Mvx2

Generated by Doxygen 1.8.16

Chapter 1

Mantis Vision: Mvx2

A framework for creation and execution of data-processing pipelines and graphs.

Description

Mvx2 is a collection of base classes, as well as utility classes, which together provide a way to compose, customize and execute data-processing pipelines and graphs.

Table of Contents

Supported Platforms

Currently the framework works on these platforms:

- Windows (x64),
- Linux (x64, arm64)
- MacOS (x64),
- Android (armeabi-v7a, arm64-v8a),
- iOS (arm64) and
- LuminOS.

Chapter 2

Mvx2API

An API for compilation and execution of data-processing graphs.

Description

Mvx2API is a collection of classes and functions which together form a public application programming interface (API) of Mvx2 framework, more specifically a part of the framework's API which enables composition, compilation and execution of data-processing Mvx2 graphs.

Architecture

The API consists of multiple sets of classes for different purposes. The three core sets of classes and the actions they allow to perform are:

- **graph nodes** - represent basic building blocks of processing graphs and they are responsible for actual data processing,
- **graph builders** - provide ways to create graphs from graph nodes and
- **graph runners** - provide ways for execution of graphs.

Each of the actions is described in more detail in the subsequent text.

Physically the API is designed with a modularity in mind. This means that all the core features are implemented as part of Mvx2 module, but there are also extension modules (for example *Mvx2BasicIO*), which add additional features to the overall API. The benefit is that various domains of additional features are organized in standalone and independent modules and an application built on top of Mvx2API does not have to deal with features it does not need simply by not using specific modules.

Workflow

The basic usage workflow of Mvx2API is as follows:

1. Create a graph builder.
2. Append a list of specific graph nodes to the graph builder.
3. Keep references to the graph nodes in order to control their behaviour later.
4. Compile a graph from the graph builder.
5. Wrap the graph in one of the available graph runners.
6. Use the graph runner to control the execution of the graph.

Graph Builders

Graph builders are responsible for creation of graphs. Even though the terminology uses the **graph** term already, current implementation of Mvx2 framework does not actually support true graphs yet. Mvx2API therefore also only allows creation of single-path graphs, i.e. **pipelines**, via its graph builders.

The basic implementation of a graph builder is the Mvx2API::ManualGraphBuilder class. An object of this class can be used to append any number of graph nodes to a graph being built. The graph nodes together form a sequence of nodes, which in a modular way process frames - the sequence of specific graph nodes determines what frames (i.e. what frame data) there are at the graph's end.

There are multiple rules that have to be satisfied when building any graph. One of them states that there has to be a **source** graph node at the beginning of the graph. Various sources shall be found in Mvx2API's extension modules. For example *Mvx2BasicIO* extension module provides a graph node that is able to read Mvx2-formatted files, extract their data and provide them for the processing by a graph (see Mvx2FileReaderGraphNode). Consult documentation of Mvx2 framework, the part about filters and plugins, for details about various types of filters.

Behind a source graph node there can be any number of graph nodes of any type (except source) appended to the graph, but current limitation of Mvx2 framework is that the last graph node has to be a **target** node. *Mvx2BasicIO* extension module provides for example a graph node that is able to store processed frames into an Mvx2-formatted file (see Mvx2FileWriterGraphNode). Again see Mvx2 framework documentation for details.

Graph Runners

Graph runners are responsible for execution of graphs. There are three different implementations of graph runners available, each providing different means for controlling the execution:

- **auto sequential graph runner**- runs automatically and sequentially, a client only has to trigger the playback and basically does not care about the rest. See class Mvx2API::AutoSequentialGraphRunner for details.
- **manual sequential graph runner** - a client has to trigger each update of a graph individually, but frames are also processed sequentially. See class Mvx2API::ManualSequentialGraphRunner for details.
- **random-access graph runner** - a client has to trigger each update of a graph individually, but the frame to be processed during this update has to be specified explicitly. See class Mvx2API::RandomAccessGraphRunner for details.

Which graph runner implementation to use in an application depends on its specific needs.

In case of auto sequential and manual sequential graph runners, the frame to be processed next is determined by **playback mode** (RunnerPlaybackMode), which is specified at the beginning of the playback. There are multiple playback modes available but some of them can only be used in some cases and in some cases only one of the playback modes is usable. Which playback modes can be used in what situation depends on the type of source graph node used by a graph. All sources can generally be categorized as either *live* or *offline*. Examples of live sources would be any kind of images-grabbing cameras or network receivers. Example of the offline source would be a file reader. Live sources can only work with **realtime** playback mode, because other playback modes do not make sense for them. Offline sources on the other hand can easily work also with special playback modes like **ping-pong**, **loop** or **backward** playback modes.

Frame Data Access

The essential feature of Mvx2API is access to data of processed frames. The way to do so is by using one of the two available graph node implementations: Mvx2API::FrameAccessGraphNode and Mvx2API::AsyncFrameAccessGraphNode. Since access is implemented using graph nodes architecture, it is possible and completely valid to add multiple frame-accessing graph nodes to a single graph at various places, which makes it possible to access data of a frame at different stages of its processing.

