.javaworld.com.tw/roller/qing/resource/DirectShowNetCF.zip>

# DirectShow.NET 定義了廣泛的類別

```
/// <summary>
/// CLSID_FilterGraph
/// </summary>
[ComImport, Guid("e436ebb3-524f-11ce-9f53-0020af0ba770")]
public class FilterGraph
{
}

/// <summary>
/// CLSID_FilterGraphNoThread
/// </summary>
[ComImport, Guid("e436ebb8-524f-11ce-9f53-0020af0ba770")]
public class FilterGraphNoThread
{
}

/// <summary>
/// CLSID_CaptureGraphBuilder2
/// </summary>
[ComImport, Guid("BF87B6E1-8C27-11d0-B3F0-00AA003761C5")]
public class CaptureGraphBuilder2
{
}

/// <summary>
/// CLSID_DvdGraphBuilder
/// </summary>
[ComImport, Guid("FCC152B7-F372-11d0-8E00-00C04FD7C08B")]
public class DvdGraphBuilder
{
}
```

# DirectShow.NET 定義了廣泛的介面

```
[ComImport,
Guid("56a86891-0ad4-11ce-b03a-0020af0ba770"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
public interface IPin
{
    [PreserveSig]
    int Connect(
        [In] IPin pReceivePin,
        [In, MarshalAs(UnmanagedType.LPStruct)] AMMediaType pmt
    );

    [PreserveSig]
    int ReceiveConnection(
        [In] IPin pReceivePin,
        [In, MarshalAs(UnmanagedType.LPStruct)] AMMediaType pmt
    );

    [PreserveSig]
    int Disconnect();

    [PreserveSig]
    int ConnectedTo(
        [Out] out IPin ppPin);

    /// <summary>
    /// Release returned parameter with DsUtils.FreeAMMediaType
    /// </summary>
    [PreserveSig]
    int ConnectionMediaType(
        [Out, MarshalAs(UnmanagedType.LPStruct)] AMMediaType pmt);
}
```

# DirectShow.NET 定義了廣泛的資料結構

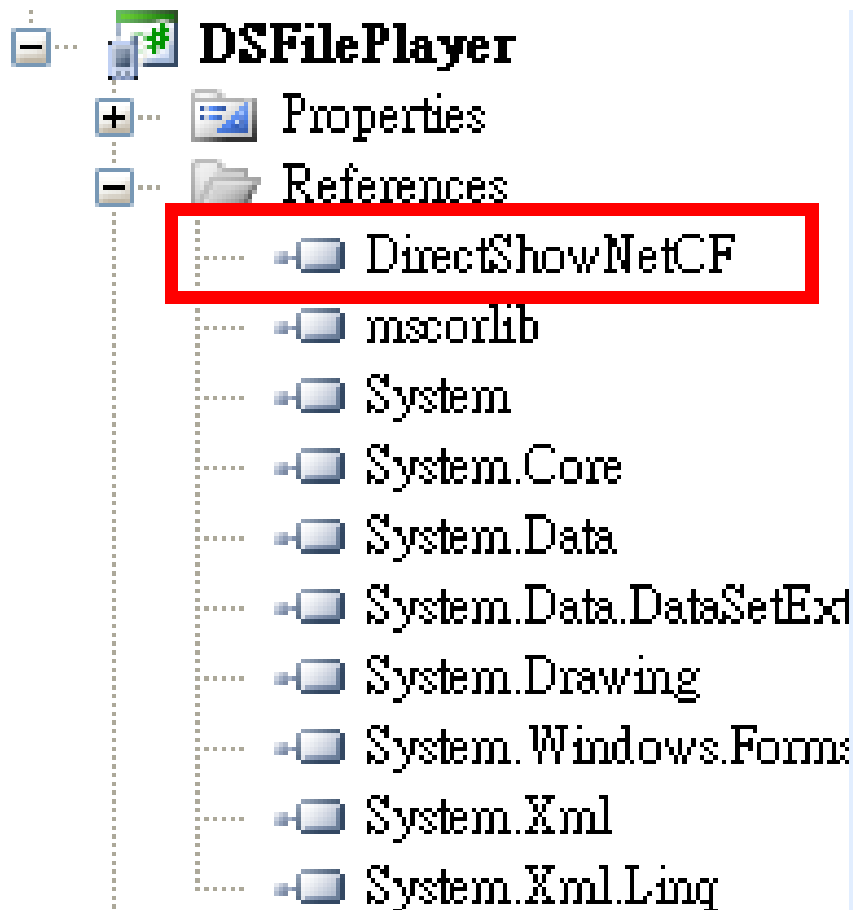
```
/// <summary>
/// From PIN_INFO
/// </summary>
[StructLayout(LayoutKind.Sequential, CharSet=CharSet.Unicode)]
public struct PinInfo
{
    [MarshalAs(UnmanagedType.Interface)] public IBaseFilter filter;
    public PinDirection dir;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst=128)] public string name;
}

/// <summary>
/// From AM_MEDIA_TYPE - When you are done with an instance of this class,
/// it should be released with FreeAMMediaType() to avoid leaking
/// </summary>
[StructLayout(LayoutKind.Sequential)]
public class AMMediaType
{
    public Guid majorType;
    public Guid subType;
    [MarshalAs(UnmanagedType.Bool)] public bool fixedSizeSamples;
    [MarshalAs(UnmanagedType.Bool)] public bool temporalCompression;
    public int sampleSize;
    public Guid formatType;
    public IntPtr unkPtr; // IUnknown Pointer
    public int formatSize;
    public IntPtr formatPtr; // Pointer to a buff determined by formatType
}

/// <summary>
/// From PIN_DIRECTION
/// </summary>
public enum PinDirection
{
    Input,
    Output
}
```

