Decoding Sample

Overview

Decoding Sample works with **Intel® Media SDK 2017 and Intel® Media Server Studio 2017 - SDK for Windows* Server** (hereinafter referred to as "**SDK**").

It demonstrates how to use the **SDK** API to create a sample console application that performs decoding of various video compression formats.

This sample also demonstrates how to use custom Stereoscopic 3D (S3D) Application Programming Interface (API) for the Intel® HD Graphics to display the decoded MVC stream on a 3D capable monitor or TV.

Stereoscopic 3D rendering using Microsoft* Direct3D* version 11.1 is also demonstrated.

The sample can work together with **Intel® Media Server Studio – HEVC Decoder & Encoder** (hereinafter referred to as "**HEVC**").

Note: To run HEVC, please read the instructions in the “HEVC Plugin” section carefully.

Features

Decoding Sample supports the following video formats:

Format type	
input (compressed)	H.264 (AVC, MVC – Multi-View Coding), MPEG-2 video, VC-1, JPEG*/Motion JPEG, HEVC (High Efficiency Video Coding),VP9
output (uncompressed)	YUV420

Note 1:Decoding Sample supports decoding of JPEG/Motion JPEG bitstreams that conform to the ITU-T* Rec. T.81 / ISO*/IEC* 10918-1, with an EXIF* or JFIF* header. Additionally, Microsoft* AVI1 Motion JPEG headers are supported to obtain stream scan type and in order to demonstrate **SDK** Decode support for interlaced motion JPEG streams.

Note 2: Decoding Sample renders the decoded video stream to a file in YUV 4:2:0 sampling format, with the color planes Y, U and V in that order.

Decoding Sample includes specific rendering features which can be used to arrange a video wall demo - several instances of **Decoding Sample** each rendering its’ output into a specified cell of the window. For a sample script please refer to `<install-folder>\_bin\<arch>\sample_video_wall.bat`.

Hardware Requirements

See `<install-folder>\Media_Samples_Guide.pdf`.

For S3D rendering using Microsoft Direct3D* 11.1

- Hardware with Microsoft Direct3D 11.1 support

Software Requirements

See `<install-folder>\Media_Samples_Guide.pdf`.

Stereoscopic 3D Application Programming Interface

Stereoscopic 3D (S3D) Application Programming Interface (API) for Intel® HD Graphics is provided by a standalone library igfx_s3dcontrol.lib under `<install-folder>\igfx_s3dcontrol\lib\<PlatformName>` which can be used independently from **SDK** library.

The IGFXS3DControl C++ contains the methods and structures needed to control S3D display modes. Please see the header file `igfx_s3dcontrol.h` at `<install-folder>\igfx_s3dcontrol\include` for the complete API definition.

A typical flow of an application displaying S3D would be as follows:

- Create `IGFXS3DControl` instance.
- Determine what screen resolutions and refresh rates are currently possible by calling `GetS3DCaps(..)`
- Switch display to S3D mode with specific resolution and refresh rate by calling `SwitchTo3D(..)`
- Create a `Direct3D9Ex` device and pass it via `Direct3D9DeviceManager` to the `IGFXControl->SetDevice(..)`.
- Prepare to create post processing device for left view by calling `SelectLeftView()`.
- Create post processing device, based on standard DXVA2 or DXVA HD.
- Repeat for right view.
- Post process left and right views on correspondent devices to the same render target.
- Present render target.
- Repeat from step 8 for all frames.
- Destroy post processing devices and switch display back to 2D mode by calling `SwitchTo2D()`.
- Destroy `IGFXS3DControl`.

The following restrictions must be considered:

- Direct3D9Ex device must be created after switching to S3D mode.
- Direct3D9Ex device must be created and passed to the IGFXS3DControl before calling SelectView() methods.
- An overlay surface must be acquired for *each* presented frame (left view + right view) via IDirect3DDevice9::GetBackBuffer method.
- The application shall not send S3D content when display is set to 2D mode or vice versa.
- With HDMI 1.4* S3D display mode is limited to 1080p23.98/24, 720p59.94/60 & 720p50 graphics modes. With eDP* all modes are supported (for non-native modes default S3D behavior is to scale OS mode to native mode of panel).
- With multiple monitors S3D library can handle the primary display only.

How to Build the Application

See `<install-folder>\Media_Samples_Guide.pdf`.

Running the Software

See `<install-folder>\Media_Samples_Guide.pdf`.