The difference between the two frame-accessing graph nodes is in the way how the frame data are accessed. **FrameAccessGraphNode** caches last processed frame and a client has to manually call its function to get the frame. **AsyncFrameAccessGraphNode** works asynchronously - a client has to create the graph node instance with a custom **frames listener** object (Mvx2API::FrameListener). The graph node calls this listener's callback function every time there is a new processed frame.

Selection of the implementation depends on specific needs of an application, but it does not make much sense to use the synchronous implementation when the graph's execution is controlled by an auto sequential graph runner because of its asynchronous nature.

Anyways, in both cases a frame that is received is an object of Mvx2API::Frame class. This class is a starting point for accessing frame data. There is no generic way for accesing just any frame data - instead of that, specialized features of Mvx2API and its extension modules shall be used to extract specific data. For example *Mvx2API* itself provides extractors for accessing mesh data, texture data, audio data et cetera (see below).

The Mvx2 framework supports **multi-stream** processing and for this reason even Mvx2API provides ways to access data of different streams. The API implements this feature through frames - at any point in time there is exactly one stream marked as active and any data extractions performed over a frame are performed over this specific active stream. API of frames naturally contains functions which deal with multiple streams - it is possible to query for number of actual streams in a frame and also to activate a stream at an index.

Data Extractors

The API provides multiple extractors of specific data layers of processed frames:

- **Mvx2API::FrameAudioExtractor** for extraction of audio data,
- **Mvx2API::FrameTextureExtractor** for extraction of texture data in various formats (including depth-maps and IR textures),
- **Mvx2API::FrameMeshExtractor** for extraction of mesh-related data (vertex positions, normals, etc.) and
- **Mvx2API::FrameMiscDataExtractor** for extraction of other useful frame data.

As a first argument the extraction functions of the extractors always expect a reference to Mvx2API::Frame object, which data shall be extracted from. In case the Mvx2API::Frame object contains multiple data streams, actual data are always extracted from the stream which is active at the time of the extraction.

A purpose guid parameter can also be passed to each data extraction function, so in case there are multiple data of the same type in a frame, the specific one can be extracted assuming its purpose guid is known to a caller. If no purpose guid is specified, any of the available data are picked and returned.

Mesh Data

When mesh data are extracted from a Mvx2API::Frame object, they are returned in a form of Mvx2API::MeshData object. The object groups together vertex **positions**, vertex **normals**, vertex **colors**, vertex **UV coordinates** and vertex **indices** data. Not all of them however must necessarily be present at all times - some of the data may not be present in the frame at all, other times, even when data are present, they may have different purpose guid assigned than what was requested. If no purpose guid is explicitly specified when extracting mesh data, data layers with any purpose guid are picked and returned, even in case they do not have the same purpose guid assigned.

There is also an utility class Mvx2API::MeshSplitter, which can be used in case the original mesh data returned after extraction are too big for an application. The utility is able to split original meshes into smaller submeshes, with explicitly specified maximal count of vertices.

Manual Data Sources

There are two special graph nodes implemented in Mvx2API, which make it also possible to pass frames processed and extracted from one graph to another one. This way multiple graphs can be chained together with a bit of extra glue code. The two source graph nodes are Mvx2API::ManualLiveFrameSourceGraphNode and Mvx2API::ManualOfflineFrameSourceGraphNode.

The difference between the two is that the *offline* version has to be filled with all frames in advance, because it does not accept more frames after it was added to a graph. The other one, *live* version, must only have its properties initialized in advance, but it accepts additional frames during the graph playback. The consequence is that *live* version can be considered *live* source and only works with **realtime** playback mode, the *offline* version on the other hand is *offline* source and supports any playback mode.

General-Purpose Graph Node

The idea behind graph nodes design is that whenever there is a closed processing functionality, it can be wrapped in a single graph node implementation allowing to control it. The difference from the Mvx2 framework's design of filters (see Mvx2 framework documentation) is that graph nodes are more abstract. Internally, a graph node is allowed to maintain multiple subsequent Mvx2 filters, which is appropriate when these filters form together a functional block that is easier to maintain as a single unit rather than maintaining multiple independent units (graph nodes). In the end however, such graph node implementations are just sugar, because they just simplify control over a potentially complex processing feature (i.e. by introducing a domain-specific API). The drawback is that there would have to be specialized graph node implementations for whatever feature is needed at a given time.

Fortunately, there is a graph node implementation (Mvx2API::SingleFilterGraphNode), which is in close correlation with Mvx2 framework's design of filters, and thus allows to use just any Mvx2 filter in the Mvx2API environment without having to write specialized graph node wrapper for it. To use this graph node, a client needs to know the specification of an Mvx2 filter he wants to use - its unique Guid and a list of parameters and their valid values - the graph node makes it possible to set and get filter parameters' values in a generic way (via character strings).

Mvx2BasicIO

Mvx2BasicIO is the first extension module of Mvx2API. It is documented in a standalone document, but following is a quick overview of its purpose and features:

- provides graph nodes for accessing (reading and writing) Mvx2-formatted files,
- provides graph nodes for accessing (transmission and reception) Mvx2 network streams,
- provides utility for fast extraction of basic data information about Mvx2 files.

Class Diagram

Following is a class diagram showing all important classes in the context of the overall architecture of the API.

Chapter 3

Release Notes

1.0.0

Initial version.