# 如何使用 DirectShow.NET

- ❑ 把DirectShowNetCF.dll加至你專案的References中



# 所需的namespace

---

- ❑ `using System.Runtime.InteropServices;`
- ❑ `using DirectShowLib;`

# 一個最簡單的DirectShow範例 (DSFilePlayer)

- 目的：從檔案執行撥放指定的多媒體檔案
- 步驟：
  1. 建立FilterGraph的instance
  2. 取得IMediaControl的介面（FilterGraph實作了IMediaControl的介面）
  3. 取得IVideoWindow的介面
  4. 取得IMediaPosition的介面（IFilterGraph實作了IMediaPosition的介面）
  5. 透過IMediaControl智慧型的建立Filter Graph的內容
  6. 設定IVideoWindow
  7. 透過IMediaControl啟動撥放
  8. 透過IMediaControl結束撥放
  9. 透過IMediaPosition取得進度

# DSFilePlayer : 撥放

```
graph = new FilterGraph();  
mediaControl = (IMediaControl)graph;  
mediaPosition = (IMediaPosition)graph;  
videoWindow = (IVideoWindow)graph;  
int result = mediaControl.RenderFile(ofd.FileName);  
result = videoWindow.put_Owner(pbVideo.Handle);  
result =  
    videoWindow.put_WindowStyle(WindowStyle.Child |  
    WindowStyle.ClipSiblings);  
result = mediaControl.Run();  
timerProgress.Enabled = true;
```

# DSFilePlayer : 停止撥放

---

```
mediaControl.Stop();
```

```
Marshal.ReleaseComObject(graph);
```

```
graph = null;
```

```
mediaControl = null;
```

```
mediaPosition = null;
```

```
videoWindow = null;
```

# DSFilePlayer : 顯示撥放進度

```
double duration = 0.0;  
double position = 0.0;  
mediaPosition.get_Duration(out duration);  
mediaPosition.get_CurrentPosition(out position);  
int d = (int) duration;  
int p = (int) position;  
lbProgress.Text = p + "/" + d;
```