The executable file requires the following command-line switches to function properly:

h265|h264|mpeg2| vc1|vp8|vp9|mvc|jpeg| capture	Input video type. This is an elementary video stream. The use of option h265 or capture is possible only if corresponding plugins are installed.
-i <InputFile>	Input (compressed) video file, name and path.
-o <Output>	Specifies output (YUV) video file(s), name and path. With MVC, specify the file name without extension to use as a pattern. The sample creates several output files with names such as “filename_N.yuv” according to the number of views in the MVC stream.

The following command-line switches are optional:

-p guid	32-character hexadecimal guid string. Optional for in-box plugins, required for user-decoder ones (HEVC, f.e.).
-path path_to_plugin	Path to decoder plugin (works only in pair with ‘`-p`’ option and requires guid to be specified).
-d3d	Use Microsoft* Direct3D9* surfaces
-d3d11	Use Microsoft Direct3D* 11.1 surfaces
-r	Render output video file to the screen **(must be used in conjunction with –d3d or –d3d11 option)**
-f	Limit rendering fps to certain value
-hw	Use platform-specific implementation of **SDK** (default)
-sw	Use software implementation of **SDK** (platform-specific implementation is used by default)
-di <bob or adi>	Enable deinterlacing:BOB or ADI
-w	Output width
-h	Output height
-jpeg_rgb	Set JPEG Color format to RGB4
-jpeg_rotate	Rotate jpeg frame n degrees (90,180,270)
-nv12, -i420, -rgb4, -p010, -a2rgb10	Output color format (native decoder's output format by default). Note that in case of -i420 option, pipeline uses nv12 color format for output (surfaces format), but during file writing data is converted into i420.
-rgb4_fcr	Set pipeline output format and output file format to RGB4 in full color range
-low_latency	Configures **Decoding Sample** for low latency mode
-calc_latency	Calculate per frame decoding latency
-wall <w h n m t tmo>	Enable video wall scenario. Description of sub-options: • w - Number of video columns • h - Number of video rows

	• n - Order of video cell in video wall, starting from 0 • m - Monitor identifier, integer start from 0 • t - 0 to disable, 1 to enable window's title • tmo - Timeout, in seconds
-scr:w	Screen capture resolution width
-scr:h	Screen capture resolution height
-w	Output width
-h	Output height
-<fourcc>	Output fourcc format for screen capture. Supported fourcc: i420(default), rgb4, p010,a2rgb10
-gpucopy::<on,off>	Enable/disable GPU Copy functionality
-timeout	Decode in a loop not less than specific time in seconds. Performs complete input stream decoding on every iteration. Output file frames amount can be bigger than in input due to buffered frames in decoder. Output file is rewrote every iteration.
-async	Depth of asynchronous pipeline. Default value is 4. Must be between 1 and 20
-?	Print help

Note 1: You need to have **HEVC** installed to run with h265 codec. In case of HW library (-sw key is not specified) it will firstly try to load HW plugin, in case of failure - it will try SW one if available.

Note 2: For video wall sample script please refer to `<install-folder>\_bin\<arch>\sample_video_wall.bat`. The script has a single input parameter which is the name of input file or input directory. In input file mode the script will populate 1 input over the whole video wall. In input folder mode the script will take all files from specified directory in a round to fill the whole video wall.

Note 3: Press '1' while rendering to toggle between full screen and windowed modes

Below are examples of a command-line to execute **Decoding Sample**:

```
sample_decode h264 -i input.264 -o output.yuv –d3d –sw
```

```
sample_decode mvc -i mvc_in_stream.264 -o mvc_out
```

For the second example, if the input stream contains N views, files `mvc_out_0.yuv,…,mvc_out_N-1.yuv` will be created.

HEVC Plugin

HEVC codec is implemented as a plugin unlike codecs such as MPEG2 and AVC. We provide multiple implementations of the HEVC plugin for Decode and Encode – SW, HW and GPU-accelerated. In our samples, depending on the underlying platform, the HW plugin is loaded. If the HW plugin is not supported, the SW plugin gets loaded.

Note 1: The HEVC SW and GPU-accelerated plugins (HEVC Decode SW, HEVC Encode SW, and HEVC Encode GPU-accelerated) are available in the HEVC package which is part of the Intel® Media Server Studio Professional Edition. You can find the available plugins and their IDs from `$MFX_ROOT/include/mfxplugin.h` file.

Note 2: HW Accelerated HW plugins for HEVC Encode and Decode are supported from 6th Generation Intel CoreTM Processors with Intel IrisTM Pro Graphics or Intel HD Graphics 6000/7000+ Series (codename Skylake).

