

Playback Encrypted DivX Stream (exoDivX)

NDK 5.7/MPTK 2.4

Application Note, 12 December 2007

Overview

This example program demonstrates the use of the tmVdecDivx component. It also uses AviRead and DrmDivx, AdecMp3 and AdecAc3 components. This application decodes DivX 3.11, 4.12 and 5.05 streams. Streams may be encrypted or located on a HTTP server.

Usage for compilation

```
build_exe apps/exoDivx.
```

Description

The application builds the following TSSA chain:

```
tmAviread <-|-> tmDrmDivx <-> tmVdecDivx <-> tmVrendGfxVo
      |---| -- tmAdecMp3 -----|
      | -- tmAdecAc3 -----| --- <-> tmArendAo
```

Use the build option `V_OUTPUT=VREND` to create this TSSA chain in makefile. `V_OUPTUT=FILE` replaces the `tmVrendGfxVo` with the `FwriteVideo` component to write the video output to a remote file. If `V_OUTPUT` is not set in the environment, makefile will use the `VREND` value as default.

Instructions

The sample application is configured to decode and render local files and files from HTTP.

You only have to specify the location of the resource you want to play using the command argument `"-url"`.

The naming convention used is the convention defined in RFC 1738.

Examples:

```
-url <optional path>/<myfile.avi>
```

User Options

When the output is set to rendering, the application accepts the following command line arguments:

`[-url <url>]`

With this command, the location of the resource is set. You must specify the location of a resource in order to use the sample application. The syntax of the URL has to be compliant with RFC 1738.

If no input url is specified the application look for file "default.avi" in the local directory.

`[-avilist <url>]`

The url contains a list of avi files, separated by a comma and ending with a semicolon

`[-startstop x]`

How long to play stream before the playback will be stopped automatically (it is playing till end of file). "x" specifies how many seconds the file should be played.

`[-showAviHeaderInfo]`

Displays the information that is extracted from the header of the avi file.

`[-mode|videoOutMode|videooutmode <valid mode>]`

format: <width>x<height>x<framerate><scantype>

Example: 720x480x60p

Example: 720x576x25i (PAL)

`[-videoOutScalingMode <asIs//fullScreen>]`

asIs: render as is, no scaling (default)

fullScreen: full screen, not respecting aspect ratio

`[-fullscreen|letterbox]`

Set the proportions of output video:

fullscreen - full screen, not respecting aspect ratio

letterbox - stretch window without changing aspect ratio

`[-filmDetectionMode <none|trustFlags>]`

none - The operating & deinterlacing modes are set in the command line by the user. Alternatively, the default values will be used.

trustFlags - The operating & deinterlacing modes are controlled by the scan type flags found in the stream (examples, vdfInterlaced, vdfProgressive). Therefore the -operatingMode and -deinterlaceMode options are ignored.

`[-operatingMode <field|frame>]`

field - The operating mode of the video renderer will be set to field based.

frame - The operating mode of the video renderer will be set to frame based.

`[-deinterlaceMode|deinterlacemode <valid mode>]`

mode:

none|median|majority2|majority3|majority3enh|majority3dyn|majority3dynenh|auto

`[-eddiLevel|eddilevel <valid level>]`

level: off|NMcLowNoiseSafe|NMcLowNoiseNormal|NMcLowNoiseAggressive|NMcSafe|NMcNormal|NMcAggressive|McLowNoiseSafe|McLowNoiseNormal|McLowNoiseAggressive|McSafe|McNormal|McAggressive

`[-enablegracefuldegradation]`

Enabling graceful degradation allows the application to skip decoding some of the video frames and thus ensure that video decoding completes in real-time.

`[-outputwindow <ul.x ul.y lr.x lr.y>]`

Specifies the dimensions of the output video window using four integer values: x coordinate of upper left corner, y coordinate of upper left corner, x coordinate of lower right corner, y coordinate of lower right corner. Can be used instead of `-videoOutScalingMode`, `-fullscreen` and `-letterbox` switches.

`[-trickmode]`

Enables usage of trick modes (jump, gettime, stop, (re)start, pause and resume).