1.1.0

Documentation

- **1.1.0_D1** | added 'release notes' section
- **1.1.0_D2** | added/updated missing API reference documentation

Samples

- **1.1.0_S1** | fixed output name ('_mvp' suffix)
- **1.1.0_S2** | improved sample filters

2.0.0

Framework

- **2.0.0_F1** | separated MVCommon as a standalone independent module (currently used MVCommon version: 1.2.0)
- **2.0.0_F2** | refactored MVX::PluginDatabase::AddPlugin to return boolean indication about result of adding a plugin and an option to provide failure reason
- **2.0.0_F3** | replaced MVX::VersionInfo class by MVCommon::VersionInfo (the definition is the same)
- **2.0.0_F4** | renamed Mvx2's VersionInfo.h/cpp file to MvxVersion.h/cpp
- **2.0.0_F5** | renamed MVX::mvxCompileVersion member to MVX::MVX_COMPILE_VERSION
- **2.0.0_F6** | refactored MVX::GetMvxRuntimeVersion() function into MVX::MVX_RUNTIME_VERSION constant

- **2.0.0_F7** | extended MVX_PLUGIN macro to imprint Mvx2 framework version, as well as its string-literal form, into the compiled plugin module
- **2.0.0_F8** | fixed initialization deadlock occurring on some platforms (Windows 7)
- **2.0.0_F9** | fixed crashes occuring when messages are logged via Mvx2's logger during an application initialization
- **2.0.0_F10** | added `purposeGuid` parameter to MVX::TextureFormatConverter's functions:
  - MVX::TextureFormatConverter::ConvertFromNVXtoRGB(),
  - MVX::TextureFormatConverter::ConvertFromDXT5YCOCGtoRGB(),
  - MVX::TextureFormatConverter::ConvertFromDXT1toRGB(),
  - MVX::TextureFormatConverter::ConvertFromNV12toRGB(),
  - MVX::TextureFormatConverter::ConvertFromNV21toRGB(),
  - MVX::TextureFormatConverter::ConvertFromBGRtoRGB(),
  - MVX::TextureFormatConverter::ConvertFromHSL24toRGB24(),
  - MVX::TextureFormatConverter::ConvertFromHSL30toRGB24(),
    The explicit purpose guid is applied to resulting textures
- **2.0.0_F11** | fixed TransformTextureNVX filter's output profile generation (purpose guid of input (to-be-converted) texture is preserved in the output profile)
- **2.0.0_F12** | fixed purpose guid of TransformTextureNVX and TransformTextureRGB filters' results - the output texture has the same purpose guid as the input one did
- **2.0.0_F13** | fixed FloatCompressor::DecompressFloatsFrom16Bit crash on Windows 7

MVGraphAPI

- **2.0.0_GA1** | added MVGraphAPI module (initial version) to the framework, including its .Net wrapper-module MVGraphAPINet
- **2.0.0_GA2** | added MVGraphAPI::AutoSequentialGraphRunner::GetPlaybackState() function

Build support

- **2.0.0_BS1** | replaced MVXConfig.cmake by Mvx2Config.cmake to reflect independence of MVCommon module (removed MVX::MVCommon target and MVX::Mvx2 target renamed to component-less target called Mvx2)
- **2.0.0_BS2** | fixed a bug in Mvx2Config.cmake related to fallback build configurations resolution
- **2.0.0_BS3** | refactored internal implementation of Mvx2Config.cmake
- **2.0.0_BS4** | introduced MVGraphAPIConfig.cmake, MVGraphAPINetConfig.cmake and MVGraphAPINet_↩ iOSConfig.cmake for the new framework additions (see **2.0.0_GA1**)

Tools

- **2.0.0_T1** | added MVPluginTester utility for testing loadability of plugin modules (check 'app/mvplugintester.py' script for composing the executable)

Documentation

- **2.0.0_D1** | switched documentation from xml-style comments to doxygen-style comments
- **2.0.0_D2** | introduced release notes identifiers
- **2.0.0_D3** | introduced documentation for MVGraphAPI and MVGraphAPINet for the new framework additions (see **2.0.0_GA1**)

3.0.0

Framework

- **3.0.0_F1** | updated `MVCommon` 3rdparty dependency to version 2.0.0
- **3.0.0_F2** | `MVX::MutateAtomMultiThread`-derived filters accept value 0 of **"Threads count"** parameter with extraordinary interpretation: the count of spawned threads is the same as the number of streams in frames
- **3.0.0_F3** | `MVX::MutateFrameMultiThread`-derived filters accept value 0 of **"Threads count"** parameter with extraordinary interpretation: the count of spawned threads is the same as the number of streams in frames
- **3.0.0_F4** | updated `libjpeg-turbo` 3rdparty dependency to version 2.0.2
- **3.0.0_F5** | fixed performance of texture compression algorithms on all platforms
- **3.0.0_F6** | extended `MVX::CirclesStatistics` structure with new fields
- **3.0.0_F7** | evolved `MVX::DataTypePatternDetector` data layer class to version 1, which utilizes the extended `MVX::CirclesStatistics` structure
- **3.0.0_F8** | fixed return type of `MVX::DataTypePatternDetector::GetDetectedColor↩FrameCountPerCam()` function from `float` to `uint32_t`
- **3.0.0_F9** | added public static functions `ValueToString()` to
  - `MVX::FilterParamBool`,
  - `MVX::FilterParamFloat`,
  - `MVX::FilterParamInt64`,
  - `MVX::FilterParamInt32`,
  - `MVX::FilterParamUInt32`,
  - `MVX::FilterParamColorRgba`,
  - `MVX::FilterParamVector2`,
  - `MVX::FilterParamVector3`,
  - `MVX::FilterParamVector4` and
  - `MVX::FilterParamMatrix4x4f`
    so clients can manually convert typed values to their string representations the same way as the respective filter param classes do internally
- **3.0.0_F10** | added public static functions `StringToValue()` to
  - `MVX::FilterParamUInt32`,
  - `MVX::FilterParamColorRgba`,
  - `MVX::FilterParamVector2`,
  - `MVX::FilterParamVector3`,
  - `MVX::FilterParamVector4` and