For previous generations (4th and 5th Generation Intel CoreTM Processors with Intel IrisTM Pro Graphics or Intel HD Graphics 4200+ Series), the HEVC SW and GPU-accelerated versions are supported.

Note 3: HEVC Encode has the GPU-accelerated (henceforth referred to as GACC) plugin. To load the plugin, you have to explicitly specify the plugin ID “e5400a06c74d41f5b12d430bbaa23d0b” using the “-p” parameter.

Our samples load the SW HEVC plugins, unless the GUI for the GACC counterparts are specified using “-p” parameter. For example, the following command-lines will use the SW HEVC Decode and Encode plugin respectively:

```
$ sample_decode h265 -i input.265 -o output.yuv
```

```
$ sample_encode h265 -i input.yuv -o output.h265 –w 720 –h 480 –b 10000 –f
 30 –u quality
```

You can enforce a plugin to be loaded by specifying the plugin ID for the same. For example, the below command for sample_decode will load the HW HEVC plugin, and the sample_encode will load the GPU-accelerated plugin:

```
$ sample_decode h265 -i input.265 -o output.yuv –p
 33a61cb4c27454ca8d85dde757c6f8e
```

```
$ sample_encode h265 -i input.yuv -o output.h265 –w 720 –h 480 –b 10000 –p
 e5400a06c74d41f5b12d430bbaa23d0b
```

Known Limitations

- **Decoding Sample** does not fully decode some video streams from a networked folder. Instead, copy the input file to local storage prior to decoding.
- **Decoding Sample** renders output in the simplest way. The rendering window does not support time stamps, aspect ratio. The size of the window is defined by the Windows* constant `CW_USEDEFAULT`.
- Decoding Sample cannot render P010 streams in case of SW library usage.
- `-low_latency` and `–calc_latency` options should be used with H.264 streams having exactly 1 slice per frame. Preferable streams for an adequate latency estimate are generated by **Conferencing Sample**. The options are also effective for JPEG* input streams. For all other input formats application would return an error.
- If overlay is not supported by your hardware or software you won’t be able to render the decoded MVC stream.
- Due to usage of Microsoft* Direct3D9Ex* the application cannot run on operating systems which lack support of this extension (earlier than Microsoft Windows Vista*). To overcome this limitation you may replace `Direct3D9Ex` with `Direct3D9`* in the sample source code and rebuild the sample.
- The *RTP*Payload Format for JPEG-compressed Video* specification stands that there could be 4 combinations of scan types and orders: progressive scan, interlaced scan-Top Field First order, interlaced scan-Bottom Field First order and interlaced scan-single field. In last case frame should be constructed using the single field as top and bottom.

First 3 cases are handled correctly by the sample. However the last case is mistakenly treated as progressive case. To enable correct handling one can set parameter `MFX_PICSTRUCT_FIELD_TFF` on the decoder initialization and duplicate single field data in the input bitstream on the application level.

- S3D rendering using Microsoft Direct3D* 11.1 feature returns “S3D enabled” property of the display to its original state when rendering is finished. If “S3D enabled” was set before sample run the refresh rate of the display would switch to the original value after sample run. If “S3D enabled” was unset before sample run the refresh rate will stay at the new “S3D mode” value after sample run.
- S3D rendering in multiple monitor environment when Intel® Graphics is not primary display is supported only with `–d3d11` option.
- Application may return error for some MJPEG streams decoded with option –low_latency.
- Video wall sample with `–d3d11` option has worse performance than with `–d3d9`. It happens if VerticalCells and HorizontalCells field in video wall script are greater than 1.

- In case of using HEVC plugin (h265 video type), only software implementation of that plugin is used by default (even if you provide -hw option). To force usage of HEVC HW plugin implementation, please use -p option with proper plugin GUID.
- VP8 HW decoding is not working if system memory is used.
- SW HEVC plugin in 10bit mode cannot be used together with HW library VPP. Although library allows that, this is bad practice because additional per-pixel data shift is required. Please use HW HEVC + HW library or SW HEVC + SW library instead.
- Low latency mode (-low_latency) is not compatible with HEVC decoder.
- Sample may exit with wrong status code (MFX_MORE_SURFACE) and messages

  [ERROR], sts=MFX_ERR_MORE_SURFACE(-11), RunDecoding, Unexpected error!!

  [ERROR], sts=MFX_ERR_MORE_SURFACE(-11), main, Pipeline.RunDecoding failed

  This may occur when number of outputted frames becomes equal to number of frames specified by "-n <frames>" command line parameter while current status is MFX_MORE_SURFACE. No negative side effects present, requested number of frames are properly decoded.

Legal Information

other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804