# 從擷取設備（視訊或音訊）取得資訊 (VideoCap)

- 目的：從擷取設備擷取多媒體資訊
- 步驟：
  1. 建立FilterGraph的instance
  2. 取得FilterGraph所實作的所需介面
  3. 建立CaptureGraphBuilder的instance
  4. 取得ICaptureGraphBuilder2介面  
（ CaptureGraphBuilder所實作的）並設定filter graph
  5. 取得Video Capture Filter並設定
  6. 取得Video Renderer Filter並設定
  7. 將filters加入filter graph
  8. ICaptureGraphBuilder2.RenderStream()
  9. 設定IVideoWindow
  10. 透過IMediaControl啟動撥放
  11. 透過IMediaControl結束撥放

# VideoCap: 初始動作

```
graph = new FilterGraph();  
graphBuilder = (IGraphBuilder) graph;  
mediaControl = (IMediaControl)graph;  
videoWindow = (IVideoWindow)graph;  
//  
captureGraph = new CaptureGraphBuilder();  
captureGraphBuilder =  
    (ICaptureGraphBuilder2)captureGraph;  
captureGraphBuilder.SetFiltergraph(graphBuilder);
```

# VideoCap: 取得設定 Video Capture Filter (1/2)

```
[ComImport, Guid("F80B6E95-B55A-4619-AEC4-  
A10EAEDE980C")]  
public class VideoCapture  
{  
}  
  
videoCapture = new VideoCapture();  
videoCaptureFilter = (IBaseFilter) videoCapture;  
//  
string name = "";  
if (!getName(ref name))  
    return false;
```

# VideoCap: 取得設定 Video Capture Filter (2/2)

```
IPersistPropertyBag propBag =  
    (IPersistPropertyBag)videoCaptureFilter;  
if (propBag == null)  
    return false;  
CPropertyBag bag = new CPropertyBag();  
object oname = name;  
bag.Write("VCapName", ref oname);  
int hr = propBag.Load(bag, null);
```

# VideoCap: 取得 Video Capture 設備的名稱 (1/4)

```
public struct DEVMGR_DEVICE_INFORMATION
{
    public uint dwSize;
    public IntPtr hDevice;
    public IntPtr hParentDevice;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 6)]
    public string szLegacyName;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 260)]
    public string szDeviceKey;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 260)]
    public string szDeviceName;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 260)]
    public string szBusName;
}
```

# VideoCap: 取得 Video Capture 設備的名稱 (2/4)

```
public class PInvoke
{
    [DllImport("coredll.dll")]
    public static extern IntPtr FindFirstDevice(
        [In] int searchType,
        [In] IntPtr searchParam,
        [In, Out] ref
        DEVMGR_DEVICE_INFORMATION pdi);
    [DllImport("coredll.dll")]
    public static extern int FindClose([In] IntPtr
    hFindFile);
}
```

# VideoCap: 取得 Video Capture 設備的名稱 (3/4)

```
IntPtr handle = IntPtr.Zero;  
IntPtr guid =  
    Marshal.AllocHGlobal(Marshal.SizeOf(typeof(Guid))  
    );  
Marshal.StructureToPtr(CLSIDDef.Camera, guid, false);  
//  
DEVMGR_DEVICE_INFORMATION di = new  
    DEVMGR_DEVICE_INFORMATION();  
di.dwSize = (uint)  
    Marshal.SizeOf(typeof(DEVMGR_DEVICE_INFOR  
    MATION));
```

```
public static readonly Guid Camera = new Guid("CB998A05-122C-4166-846A-933E4D7E3C86");
```

# VideoCap: 取得 Video Capture 設備的名稱 (4/4)

```
handle = PInvoke.FindFirstDevice(3, guid, ref di);
Marshal.FreeHGlobal(guid);
if ((handle == IntPtr.Zero) || (di.hDevice == IntPtr.Zero))
    return false;
//
PInvoke.FindClose(handle);
name = di.szLegacyName;
```