  - `MVX::FilterParamMatrix4x4f`
    so clients can manually convert string representations to typed values the same way as the respective filter param classes do internally

- **3.0.0_F11** | replaced `int` value type of `MVX::FilterParamInt32` with `int32_t` which has strict size independent from platform
- **3.0.0_F12** | refactored both API and internal structure of
  - `MVX::FilterParamVector2`,
  - `MVX::FilterParamVector3` and
  - `MVX::FilterParamVector4`
    implementations, so internally they use `MVCommon::Vector2f`, `MVCommon::Vector3f` and `MVCommon::Vector4f` objects respectively instead of the raw data arrays
- **3.0.0_F13** | `MVX::VisualGraph` is now derived from `MVX::ErrorHolder` so it can store and provide last (human-readable) error when its `InstantiateMvxGraph()` or `InstantiateSimpleMvx↩Graph()` functions fail to instantiate an MVX graph

Build support

- **3.0.0_BS1** | size of Android and LuminOS libraries reduced by ~90%
- **3.0.0_BS2** | android API level raised from 19 to 21
- **3.0.0_BS3** | Linux and MacOS binaries do not consist of a versioned library file and a version-neutral symlink file anymore - the library file itself has version-neutral name

4.0.0

MVGraphAPI

- **4.0.0_GA1** | integrated MVGraphAPI module directly into Mvx2 framework:
  1. `MVGraphAPI` product renamed to `Mvx2API`
  2. public header files of MVGraphAPI moved to `include/Mvx2API` directory, which is a sibling of Mvx2 framework's original `include/Mvx2` directory
  3. `MVGraphAPI` namespace renamed to `Mvx2API`
  4. updated and merged MVGraphAPI's documentation into Mvx2's documentation as a subpage
  5. removed `Mvx2/Mvx2API.h` file containing `MVX2_API` macro definition
  6. renamed `MV_GRAPH_API` macro to `MVX2_API` in `Mvx2API/Mvx2API.h` file
  7. removed `MVGraphAPIConfig.cmake` cmake-build file
  8. removed MVGraphAPI as a standalone module completely (library files, header files, documentation files, cmake config files)
- **4.0.0_GA2** | renamed MVGraphAPINet module to Mvx2Net:
  1. `MVGraphAPI` product renamed to `Mvx2API`
  2. `MVGraphAPI` namespace renamed to `Mvx2API`
  3. `MVGraphAPINet.zip` file containing MVGraphAPINet/Mvx2Net documentation renamed to `Mvx2↩Net.zip`
  4. `MVGraphAPINetConfig.cmake` and `MVGraphAPINet_iOSConfig.cmake` cmake-build files updated and renamed to `Mvx2NetConfig.cmake` and `Mvx2Net_iOSConfig.cmake` respectively
  5. `MVGraphAPI::MVGraphAPINetConstants` class renamed to `Mvx2API::Constants` and its `MV_GRAPH_API_INTEROP_DLL` field to `INTEROP_DLL`

Mvx2API

- **4.0.0_MA1** | renamed `Mvx2API::IFrameListener` class to `Mvx2API::FrameListener`
- **4.0.0_MA2** | introduced Mvx2API::AutoCompressorGraphNode and Mvx2API::AutoDecompressorGraphNode for compression and decompression of Mvx2 data
- **4.0.0_MA3** | introduced Mvx2API::InjectFileDataGraphNode and Mvx2API::InjectMemoryDataGraphNode for injection of file- or memory-stored binary data to a pipeline
- **4.0.0_MA4** | introduced Mvx2API::MeshData structure holding mesh data
- **4.0.0_MA5** | introduced Mvx2API::MeshSplitter utility for splitting meshes into smaller ones
- **4.0.0_MA6** | introduced Mvx2API::BasicDataLayersGuids providing a collection of basic data Guids
- **4.0.0_MA7** | introduced frame data extractors for data extraction from frames:
  - Mvx2API::FrameAudioExtractor
  - Mvx2API::FrameMeshExtractor
  - Mvx2API::FrameMiscDataExtractor
  - Mvx2API::FrameTextureExtractor
- **4.0.0_MA8** | introduced keyboard and mouse event data structures:
  - Mvx2API::KeyDownEvent
  - Mvx2API::KeyUpEvent
  - Mvx2API::MouseDownEvent
  - Mvx2API::MouseUpEvent
  - Mvx2API::MouseDoubleClickEvent
  - Mvx2API::MouseMoveEvent
  - Mvx2API::MouseWheelEvent
- **4.0.0_MA9** | introduced experimental Mvx2API::Experimental::RendererGraphNode for rendering visual Mvx2 data