The menu that will pop up is pretty straightforward, so no further explanation here.

`[-UseSemiplanar]`

Use semiplanar output to renderer iso. planar output.

`[-UsePostprocessing <level>]`

Use postprocessing level as defined by DivX. Level 0 means no postprocessing, level above 49 means full postprocessing.

`[-operatingMode | operatingmode <field | frame>]`

If `filmDetectionMode` is set to `filmDetectionModeTrustFlags` then this option will set video renderer in following operating mode:

field - field mode

frame - frame mode

`[-selectAudioTrack <track>]`

playback with a given audio track number. If the application is running in trick mode, the user can change the audio track later. 0 means it plays mute. Default value is 1. If a given of default track number is not a valid audio track (such as PCM audio, the application will play the first valid audio track.

`[-selectSubtitleTrack <track>]`

This command allows the start of playback with a given subtitle track number. If the application is running in trick mode, the user can change the audio track later. 0(default value) means do not display any subtitle.

`[-d drm_input_file]`

Specifies the file where the content of DRM non-volatile memory is stored.

`[-drm_reset]`

Reset DRM memory.

`[-drm_get_reg_code]`

Generates a registration code to be sent to the VOD provider.

Notes

If the same option is set several times in the command line, the last setting will be used.

If two conflicting options are set, the last one will be used. An example of conflicting options: `-fullscreen` and `-letterbox`.

When the application is running, you can only give commands for trick modes, when you have enabled this with the command line argument `"-trickmode"`.

When output is set to file the following options are available. Parameters:

<code>-url</code>	Input file name in the local directory or full path or URL.
<code>-o</code>	Video output file name, local file name, optionally with path.
<code>-oa</code>	Audio output file name, local file name, optionally with path.
<code>-help</code>	A short help screen with usage.

If no input file name is specified the application look for file `"default.avi"` in the local directory.

If no output file name is specified the output will be stored in the local directory under name same as input file but extension replaced for `.yuv`. If no input file was specified also the output file name will be `default.yuv`.

Expected output

If your stream contains video, then the video should appear in the center of the screen on a black background. Mp3 or Ac3 audio, when present in the stream, should be heard.

In the console, the application prints only name of input and output files. Upon completion the application writes `"Application complete"`.

In the debug and trace versions you can adjust level of output prints from the application and the components. This is called debug information. Debug information can be seen in the DP buffer when the application is executed from a `dvpmon` program. The debug prints of the application can be redirected to the console. However, this is discouraged.

Note: Using the `DBG_LEVEL_4`, `DBG_LEVEL_5`, `DBG_INTERNAL_ENTER` and `DBG_INTERNAL_LEAVE` levels of debug prints for components may slow down the system as the amount of debug prints may be overwhelming.

```
exolDivx: TriMedia DivX stream decoding application, 0L layer. Version 1.2
exolDivx: CheckArgcv: Input file specified as
file:///E:\Testing\HTS\avi\divx1_smpte720x480_505sp_prog_4mbps.avi
Application completed
```

```
CRunTime : TriMedia # 0 Program Exit Code [0]
```

Frequently asked questions

- ***I cannot convince the application to open a file.***

Make sure the pathname and filename are correct. Use a RFC-1738 compliant or windows-style path (use back-slash).
- ***The application hangs after a first play.***

Reset the NREF board and clear memory before starting the new play.
- ***Video screen flickers after application is complete.***

Known error. Reset the NREF board. See release notes for details.
- ***Green or red bars (patches) are seen at the start of playback.***

Known issue related to the video render initialization. This is not an issue of a video decoder. After beginning of playback you may jump back to the start of the stream with the `j0` command in a trick mode and then the stream should be playing without any artifact.
- ***Blocking artifacts are seen on a stream.***

Some artifacts can already be coded in the stream itself. This is due to limitations on the bit rate of a stream. Please, view this stream with a DivX player on a Windows-based PC to check if you see the same artifacts there.
- ***Video is too slow.***

Make sure that you are running application in retail mode and your input local drive or network are fast enough to provide input data and the peak bit rate of the stream.