MVX2FileStream

Generated by Doxygen 1.8.16

Chapter 1

Mantis Vision: MVX2FileStream

A plugin for progressive streaming of Mvx2-formatted files.

Description

The plugin is internally based on facilities of curl library (https://curl.se/). In case of Android build, curl library further depends on OpenSSL library (https://www.openssl.org/). This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

Table of Contents

Chapter 2

Release Notes

1.0.0

Initial version.

Plugin

- **1.0.0_P1** | updated `curl` 3rdparty dependency to version 7.75.0
- **1.0.0_P2** | updated `MVCommon` 3rdparty dependency to version 4.0.0
- **1.0.0_P3** | updated for `Mvx2` framework version 5.0.0

Documentation

- **1.0.0_D1** | added 'release notes' section
- **1.0.0_D2** | added a section for each plugin's filter

1.1.0

Added caching curl downloader which requests bigger chunks of data containing more frames and caches them to a local file specified by a path. Added new source filter for directly reading byte arrays from provided pointer.

Plugin

- **1.1.0_P1** | added option to cache the downloaded data to a local file and improved download performance
- **1.1.0_P2** | added option for caching downloader to block playback when slow download is detected until enough data is downloaded to ensure smooth playback at current average download speed
- **1.1.0_P3** | added new source filter for reading mvx data from provided byte array pointer
- **1.1.0_P4** | updated `curl` 3rdparty dependency with newer version of `openSSL` version 1.1.1s

Chapter 3

Reading filters

3.1 SourceRemoteMVX2FileSyncReader

A source filter for synchronous reading of frames from a remote MVX-formatted binary file. A file to read may reside on a remote location and it must be accessible via one of standard data transport protocols, for example `http` or `https`. A crucial limitation of the reader is that tranport protocol being used for accessing files must support specification of data to download via data ranges.

The reader downloads/reads frames synchronously from the pipeline playback thread when they are needed.

The reader is able to process both legacy MVX2 files, which do not contain a *Look-up Table (LUT)* at their end, and new files with the *LUT* attached to their end. However, in order to utilize the *LUT* of a file (if it contains one), the transport protocol used for accessing the file must currently be one of:

- *HTTP/1.0*
- *HTTP/1.1*

If a different protocol is used, the file is treated as if it did not contain a *LUT*.

Depending on the presence of a *LUT* in the remote file, the reader either supports all *playback modes* (when *LUT* is present) or is limited to the *realtime playback mode* (when *LUT* is not available). The reason for the limitation is that without a *LUT* the information about precise location of individual frames is missing and the file can only be treated as a stream, from which subsequent frames are read.

In a *realtime playback mode* a difference in the reading behaviour when a *LUT* of a remote file is available, and when it is not, lies in the actual frames downloaded/read and pushed to the pipeline for next processing:

- when *LUT* is utilized, frames can be ordered in the file randomly and they are always correctly downloaded based on information stored in the *LUT*,
- when *LUT* is not utilized, frames are read from the file sequentially with their frame numbers being ignored. The same applies also for atoms of the frames - subsequent atoms are treated as a single frame even if that may not truly be the case.

Similarly as in regular local MVX2 file readers, the responsibility of the remote file reader is also loosening/duplication of the data optimized by writers from the first atom of each stream to all its atoms.

Type GUID

EE9E6123-43B5-4F9F-BE48-9BC4AFF947AF

Parameters

- **"MVX File URL"**
  Specifies an URL of the remote file to read.
- **"Look-up Table Present"**
  An indication about a presence of a *Look-up Table (LUT)* in the file and its employment status. *LUT* presence in the file preconditions a support for playback modes other than *realtime*.
- **"Employ Look-up Table"**
  Specifies whether a *Look-up Table* shall be employed if present in the file. Allows to disable *LUT* utilization even if it is available.
- **"Realtime Stream Ended"**
  An indication (in a *realtime playback mode*) that the end of the file (stream) was reached. Allows to stop the playback at an appropriate moment, since *realtime playback mode* continues running indefinitely even when the last frame of a stream was reached already.
- **"Frame Cache Type"**
  Specifies a type of frame cache to be used for caching already downloaded frames, so when the same frames are needed again, they do not have to be downloaded again. It must be one of:
  - *"None"* - frames are not cached (i.e. if the same frame is needed multiple times, it will be re-downloaded every time),
  - *"Memory"* - frames are kept in the memory.
- **"SSL Verification Mode"**
  When a TLS/SSL connection is negotiated, specifies requirements for the server certificate verification. If the verification fails, the connection is dropped. It must be one of:
  - *"Certificate&Host"* - most secure - server's certificate must be signed by one of the certificates in the CA bundle, and the server's hostname must match value in the ceritificate,
  - *"Certificate"* - server's certificate must be signed by one of the certificates in the CA bundle, but the server's hostname can differ from value in the ceritificate,
  - *"None"* - least secure - verification of the server's certificate is skipped.