Framework

- **4.0.0_F1** | FILTER_DECL macro does not export any symbols anymore - to declare a filter with exported symbols, a new FILTER_DECL_EXPORT macro shall be used with custom export macro
- **4.0.0_F2** | DATALAYER_DECL macro does not export any symbols anymore - to declare a data layer with exported symbols, a new DATALAYER_DECL_EXPORT macro shall be used with custom export macro
- **4.0.0_F3** | removed invalid MVX2_API macro decoration from template functions:
  - MVX::DataLayerFactory::CreateDataLayer (2 overloads)
  - MVX::FilterFactory::CreateFilter
  - MVX::FilterCategoryDeterminer::DetermineFilterCategory

Documentation

- **4.0.0_D1** | updated 'Mvx2API' section, including a class diagram on the page

4.1.0

Mvx2API

- **4.1.0_MA1** | added a support for accessing NV12 and NV21 textures:
  - added Mvx2API::BasicDataLayersGuids::NV12_TEXTURE_DATA_LAYER and Mvx2API::BasicDataLayersGuids::NV21_TEXTURE_DATA_LAYER
  - added Mvx2API::FrameTextureExtractor::TextureType::TT_NV12 and Mvx2API::FrameTextureExtractor::TextureType::TT_NV21
- **4.1.0_MA2** | added a support for reinitialization of existing graphs (see Mvx2API::Graph::Reinitialize)
- **4.1.0_MA3** | fixed invalid values returned from Mvx2API::Frame::StreamContainsDataLayer and Mvx2API::SourceInfo::ContainsDataLayer caused by bugged compiler optimization on Windows
- **4.1.0_MA4** | introduced filter parameter names-enumerating feature in Mvx2API::SingleFilterGraphNode represented by Mvx2API::SingleFilterGraphNode::ParameterNamesBegin and Mvx2API::SingleFilterGraphNode::ParameterNames functions
- **4.1.0_MA5** | Mvx2API::GraphBuilder::CompileGraphAndReset now performs a complete graph reinitialization before the graph is returned so filter parameter changes which would potentially modify the graph behaviour can take effect

Framework

- **4.1.0_F1** | added an `alsoDepreinitialize` parameter to MVX::Filter::Reset, which allows the function to deinitialize a filter's parameters as well if requested. Default value is `false` to secure compatibility with existing calls of the function
- **4.1.0_F2** | introduced MVX::StatusPropertyUInt64 class for 64-bit unsigned int status properties
- **4.1.0_F3** | introduced MVX::DataTypePointer64 data layer class for storing raw C pointers on 64bit platforms
- **4.1.0_F4** | introduced MVX::Filter::GetParameters which returns a reference to a filter's collection of registered parameters
- **4.1.0_F5** | introduced MVX::FilterParamStringChoices::GetChoices for getting available choices
- **4.1.0_F6** | introduced MVX::DataTypeH264CompressedTexture as a temporary solution to the **4.1.0_KB1** bug for compressed H264 data - the data layer type is now implemented directly in the framework so it is always known by it

Known bugs

- **4.1.0_KB1** | the framework crashes when it needs to deserialize a data layer of a type derived from MVX::DataTypeCompressedBlob, which is not known by the framework at the time (i.e. a data layer type is implemented in a plugin module, but the plugin module is not available to the framework). The bug is only related to derivatives of the MVX::DataTypeCompressedBlob - the same scenario works without issues with other data layer types

4.2.0

Framework

- **4.2.0_F1** | fixed linker errors occuring when MVX::MutateTextureColor, MVX::TransformTextureConversion and MVX::MutateCompressor classes are used or derived from
- **4.2.0_F2** | introduced new named purpose guids:
  - MVX::PurposeGuid_MULTIPATCH_COLOR
  - MVX::PurposeGuid_MULTIPATCH_DEPTH
  - MVX::PurposeGuid_MULTIPATCH_COMBINED
- **4.2.0_F3** | introduced new functions to MVX::TextureFormatConverter for converting textures to NV12 format:
  - MVX::TextureFormatConverter::ConvertFromRGBtoNV12()
  - MVX::TextureFormatConverter::ConvertFromNVXtoNV12()
- **4.2.0_F4** | introduced filters for converting textures to NV12 format:
  - MVX::TransformTextureRGBtoNV12 for RGB to NV12 conversions
  - MVX::TransformTextureNVXtoNV12 for NVX to NV12 conversions
- **4.2.0_F5** | fixed NV12 to NVX texture format conversion algorithm implemented in MVX::TextureFormat↩ Converter::ConvertFromNV12toNVX function
- **4.2.0_F6** | fixed MVX::TransformTextureNV12toNVX filter's conversion of NV12 textures to NVX textures (see **4.2.0_F5**)

