H264

Generated by Doxygen 1.8.16

1 Mantis Vision: H264 **1**

2 Release Notes **3**

3 Compressor filters **7**

3.1 MutateH264Compressor 7

4 Data layer types **9**

4.1 DataTypeH264Data 9

4.2 DataTypeH264Texture 9

5 Decompressor filters **11**

5.1 MutateH264Decompressor 11

Chapter 1

Mantis Vision: H264

A plugin for compression and decompression of textures using H.264 standard.

Table of Contents

- Release Notes

Chapter 2

Release Notes

1.0.0

Initial version.

Plugin

- **1.0.0_P1** | updated for `Mvx2` framework version 3.0.0
- **1.0.0_P2** | MutateH264Compressor only supported on Windows platform
- **1.0.0_P3** | MutateH264Decompressor supported on Windows, Android, LuminOS, iOS and MacOS platforms
- **1.0.0_P4** | legacy data layer DataTypeH264CompressedTexture replaced by a modern DataTypeH264Data as a result of compression, but remains supported for decompression

Documentation

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

2.0.0

Plugin

- **2.0.0_P1** | updated for `Mvx2` framework version 4.1.0
- **2.0.0_P2** | removed DataTypeH264CompressedTexture which is implemented directly in `Mvx2` framework since version 4.1.0

2.1.0

Plugin

- **2.1.0_P1** | updated MacOS implementation of MutateH264Decompressor to use hardware-accelerated H264 decoder if available
- **2.1.0_P2** | refactored Windows MediaFoundation-based implementation of MutateH264Compressor to first search for asynchronous hardware-accelerated encoder and only in case there is none use synchronous software encoder
- **2.1.0_P3** | Windows MediaFoundation-based implementation of MutateH264Compressor does not encode textures that have different dimensions than the texture whose dimensions were used for the native encoder's initialization (i.e. the first texture processed) - empty H.264 data are generated for such textures

2.2.0

Plugin

- **2.2.0_P1** | fixed implementation of MutateH264Decompressor for Apple platforms (iOS, MacOS, MacOS-↩
Unity) so it generates NV12 textures with correct dimensions - the resulting textures were invalid because of the bug
- **2.2.0_P2** | fixed implementation of MutateH264Decompressor for all platforms, which was occasionally failing to generate valid NV12 textures because of incorrect manipulation with data timestamps
- **2.2.0_P3** | fixed implementation of MutateH264Compressor for all platforms, which was occasionally failing to generate valid H264 data because of incorrect manipulation with data timestamps
- **2.2.0_P4** | updated implementation of MutateH264Decompressor for Apple platforms (iOS, MacOS, MacO↩
S-Unity) to avoid deadlocks that occur on some combinations of Apple devices and OSes when stopping the filter
- **2.2.0_P5** | fixed implementation of MutateH264Decompressor to initialize **"Data layer to decompress"** parameter even when a preceding filter in the graph does not add anything to the aggregated output profile (e.g. all filters derived from MVX::Target)
- **2.2.0_P6** | fixed implementation of MutateH264Compressor to initialize **"Data layer to compress"** parameter even when a preceding filter in the graph does not add anything to the aggregated output profile (e.g. all filters derived from MVX::Target)
- **2.2.0_P7** | fixed invalid values in multiple logged messages
- **2.2.0_P8** | fixed a bug of MutateH264Decompressor on Android and LuminOS platform which prevented successful decompression of a first few data, because the decompressor's initialization was not yet done. The fixed implementation gives the decompressor a reasonable time to complete its initialization and proclaims that the decoder failed to initialize in case it does not manage to do so in the given time
- **2.2.0_P9** | introduced a **"Start decoding timeout"** parameter to MutateH264Decompressor, which limits how long the decompressor is allowed to wait for resources in order to start decoding a compressed data

2.3.0

Plugin

- **2.3.0_P1** | updated `MVCommon` 3rdparty dependency to version 3.0.0
- **2.3.0_P2** | updated for `Mvx2` framework version 5.0.0

2.4.0

Plugin

- **2.4.0_P1** | updated `MVCommon` 3rdparty dependency to version 4.0.0
- **2.4.0_P2** | updated for `Mvx2` framework version 6.0.0

Documentation

- **2.4.0_D1** | introduced PDF documentation as an alternative to the HTML one:
  - `doc/H264.pdf`

2.5.0

Plugin

- **2.5.0_P1** | added legacy DataTypeH264Texture data type equivalent to the previously removed DataType↩ H264CompressedTexture (see **2.0.0_P2**)
- **2.5.0_P2** | added support for decompressing DataTypeH264Texture to the MutateH264Decompressor

2.5.1

Plugin

- **2.5.1_P1** | fixed a bug of MutateH264Decompressor on Android platform which prevented successful decompression of data on `Oculus Quest 1` device
- **2.5.1_P2** | fixed implementation of MutateH264Decompressor for Android platform so it does not generate errors about 'invalid returning of buffers'

2.5.2

Plugin

- **2.5.2_P1** | fixed a bug of MutateH264Decompressor on Android platform which led to incorrect extraction of decoded NV12 textures from the internal decoder on `Google Pixel (4 and 4a)` devices
- **2.5.2_P2** | fixed a bug of MutateH264Decompressor which led to incorrect extraction of decoded NV12 textures from the internal decoder in case the textures' dimensions were not multiples of 16 bytes, on multiple platforms:
  - Windows
  - Android
  - LuminOS

2.5.3

Plugin

- **2.5.3_P1** | redesigne of MutateH264Decompressor on Android platform to utilize AImageReader to properly deal with COLOR_FormatYUV420Flexible format

Chapter 3

Compressor filters

MutateH264Compressor

3.1 MutateH264Compressor

A mutate filter for compression of NV12 textures using H.264 standard.