3.2 SourceRemoteMVX2FileAsyncRealtimeReader

A source filter for asynchronous reading of frames from a remote MVX-formatted binary file. A file to read may reside on a remote location and it must be accessible via one of standard data transport protocols, for example `http` or `https`. A crucial limitation of the reader is that tranport protocol being used for accessing files must support specification of data to download via data ranges.

The reader downloads/reads frames asynchronously from a thread that is parallel to the pipeline playback thread. At each moment there are multiple future frames scheduled for downloading, so that the frames are ready when they are actually needed. An ultimate limitation of the reader is that it can only operate in a *realtime playback mode*. For support of other playback modes check SourceRemoteMVX2FileMutateAsyncReaderBackend.

The reader is able to process both legacy MVX2 files, which do not contain a *Look-up Table (LUT)* at their end, and new files with the *LUT* attached to their end. However, in order to utilize the *LUT* of a file (if it contains one), the transport protocol used for accessing the file must currently be one of:

- *HTTP/1.0*
- *HTTP/1.1*

If a different protocol is used, the file is treated as if it did not contain a *LUT*.

A difference in the reading behaviour when a *LUT* of a remote file is available, and when it is not, lies in the actual frames downloaded/read and pushed to the pipeline for next processing:

- when *LUT* is utilized, frames can be ordered in the file randomly and they are always correctly downloaded based on information stored in the *LUT*,
- when *LUT* is not utilized, frames are read from the file sequentially with their frame numbers being ignored. The same applies also for atoms of the frames - subsequent atoms are treated as a single frame even if that may not truly be the case.

Similarly as in regular local MVX2 file readers, the responsibility of the remote file reader is also loosening/duplication of the data optimized by writers from the first atom of each stream to all its atoms.

Type GUID

4295B4C6-E58E-47C4-9272-9A8F49EEBDD2

Parameters

- **"MVX File URL"**
  Specifies an URL of the remote file to read.
- **"Look-up Table Present"**
  An indication about a presence of a *Look-up Table* in the file and its employment status.
- **"Employ Look-up Table"**
  Specifies whether a *Look-up Table* shall be employed if present in the file. Allows to disable *LUT* utilization even if it is available.
- **"Realtime Stream Ended"**
  An indication that the end of the file (stream) was reached. Allows to stop the playback at an appropriate moment, since *realtime playback mode* continues running indefinitely even when the last frame of a stream was reached already.
- **"Buffer Size"**
  Specifies a size of the frames-buffer populated from a downloading thread and depopulated from the pipeline thread. Represents a maximum count of pre-downloaded frames. In some modes (see **"Mode"** parameter) represents also a size of the frame-requests-buffer.
- **"Mode"**
  Specifies a frames-requesting mode manifested when downloading of frames is slower than playback in a pipeline thread. It must be one of:
  - *"Blocking"* - pipeline thread is blocked, if buffer of the frame requests is full, until a frame is downloaded,
  - *"Nonblocking-Bounded"* - pipeline thread is not blocked, if buffer of the frame requests is full - the oldest not yet addressed requests are discarded,
  - *"Nonblocking-Unlimited"* - pipeline thread is not blocked, because the buffer of the frame requests is unlimited.

- **"SSL Verification Mode"**
  When a TLS/SSL connection is negotiated, specifies requirements for the server certificate verification. If the verification fails, the connection is dropped. It must be one of:
  - *"Certificate&Host"* - most secure - server's certificate must be signed by one of the certificates in the CA bundle, and the server's hostname must match value in the ceritificate,
  - *"Certificate"* - server's certificate must be signed by one of the certificates in the CA bundle, but the server's hostname can differ from value in the ceritificate,
  - *"None"* - least secure - verification of the server's certificate is skipped.

3.3 SourceRemoteMVX2FileMutateAsyncReaderBackend

A source filter for asynchronous reading of frames from a remote MVX-formatted binary file. A file to read may reside on a remote location and it must be accessible via one of standard data transport protocols, for example `http` or `https`. A crucial limitation of the reader is that tranport protocol being used for accessing files must support specification of data to download via data ranges.

A difference from SourceRemoteMVX2FileAsyncRealtimeReader is that this source supports also *playback modes* other than *realtime*. Unfortunately, due to Mvx2 core framework limitations, this feature is achieved at a cost of a bit more complicated pipeline setup. The solution is based on a cooperation of two filters:

- *SourceRemoteMVX2FileMutateAsyncReaderBackend* (this filter) - contains all the reader parameters, but does not truly read frames of the remote file. Instead of that, pushes empty frames to the pipeline,
- MutateRemoteMVX2FileAsyncReader - performs actual remote file frames reading. It drops all its (empty) input frames and replaces them with the actual remote frames.

The implementation enforces that the two filters are injected to a pipeline directly one after the other (there can be no additional filter in between them). The solution downloads/reads frames asynchronously from a thread that is parallel to the pipeline playback thread.