5.0.0

Framework

- **5.0.0_F1** | updated `MVCommon` 3rdparty dependency to version 3.0.0
- **5.0.0_F2** | fixed MVX::MutateAtomMultiThread base class for multi-threaded mutate filters, so the derived filters can properly finish their thread-distributed work also in non-live playback modes (i.e. those which have an implicit end of stream)
  - previously once the stream on the input ended, the filter was unable to push its just-being-processed atoms to the output and thus became stuck
- **5.0.0_F3** | updated signature of MVX::MutateAtomMultiThread::ProcessAtom() function so errors raised during an atom-processing routine could be reported by the filter as a processing error to the graph
- **5.0.0_F4** | fixed MVX::MutateFrameMultiThread base class for multi-threaded mutate filters, so the derived filters can properly finish their thread-distributed work also in non-live playback modes (i.e. those which have an implicit end of stream)
  - previously once the stream on the input ended, the filter was unable to push its just-being-processed frames to the output and thus became stuck
- **5.0.0_F5** | updated signature of MVX::MutateFrameMultiThread::ProcessFrame() function so errors raised during a frame-processing routine could be reported by the filter as a processing error to the graph
- **5.0.0_F6** | fixed exposure of MVX::FilterParam::InvokeParameterValueChanged() function from the framework to eliminate linker errors (on Windows platform) that rendered implementation of filter parameter derivatives outside of the framework impossible

Mvx2API

- **5.0.0_MA1** | in Mvx2Net module renamed Mvx2API::MeshData::CopyBoundingBox(IntPtr targetBounding↩ Box) function to Mvx2API::MeshData::CopyBoundingBoxRaw(IntPtr targetBoundingBox)
- **5.0.0_MA2** | introduced Mvx2API::FrameAudioExtractor::CopyPCMDataRaw() functions to Mvx2Net module as alternatives to Mvx2API::FrameAudioExtractor::CopyPCMData() which expect a System.IntPtr pointer to a target memory as a parameter instead of a typed array
- **5.0.0_MA3** | introduced Mvx2API::FrameTextureExtractor::CopyTextureDataRaw() functions to Mvx2Net module as alternatives to Mvx2API::FrameTextureExtractor::CopyTextureData() which expect a System.IntPtr pointer to a target memory as a parameter instead of a typed array
- **5.0.0_MA4** | fixed a bug of Mvx2API::FrameListener in Mvx2Net which prevented its independent instances from processing frames at the same time
- **5.0.0_MA5** | fixed a bug of Mvx2API::ParameterValueChangedListener in Mvx2Net which prevented its independent instances from notifying about changed parameters at the same time
- **5.0.0_MA6** | added a support for enumerating data profiles of frames:
  - introduced Mvx2API::DataProfile class
  - introduced Mvx2API::DataProfileIterator to Mvx2 module and Mvx2API::DataProfileEnumerator to Mvx2Net module
  - introduced Mvx2API::SourceInfo::DataProfilesBegin() and Mvx2API::SourceInfo::DataProfilesEnd() functions to Mvx2 module
  - introduced Mvx2API::Frame::DataProfilesBegin() and Mvx2API::Frame::DataProfilesEnd() functions to Mvx2 module
  - introduced Mvx2API::SourceInfo::CreateDataProfilesEnumerator() function to Mvx2Net module
  - introduced Mvx2API::Frame::CreateDataProfilesEnumerator() function to Mvx2Net module

Build support

- **5.0.0_BS1** | CMake minimal required version increased from 3.9 to 3.14
  - updated `Mvx2Config.cmake`, `Mvx2NetConfig.cmake` and `Mvx2Net_iOSConfig.cmake` scripts and their dependencies

Tools

- **5.0.0_T1** | updated 'app/mvplugintester.py' script for composing the MVPluginTester tool executable to expect a path to the root directory of the MVCommon dependency as a first and mandatory parameter for `grab_↩ app` task

Samples

- **5.0.0_S1** | CMake minimal required version increased from 3.9 to 3.14
  - updated `CMakeLists.txt` of `mvx2plugin` sample
- **5.0.0_S2** | updated `mvx2plugin` sample's `CMakeLists.txt` to expect MVCommon dependency on a potentially different path than Mvx2 dependency
  - introduced `build/local_config/mvcommon_root_dir.cfg` config file inside the sample root directory, which shall specify a path to the MVCommon root directory

6.0.0

Framework

- **6.0.0_F1** | updated `MVCommon` 3rdparty dependency to version 4.0.0
- **6.0.0_F2** | upgraded multiple internal dependencies with possible effect on:
  - MVX::PluginDatabase
  - Mvx2API::PluginsLoader

Build support

- **6.0.0_BS1** | from now on the windows libraries are compiled using msvc compiler version 142 (VS 2019)
- **6.0.0_BS2** | upgraded `cmake/toolchains/ios.cmake` toolchain file used for building for iOS platform

Documentation

- **6.0.0_D1** | introduced PDF documentation as an alternative to the HTML one:
  - `doc/Mvx2.pdf`
  - `doc/Mvx2Net.pdf`

Samples

- **6.0.0_S1** | from now on the windows libraries of the samples are compiled using msvc compiler version 142 (VS 2019)

