

DirectShow Brief Description

Introduction to DirectShow

Microsoft® DirectShow® is an architecture for streaming media on the Microsoft Windows® platform. DirectShow provides for high-quality capture and playback of multimedia streams. It supports a wide variety of formats, including Advanced Systems Format (ASF), Motion Picture Experts Group (MPEG), Audio-Video Interleaved (AVI), MPEG Audio Layer-3 (MP3), and WAV sound files. It supports capture using Windows Driver Model (WDM) devices or older Video for Windows devices. DirectShow is integrated with other DirectX technologies. It automatically detects and uses video and audio acceleration hardware when available, but also supports systems without acceleration hardware.

DirectShow System Overview

Working with multimedia presents several major challenges:

- Multimedia streams contain large amounts of data, which must be processed very quickly.
- Audio and video must be synchronized so that it starts and stops at the same time, and plays at the same rate.
- Data can come from many sources, including local files, computer networks, television broadcasts, and video cameras.
- Data comes in a variety of formats, such as Audio-Video Interleaved (AVI), Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), and Digital Video (DV).
- The programmer does not know in advance what hardware devices will be present on the end-user's system.

The DirectShow Solution

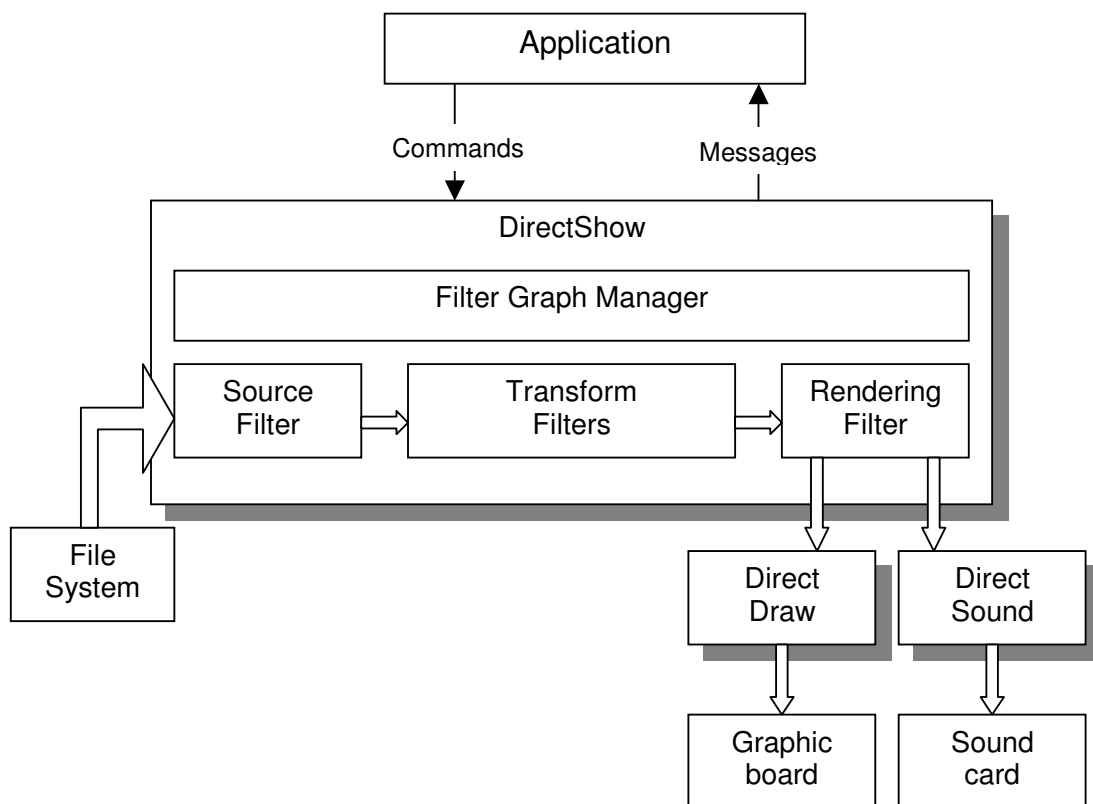
DirectShow is designed to address each of these challenges. Its main design goal is to simplify the task of creating digital media applications on the Windows® platform, by isolating applications from the complexities of data transports, hardware differences, and synchronization.

To achieve the throughput needed to stream video and audio, DirectShow uses DirectDraw® and DirectSound® whenever possible. These technologies render data efficiently to the user's sound and graphics cards. DirectShow synchronizes playback by encapsulating media data in time-stamped samples. To handle the variety of sources, formats, and hardware devices that are possible, DirectShow uses a modular architecture, in which the application mixes and matches different software components called filters.

