

Home > Multimedia

Using a Camera with Windows Mobile 5

By CodeGuru Staff July 21, 2006



CodeGuru content and product recommendations are editorially independent. We may make money when you click on links to our partners. [Learn More](#).

A number of Windows Mobile 5.0 APIs (for example, SHCameraCapture) make it trivial for a mobile application developer to access a camera, but their ease of use comes at a price—flexibility. Most of the time, using the API directly would offer a solution, but sometimes you need more control and flexibility. That's where Microsoft's DirectShow framework comes in. This article shows how to use DirectShow to access a camera. It demonstrates how to build a filter graph manually and how to handle graph events in the application message handler. Having some prior knowledge of DirectShow and COM will be helpful, but it's not necessary.

Figure 1 depicts the components in the filter graph you will use to capture video.

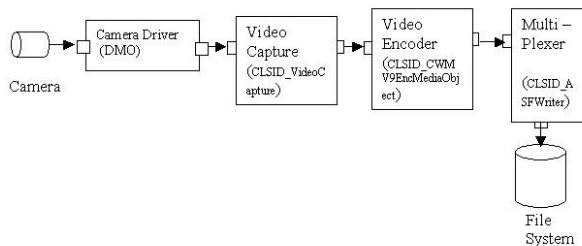


Figure 1: Filter Graph for Video Capture

The camera is the hardware component. For an application to interact with the camera, it would need to talk to its drivers. Next, the video capture filter enables an application to capture video. After capture, you encode the data using WMV9EncMediaObject, a DirectX Media Object (DMO). You can use a DMO inside a filter graph with the help of a DMO Wrapper filter. Next, the encoded video data needs to be multiplexed. You use a Windows Media ASF writer filter for this task. The ASF writer multiplexes the video data and writes it to an .asf file. With that, your filter graph is ready. Now, it's just a matter of running it. As you will see, building the graph is pretty easy too.

Set the Build Environment

First, you need to set the build environment. Add the following libraries in the linker setting of a Visual Studio 2005 Smart Device project:

- dmoguids.lib
- strmiids.lib
- strmbase.lib
- uuid.lib

Also include the following header files in your project:

- atlbase.h
- dmodeshow.h
- dmores.h
- wmcodeids.h

NOTE: FOR THE SAKE OF CLARITY, THIS EXAMPLE DOESN'T SHOW ERROR HANDLING. HOWEVER, A REAL WORLD APPLICATION WOULD REQUIRE ERROR HANDLING.

Get the Free Newsletter!

Subscribe to Developer Insider for top news, trends & analysis

☐ By signing up to receive our newsletter, you agree to our [Terms of Use](#) and [Privacy Policy](#).

Must Read



Drawing 3D OpenGL Graphics on Google Maps

Pravin Kumar Sinha - May 18, 2016



Mandelbrot Using C++ AMP

CodeGuru Staff - January 27, 2012

Building the Graph

A filter graph that performs audio or video capture is known as a Capture graph. DirectShow provides a Capture Graph Builder object that exposes an interface called ICaptureGraphBuilder2; it exposes methods to help build and control a capture graph.

First, create instances of IGraphBuilder and ICaptureGraphBuilder2 by using the COM function CoCreateInstance:

```
HRESULT hResult = S_OK;

IGraphBuilder *pFilterGraph;
ICaptureGraphBuilder2 *pCaptureGraphBuilder;

hResult=CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC,
                        IID_IGraphBuilder, (void**)&pFilterGraph);

hResult=CoCreateInstance(CLSID_CaptureGraphBuilder, NULL,
                        CLSCTX_INPROC, IID_ICaptureGraphBuilder2,
                        (void**)& pCaptureGraphBuilder);
```

CoCreateInstance takes five parameters:

1. The first is a class ID.
2. The second decides whether the object created is part of an aggregator.
3. The third specifies the context in which the newly created object would run.
4. The fourth parameter is a reference to the identifier of the interface you will use to communicate with the object.
5. The last parameter is the address of the variable that receives the interface pointer requested.

Once you have created the IGraphBuilder and ICaptureGraphBulder2 instances, you need to call the SetFilterGraph method of the ICaptureGraphBuilder2 interface:

```
hResult = m_pCaptureGraphBuilder->SetFiltergraph( pFilterGraph );
```

The SetFilterGraph method takes a pointer to the IGraphBuilder interface. This specifies which filter graph the capture graph builder will use. If you don't call the SetFilterGraph method, the Capture graph builder automatically creates a graph when it needs it.

Now, you're ready to create an instance of the video capture filter. The following code initializes a Video capture filter, the pointer of which is returned by the CoCreateInstance:

```
IBaseFilter *pVideoCaptureFilter

hResult=CoCreateInstance(CLSID_VideoCapture, NULL, CLSCTX_INPROC,
                        IID_IBaseFilter, (void**)&pVideoCaptureFilter);
```

You then need to get a pointer to IPersistPropertyBag from the video capture filter. You use this pointer to set the capture device (in other words, the camera) that the capture filter will use, as follows:

```
IPersistPropertyBag *pPropertyBag;

hResult=pVideoCaptureFilter->QueryInterface( &pPropertyBag );
```