# VideoCap: 取得 Video Renderer Filter

```
[ComImport, Guid("70E102B0-5556-11CE-97C0-00AA0055595A")]
```

```
public class VideoRenderer
```

```
{
```

```
}
```

```
videoRenderer = new VideoRenderer();
```

```
videoRendererFilter = (IBaseFilter)videoRenderer;
```

# VideoCap: 建立 Filter Graph

---

```
graphBuilder.AddFilter(videoCaptureFilter, "Video  
Capture Filter");
```

```
graphBuilder.AddFilter(videoRendererFilter, "Video  
Renderer Filter");
```

```
captureGraphBuilder.RenderStream(null, null,  
videoCaptureFilter, null, videoRendererFilter);
```

# VideoCap: 設定視窗並撥放

```
videoWindow.put_Owner(pbVideo.Handle);  
videoWindow.put_WindowStyle(WindowStyle.Child |  
    WindowStyle.ClipSiblings);  
mediaControl.Run();
```

# VideoCap: 釋放所有產生的物件

```
if (graph != null)
{
    Marshal.ReleaseComObject(graph);
    graph = null;
}
if (videoCapture != null)
{
    Marshal.ReleaseComObject(videoCapture);
    videoCapture = null;
}
if (videoRenderer != null)
{
    Marshal.ReleaseComObject(videoRenderer);
    videoRenderer = null;
}
```

# 音訊的擷取

- 和視訊雷同，但只需建立AudioCapture，毋需設定設備名稱

```
[ComImport, Guid("e30629d2-27e5-11ce-875d-00608cb78066")]
```

```
public class AudioCapture
```

```
{
```

```
}
```

# 檔案的儲存

---

```
IBaseFilter asfWriter = null;  
IFilterSink fileSink = null;  
captureGraphBuilder.SetOutputFileName(MediaSubType.  
Asf, "\\My Documents\\VidCap.wmv", out  
asfWriter, out fileSink);
```

# 手動/半手動建立Filter Graph

---

- 許多時候，自動建立的filter graph不能滿足我們的需求，或是甚至無法自動建立，此時就需要以撰寫程式的方式，手動或半手動建立
- 欲手動連接兩個filter
  - 找出上游filter的output pin
  - 找出下游filter的input pin
  - 呼叫IFilterGraph的ConnectDirect()將二個pin連接在一塊

# FindPinOnFilter (1/2)

```
private int FindPinOnFilter(IBaseFilter filter, PinDirection pinDir, int
    pinIndex, out IPin pin)
{
    IEnumPins pEnum;
    pin = null;
    int hr = filter.EnumPins(out pEnum);
    if (hr != 0)
        return hr;
    int nFound = 0;
    while (true)
    {
        IPin[] pins = new IPin[1];
        hr = pEnum.Next(1, pins, pFetched);
        if (hr != 0)
            break;
    }
}
```

# FindPinOnFilter (2/2)

```
pin = pins[0];
PinDirection pinDir2;
hr = pin.QueryDirection(out pinDir2);
if (hr == 0 && pinDir2 == pinDir)
{
    if (nFound == pinIndex)
        break;
    nFound++;
}
}
return hr;
}
```

# 連接兩個filter

```
IPin pinOut;
```

```
IPin pinIn;
```

```
hr = FindPinOnFilter(filter1, PinDirection.Output, 0,  
out pinOut);
```

```
hr = FindPinOnFilter(filter2, PinDirection.Input, 0, out  
pinIn);
```

```
hr = graph.ConnectDirect(pinOut, pinIn, null);
```

# 下載

---

- ❑ <http://www.javaworld.com.tw/roller/qing/resource/DirectShowNetCF.zip>
  - DirectShowNetCF
  - DSPlayFile
  - VideoCap

## 結語

---

- ❑ DirectShow的架構博大精深，本講題僅試著提供一個簡單的概活性介紹，以及最基礎常用的應用做法
- ❑ 您可以本講題為基礎，進一步了解Windows Mobile上DirectShow的豐富支援及變化
- ❑ 發揮您的創意，開發出更有趣的Windows Mobile多媒體應用程式



Q&A

Thanks