DirectShow Brief Description

DirectShow provides filters that support capture and tuning devices based on the Windows Driver Model (WDM), as well as filters that support legacy Video for Windows (VfW) capture cards, and codecs written for the Audio Compression Manager (ACM) and Video Compression Manager (VCM) interfaces.

The following diagram shows the relationship between an application, the DirectShow components, and some of the hardware and software components that DirectShow supports.



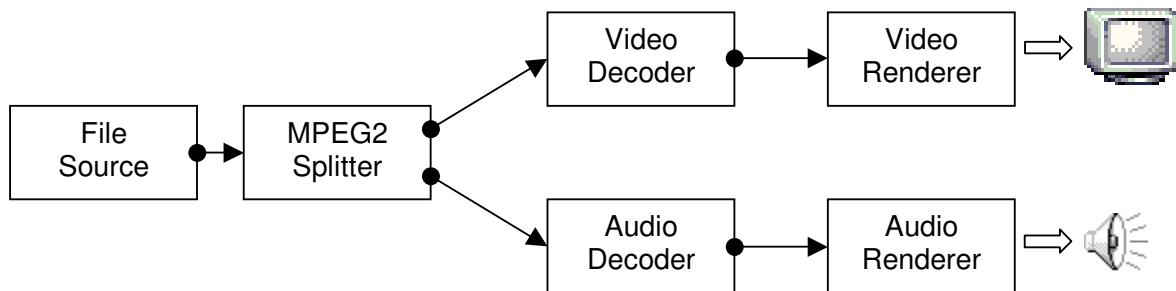
DirectShow Brief Description

About DirectShow Filters

DirectShow uses a modular architecture, where each stage of processing is done by a COM object called a filter. DirectShow provides a set of standard filters for applications to use, and developers can write their own custom filters that extend the functionality of DirectShow. To illustrate, here are the steps needed to play MPEG-2-Video with DolbyDigital® Sound.

- Read the raw data from the file as a byte stream (File Source filter)
- Examine the MPEG-Headers, and parse the byte stream into separate video frames and audio samples
- Decode the video frames (MPEG2-Decoder Filter)
- Draw the video frames (Video Renderer filter)
- Decode the audio samples (AC3 Audio Decoder)
- Send the audio samples to the sound card (Default DirectSound Device filter).

These filters are shown in the following diagram:



As the diagram shows, each filter is connected to one or more other filters. The connection points are also COM objects, called *pins*. Filters use pins to move data from one filter to the next. The arrows in the diagram show the direction in which the data travels. In DirectShow, a set of filters is called a *filter graph*.

Filters have three possible states: running, stopped, and paused. When a filter is running, it processes media data. When it is stopped, it stops processing data. The paused state is used to cue data before running.

With very rare exceptions, state changes are coordinated throughout the entire filter graph; all the filters in the graph switch states in unison. Thus, the entire filter graph is also said to be running, stopped, or paused.

DirectShow Brief Description

Filters can be grouped into several broad categories:

- A *source* filter introduces data into the graph. The data might come from a file, a network, a camera, or anywhere else. Each source filter handles a different type of data source.
- A *transform* filter takes an input stream, processes the data, and creates an output stream. Encoders and decoders are examples of transform filters.
- *Renderer* filters sit at the end of the chain. They receive data and present it to the user. For example, a video renderer draws video frames on the display; an audio renderer sends audio data to the sound card; and a file-writer filter writes data to a file.
- A *splitter* filter splits an input stream into two or more outputs, typically parsing the input stream along the way. For example, the MPEG2-Splitter, also known as Demultiplexer or Demuxer, parses a byte stream into separate video and audio streams.
- A *mux* filter takes multiple inputs and combines them into a single stream. For example, the MPEG2-Mux performs the inverse operation of the MPEG2-Splitter. It takes audio and video streams and produces an MPEG2-formatted byte stream.

The distinctions between these categories are not absolute. For example, the ASF Reader filter acts as both a source filter and a splitter filter.

DirectShow Brief Description

About the FilterGraph Manager

The FilterGraph Manager is a COM object that controls the filters in a filter graph. It performs many functions, including the following:

- Coordinating state changes among the filters.
- Establishing a reference clock.
- Communicating events back to the application.
- Providing methods for applications to build the filter graph.

Each of these functions is described briefly here.

State changes. 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 a similar fashion: The application gives a seek command to the Filter Graph Manager, which distributes it to the filters.

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 rendered 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.

Graph events. The Filter Graph Manager uses an event queue to inform the application of events that occur in the filter graph. This mechanism is similar to a Windows message loop.

Graph-building methods. The Filter Graph Manager provides methods for the application to add filters to the graph, connect filters to other filters, and disconnect filters.

One function the Filter Graph Manager does *not* handle is moving data from one filter to the next. This is done by the filters themselves, through their pin connections. Processing always happens on a separate thread.