6.1.0

Mvx2API

- **6.1.0_MA1** | added a support for refreshing a graph being built via Mvx2API::GraphBuilder:
  - introduced Mvx2API::GraphBuilder::Refresh() function
- **6.1.0_MA2** | added a support for enumerating data profiles of a graph being built by Mvx2API::GraphBuilder in its current state:
  - introduced Mvx2API::GraphBuilder::DataProfilesBegin() and Mvx2API::GraphBuilder::DataProfilesEnd() functions to Mvx2 module
  - introduced Mvx2API::GraphBuilder::CreateDataProfilesEnumerator() function to Mvx2Net module
  - introduced Mvx2API::GraphBuilder::ContainsDataProfile() function
- **6.1.0_MA3** | fixed a memory leak caused by destructor of a Mvx2API::ManualGraphBuilder
- **6.1.0_MA4** | added a support for enumerating data profiles of single-filter graph nodes:
  - introduced Mvx2API::SingleFilterGraphNode::DataProfilesBegin() and Mvx2API::SingleFilterGraphNode::DataProfilesEnd() functions to Mvx2 module
  - introduced Mvx2API::SingleFilterGraphNode::CreateDataProfilesEnumerator() function to Mvx2Net module
  - introduced Mvx2API::SingleFilterGraphNode::ContainsDataProfile() function

6.2.0

Framework

- **6.2.0_F1** | extended `MVX::SourceEmptySource` source filter with support for non-realtime playback modes
  - the source filter is no longer limited to 'live source' behaviour (i.e. it now supports also other than auto-sequential graph runners set to 'realtime' playback mode)
  - introduced `Frames count` parameter with a default value equal to a max value of uint32_t type

Samples

- **6.2.0_S1** | introduced `mvx2apidemo` and `mvx2apinetdemo` samples for showcasing usage of Mvx2API
  - both samples are compiled using Cmake and include python scripts for their simple compilation and execution

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 7

Data Structure Documentation

7.1 Mvx2API::AsyncFrameAccessGraphNode Class Reference

A graph node for asynchronous notifications about processed MVX frames.

`#include <AsyncFrameAccessGraphNode.h>`

Inherits Mvx2API::SingleFilterGraphNode.

Public Member Functions

- MVX2_API AsyncFrameAccessGraphNode (FrameListener *pFrameListener=nullptr)
  A constructor.
- virtual MVX2_API ~AsyncFrameAccessGraphNode ()
  A destructor.
- MVX2_API void SetFrameListener (FrameListener *pFrameListener)
  Sets an asynchronous frame listener to be used.

Additional Inherited Members

7.1.1 Detailed Description

A graph node for asynchronous notifications about processed MVX frames.

Internally maintains a single filter for asynchronous access to frames. The same filter is reused even when the graph node is added to multiple graphs.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AsyncFrameAccessGraphNode()

```
MVX2_API Mvx2API::AsyncFrameAccessGraphNode::AsyncFrameAccessGraphNode (
            FrameListener * pFrameListener = nullptr )
```

A constructor.

Parameters

pFrameListener	an asynchronous frames listener

7.1.3 Member Function Documentation

7.1.3.1 SetFrameListener()

```
MVX2_API void Mvx2API::AsyncFrameAccessGraphNode::SetFrameListener (
            FrameListener * pFrameListener )
```

Sets an asynchronous frame listener to be used.

Parameters

pFrameListener	an asynchronous frames listener

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/AsyncFrameAccessGraphNode.h

7.2 Mvx2API::AtomList Class Reference

A collection of streams.

```
#include <AtomList.h>
```

Public Member Functions

- MVX2_API AtomList ()
  A constructor.
- MVX2_API AtomList (AtomList const &other)
  A copy-constructor.
- MVX2_API ~AtomList ()
  A destructor.
- MVX2_API void PushBack (SharedAtomPtr const &atom)
  Pushes a stream to the collection.
- MVX2_API SharedAtomPtr & operator[ ] (uint32_t pos)
  Returns a stream at a given index in the collection.
- MVX2_API const SharedAtomPtr & operator[ ] (uint32_t pos) const
  Returns a stream at a given index in the collection.
- MVX2_API uint32_t Count () const
  Returns a count of streams in the collection.
- MVX2_API void Clear ()
  Empties the collection.

7.2.1 Detailed Description

A collection of streams.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AtomList()

```
MVX2_API Mvx2API::AtomList::AtomList (
            AtomList const & other )
```

A copy-constructor.

Parameters

other	another collection to copy streams from

7.2.3 Member Function Documentation

7.2.3.1 Count()

```
MVX2_API uint32_t Mvx2API::AtomList::Count ( ) const
```

Returns a count of streams in the collection.

Returns

streams count

7.2.3.2 operator[]() [1/2]

```
MVX2_API SharedAtomPtr& Mvx2API::AtomList::operator[] (
            uint32_t pos )
```

Returns a stream at a given index in the collection.

Parameters

pos	an index of stream to return

Returns

a stream at the index

7.2.3.3 operator[]() [2/2]

```
MVX2_API const SharedAtomPtr& Mvx2API::AtomList::operator[] (
            uint32_t pos ) const
```

Returns a stream at a given index in the collection.

Parameters

pos	an index of stream to return

Returns

a stream at the index

7.2.3.4 PushBack()

```
MVX2_API void Mvx2API::AtomList::PushBack (
            SharedAtomPtr const & atom )
```

Pushes a stream to the collection.