Supported platforms:

- **Windows** - an asynchronous/hardware(preferred) or synchronous implementation based on Microsoft Media Foundation framework.

If for any reason a compression of an NV12 texture fails, the filter creates and pushes an empty H264 data to atoms to secure consistency.

Type GUID

7549B729-3804-4F6D-9741-A6000C882635

Parameters

- **"Drop original data"**
  Indicates whether original NV12 texture shall be dropped from atoms after compression or preserved.
- **"Compressed atoms buffer size per stream"**
  Specifies a size of frames buffer per stream and determines how many frames can be compressed at the same time in case of asynchronous implementations. In case the buffer is full, data of additional atoms are not compressed and empty H264 data are created.
- **"Data layer to compress"**
  Identifies a data layer to compress via its data profile in case there are multiple NV12 textures present in the same atom.
- **"Compression profile"**
  A compression profile - must be one of: Baseline, Main, High.
- **"Quality"**
  A quality of compressed data - a value in range $<$0.0f, 1.0f$>$.
- **"CABAC enabled"**
  Indicates whether Context-adaptive binary arithmetic coding (CABAC) algorithm, as a form of entropy encoding, shall be used.
- **"All frames keyframes"**
  Indicates whether all frames shall be compressed as H264 key frames.

Chapter 4

Data layer types

DataTypeH264Data DataTypeH264Texture

4.1 DataTypeH264Data

A data layer containing H264-compressed data.

Type GUID

F241B0E2-FC34-461A-8A0E-E609F0B8C2A5

Status properties

- **"NV12 Width"**
  Indicates a width of an NV12 texture whose compressed data are stored in the layer.
- **"NV12 Height"**
  Indicates a height of an NV12 texture whose compressed data are stored in the layer (typically 1.5 times greater than an equivalent RGB texture).
- **"Size"**
  Indicates a size of the data in bytes.

4.2 DataTypeH264Texture

A legacy data layer containing H264-compressed data. The data layer type is only preserved for compatibility reason.

Type GUID

E4658894-3E9D-4F4A-A4F3-B4827918CBC4

Status properties

- **"NV12 Width"**
  Indicates a width of an NV12 texture whose compressed data are stored in the layer.
- **"NV12 Height"**
  Indicates a height of an NV12 texture whose compressed data are stored in the layer (typically 1.5 times greater than an equivalent RGB texture).
- **"Size"**
  Indicates a size of the data in bytes.

Chapter 5

Decompressor filters

MutateH264Decompressor

5.1 MutateH264Decompressor

A mutate filter for decompression of H.264 data into NV12 textures.

Supported platforms:

- **Windows** - a blocking synchronous implementation based on Microsoft Media Foundation framework,
- **Android** - a non-blocking synchronous implementation based on Android's Media Codec system, allowing multiple frames to be decompressed at the same time,
- **Lumin OS/Magic Leap** - a non-blocking synchronous implementation based Magic Leap's Media Codec system, allowing multiple frames to be decompressed at the same time,
- **MacOS, iOS** - a non-blocking asynchronous implementation based on Apple's VideoToolbox framework, allowing multiple frames to be decompressed at the same time.

If for any reason a decompression of an H264 data fails, the filter creates and pushes an empty NV12 texture to atoms to secure consistency.

Type GUID

57A38625-A0DD-46C3-B030-51B044301E45

Parameters

- **"Drop compressed data"**
  Indicates whether original compressed H264 data shall be dropped from atoms after decompression or preserved.
- **"Decompressed atoms buffer size per stream"**
  Specifies a size of frames buffer per stream and determines how many frames can be decompressed at the same time in case of asynchronous implementations. In case the buffer is full, the missing not-yet-decompressed data of the oldest buffered frame are replaced with empty NV12 textures and released to the pipeline to make a place for the new frame.
- **"Start decoding timeout"**
  Specifies a reasonable timeout for the decompressor to wait at most (per compressed data) until it is successfully given resources by an underlying system in order to start the decoding process. The value is specified in microseconds and a valid range is `0-4294967295`. The default value is `3000`.
- **"Data layer to decompress"**
  Identifies a data layer to decompress via its data profile in case there are multiple H264 data present in the same atom. The filter is able to decompress data from the DataTypeH264Data, the legacy MVX::DataType↩H264CompressedTexture and another legacy DataTypeH264Texture data layers.