The solution is able to process both legacy MVX2 files, which do not contain a *Look-up Table (LUT)* at their end, and new files with the *LUT* attached to their end. However, in order to utilize the *LUT* of a file (if it contains one), the transport protocol used for accessing the file must currently be one of:

- *HTTP/1.0*
- *HTTP/1.1*

If a different protocol is used, the file is treated as if it did not contain a *LUT*.

Depending on the presence of a *LUT* in the remote file, the solution either supports all *playback modes* (when *LUT* is present) or is limited to the *realtime playback mode* (when *LUT* is not available). The reason for the limitation is that without a *LUT* the information about precise location of individual frames is missing and the file can only be treated as a stream, from which subsequent frames are read.

In a *realtime playback mode* a difference in the reading behaviour when a *LUT* of a remote file is available, and when it is not, lies in the actual frames downloaded/read and pushed to the pipeline for next processing:

- when *LUT* is utilized, frames can be ordered in the file randomly and they are always correctly downloaded based on information stored in the *LUT*,
- when *LUT* is not utilized, frames are read from the file sequentially with their frame numbers being ignored. The same applies also for atoms of the frames - subsequent atoms are treated as a single frame even if that may not truly be the case.

Similarly as in regular local MVX2 file readers, the responsibility of the remote file reader is also loosening/duplication of the data optimized by writers from the first atom of each stream to all its atoms.

Type GUID

196E7756-F4F6-4BA5-93C2-77BBC0CEA020

Parameters

- **"MVX File URL"**
  Specifies an URL of the remote file to read.

- **"Look-up Table Present"**
  An indication about a presence of a *Look-up Table (LUT)* in the file and its employment status. *LUT* presence in the file preconditions a support for playback modes other than *realtime*.

- **"Employ Look-up Table"**
  Specifies whether a *Look-up Table* shall be employed if present in the file. Allows to disable *LUT* utilization even if it is available.

- **"Realtime Stream Ended"**
  An indication (in a *realtime playback mode*) that the end of the file (stream) was reached. Allows to stop the playback at an appropriate moment, since *realtime playback mode* continues running indefinitely even when the last frame of a stream was reached already.

- **"Buffer Size"**
  Specifies a size of the frames-buffer populated from a downloading thread and depopulated from the pipeline thread. Represents a maximum count of pre-downloaded frames. In some modes (see **"Mode"** parameter) represents also a size of the frame-requests-buffer.

- **"Mode"**
  Specifies a frames-requesting mode manifested when downloading of frames is slower than playback in a pipeline thread. It must be one of:

  - *"Blocking"* - pipeline thread is blocked, if buffer of the frame requests is full, until a frame is downloaded,
  - *"Nonblocking-Bounded"* - pipeline thread is not blocked, if buffer of the frame requests is full - the oldest not yet addressed requests are discarded,
  - *"Nonblocking-Unlimited"* - pipeline thread is not blocked, because the buffer of the frame requests is unlimited.

- **"Frame Cache Type"**
  Specifies a type of frame cache to be used for caching already downloaded frames, so when the same frames are needed again, they do not have to be downloaded again. It must be one of:

  - *"None"* - frames are not cached (i.e. if the same frame is needed multiple times, it will be re-downloaded every time),
  - *"Memory"* - frames are kept in the memory.

- **"SSL Verification Mode"**
  When a TLS/SSL connection is negotiated, specifies requirements for the server certificate verification. If the verification fails, the connection is dropped. It must be one of:

  - *"Certificate&Host"* - most secure - server's certificate must be signed by one of the certificates in the CA bundle, and the server's hostname must match value in the ceritificate,
  - *"Certificate"* - server's certificate must be signed by one of the certificates in the CA bundle, but the server's hostname can differ from value in the ceritificate,
  - *"None"* - least secure - verification of the server's certificate is skipped.

3.4 MutateRemoteMVX2FileAsyncReader

A mutate filter for asynchronous reading of frames from a remote MVX-formatted binary file. The filter is part of a duo-filter solution described in detail at SourceRemoteMVX2FileMutateAsyncReaderBackend. The filter can only be injected to a pipeline right behind the backend source filter.

Type GUID

AFACB6A7-BD00-416E-B1AA-A10615C28CF7

Parameters

All the parameters are set on the backend filter (SourceRemoteMVX2FileMutateAsyncReaderBackend).

Chapter 4

SourceMvx2ByteArrayReader

A source filter for reading frames from an MVX-formatted byte array. Pointer to the byte array is passed as parameter as int64_t. The source filter does not have the responsibility to delete the array.

Type GUID

9E6DF6ED-AFD8-419C-A323-2F6B13DD268A

Parameters

- **"MVX Byte Array Pointer"**
  Pointer to the byte array as int64_t.
- **"MVX Byte Array Length"**
  Length of the byte array.