Now, you need to get a handle on the camera you will use to capture video. You can enumerate the available camera devices by using the FindFirstDevice and FindNextDevice functions. You can have multiple cameras present on a device. (HTC Universal is one example.) To keep the code simple for this example, use FindFirstDevice to get the first available camera on the device as follows:

```
DEVMGR_DEVICE_INFORMATION devInfo;
CComVariant CamName;
CPropertyBag PropBag;

GUID guidCamera = { 0xCB998A05, 0x122C, 0x4166, 0x84, 0x6A, 0x93,
                    0x3E, 0x4D, 0x7E, 0x3C, 0x86 };

devInfo.dwSize = sizeof(devInfo);

FindFirstDevice( DeviceSearchByGuid, &guidCamera, & devInfo);

CamName=devInfo.szLegacyName

PropBag.Write( _T("VCapName"), &CamName );

pPropertyBag->Load( &PropBag, NULL );

hResult =pFilterGraph->AddFilter( pVideoCaptureFilter,
                                _T("Video Capture Filter") );

pPropertyBag.Release();
```

Note the first parameter in the FindFirstDevice, DeviceSearchByGuid. It specifies the search type. Other options are DeviceSearchByLegacyName, DeviceSearchByDeviceName, and so forth. DeviceSearchByGuid is the most reliable way to find a capture device. The information regarding the device is returned in the DEVMGR_DEVICE_INFORMATION structure. You store the szLegacyName value in the CComVariant variable, and you need an object that has implemented IPropertyBag interface.

In the code sample, CPropertyBag is a custom class that has implemented IPropertyBag. This object is needed to pass the capture device name to the filter. The string VCapName identifies the filter property for the name of the video capture device. Once you have set the capture device, you can add the Video capture filter to the filter graph. You use the AddFilter method of the graph manager for this. This method takes two parameters: the first is the pointer to the filter that is to be added, and the second is the name of the filter. The second parameter can be NULL; in this case, the filter graph manager generates a unique name for the filter. If you have provided a name that conflicts with some other filter, the manager will modify the name to make it unique.

You then need to instantiate the WMV9 encoder:

```
IBaseFilter *pVideoEncoder;
IDMOWrapperFilter *pWrapperFilter;

hResult=CoCreateInstance(CLSID_DMOWrapperFilter, NULL,CLSCTX_INPROC,
    IID_IBaseFilter, (void*)&pVideoEncoder);

hResult =pVideoEncoder->QueryInterface( &pWrapperFilter );

hResult =pWrapperFilter->Init( CLSID_CWMV9EncMediaObject,
    DMOCATEGORY_VIDEO_ENCODER );

hResult=pFilterGraph->AddFilter( pVideoEncoder, L"WMV9DMO Encoder");
```

Because the WMV9 encoder is a DMO, you can't add/use it like other filters. But, DirectShow provides a wrapper filter that enables you to use a DMO like any other filter. You first create an instance of the DMO wrapper filter and then initialize the WMV9 encoder DMO with it. After initializing the DMO, you add it into the filter graph as follows:

```
IBaseFilter *pASFMultiplexer;
IFileSinkFilter *pFileSinkFilter;

hResult = pCaptureGraphBuilder->SetOutputFileName(
    &MEDIASUBTYPE_Asf, T("\\test.asf"), &pASFMultiplexer,
    &pFileSinkFilter );
```



CodeGuru Staff

More By Author

Best Video Game Development Tools

CodeGuru Staff - September 30, 2022

[Read more](#)

Video Game Careers Overview

CodeGuru Staff - September 18, 2022

[Read more](#)

The Top Task Management Software for Developers

CodeGuru Staff - August 25, 2022

[Read more](#)

Best Online Courses for .NET Developers

CodeGuru Staff - June 30, 2022

[Read more](#)

Previous article

[Using Generic Methods](#)

Next article

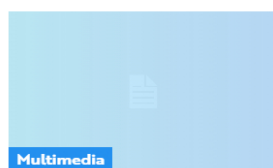
[Creating a Treeview Menu in ASP.NET with C#](#)

News & Trends



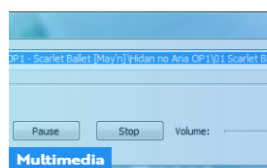
Drawing 3D OpenGL Graphics on Google Maps

Pravin Kumar Sinha - May 18, 2016



Mandelbrot Using C++ AMP

CodeGuru Staff - January 27, 2012



Simple C++ MP3 Player Class

CodeGuru Staff - August 22, 2011



CodeGuru covers topics related to Microsoft-related software development, mobile development, database management, and web application programming. In addition to tutorials and how-tos that teach programmers how to code in Microsoft-related languages and frameworks like C# and .Net, we also publish articles on software development tools, the latest in developer news, and advice for project managers. Cloud services such as Microsoft Azure and database options including SQL Server and MSSQL are also frequently covered.



Advertisers

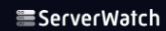
Advertise with TechnologyAdvice on CodeGuru and our other developer-focused platforms.

[Advertise with Us >](#)

Menu

[Privacy Policy](#)
[Terms](#)
[About](#)
[Contact](#)
[Advertise](#)
[California – Do Not Sell My Information](#)

Our Brands



[PRIVACY POLICY](#) [TERMS](#) [ABOUT](#) [CONTACT](#) [ADVERTISE](#) [CALIFORNIA – DO NOT SELL MY INFORMATION](#)

Property of TechnologyAdvice.
© 2023 TechnologyAdvice. All Rights Reserved

Advertiser Disclosure: Some of the products that appear on this site are from companies from which TechnologyAdvice receives compensation. This compensation may impact how and where products appear on this site including, for example, the order in which they appear. TechnologyAdvice does not include all companies or all types of products available in the marketplace.

We use cookies and other data collection technologies to provide the best experience for our customers. You may request that your data not be shared with third parties here: [Do Not Sell My Data.](#)