Parameters

atom	a stream to push

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/AtomList.h

7.3 Mvx2API::AutoCompressorGraphNode Class Reference

A graph node for auto-compression of MVX data.

```
#include <AutoCompressorGraphNode.h>
```

Inherits Mvx2API::GraphNode.

Public Member Functions

- MVX2_API AutoCompressorGraphNode (bool dropUncompressedInput=true)
  A constructor.
- virtual MVX2_API ~AutoCompressorGraphNode ()
  A destructor.

7.3.1 Detailed Description

A graph node for auto-compression of MVX data.

Internally creates a new compression filter every time the graph node is added to a new graph.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 AutoCompressorGraphNode()

```
MVX2_API Mvx2API::AutoCompressorGraphNode::AutoCompressorGraphNode (
            bool dropUncompressedInput = true )
```

A constructor.

Parameters

dropUncompressedInput	an indication whether the original uncompressed data shall be dropped

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/AutoCompressorGraphNode.h

7.4 Mvx2API::AutoDecompressorGraphNode Class Reference

A graph node for auto-decompression of MVX data.

`#include <AutoDecompressorGraphNode.h>`

Inherits Mvx2API::GraphNode.

Public Member Functions

- MVX2_API AutoDecompressorGraphNode (bool dropCompressedInput=true)
  A constructor.
- virtual MVX2_API ~AutoDecompressorGraphNode ()
  A destructor.

7.4.1 Detailed Description

A graph node for auto-decompression of MVX data.

Internally creates a new decompression filter every time the graph node is added to a new graph.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 AutoDecompressorGraphNode()

```
MVX2_API Mvx2API::AutoDecompressorGraphNode::AutoDecompressorGraphNode (
            bool dropCompressedInput = true )
```

A constructor.

Parameters

dropCompressedInput	an indication whether the original compressed data shall be dropped

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/AutoDecompressorGraphNode.h

7.5 Mvx2API::AutoSequentialGraphRunner Class Reference

A sequential runner of data-processing graphs with automatic (synchronous/asynchronous) updates-invocation.

`#include <AutoSequentialGraphRunner.h>`

Inherits Mvx2API::GraphRunner.

Public Member Functions

- MVX2_API AutoSequentialGraphRunner (Graph *graph)

  A constructor.
- virtual MVX2_API ~AutoSequentialGraphRunner ()

  A destructor.
- MVX2_API bool Play (RunnerPlaybackMode playbackMode, bool blockUntilStopped=false)

  Starts playback of the graph with a given playback mode.
- MVX2_API bool Stop ()

  Invokes stopping of the graph playback.
- MVX2_API bool Pause ()

  Pauses the graph playback.
- MVX2_API bool Resume ()

Resumes the graph playback.

- MVX2_API RunnerPlaybackState GetPlaybackState () const

  Determines current playback state of the runner.

- MVX2_API void SeekFrame (uint32_t frameID)

  Sets a frame with a given ID as the next to be processed.

- virtual MVX2_API SourceInfo * GetSourceInfo () const override

  Retrieves source information about the currently open MVX source.

7.5.1 Detailed Description

A sequential runner of data-processing graphs with automatic (synchronous/asynchronous) updates-invocation.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 AutoSequentialGraphRunner()

```
MVX2_API Mvx2API::AutoSequentialGraphRunner::AutoSequentialGraphRunner (
            Graph * graph )
```

A constructor.

Parameters

graph	a graph to create the runner for

7.5.3 Member Function Documentation

7.5.3.1 GetPlaybackState()

```
MVX2_API RunnerPlaybackState Mvx2API::AutoSequentialGraphRunner::GetPlaybackState ( ) const
```

Determines current playback state of the runner.

Returns

playback state

7.5.3.2 GetSourceInfo()

```
virtual MVX2_API SourceInfo* Mvx2API::AutoSequentialGraphRunner::GetSourceInfo ( ) const  [override],
[virtual]
```

Retrieves source information about the currently open MVX source.

Returns

information about the current MVX source or null if no source is open

Implements Mvx2API::GraphRunner.

7.5.3.3 Pause()

```
MVX2_API bool Mvx2API::AutoSequentialGraphRunner::Pause ( )
```

Pauses the graph playback.

Returns

true if the graph playback successfully paused

7.5.3.4 Play()

```
MVX2_API bool Mvx2API::AutoSequentialGraphRunner::Play (
            RunnerPlaybackMode playbackMode,
            bool blockUntilStopped = false )
```

Starts playback of the graph with a given playback mode.

Can be executed synchronously in case `blockUntilStopped` is set to true, or asynchronously when set to false.

Parameters

playbackMode	a playback mode
blockUntilStopped	an indication whether to block the call until the execution of the graph stops

Returns

true if the graph playback successfully started

7.5.3.5 Resume()

```
MVX2_API bool Mvx2API::AutoSequentialGraphRunner::Resume ( )
```

Resumes the graph playback.

Returns

true if the graph playback successfully resumed

7.5.3.6 SeekFrame()

```
MVX2_API void Mvx2API::AutoSequentialGraphRunner::SeekFrame (
            uint32_t frameID )
```

Sets a frame with a given ID as the next to be processed.

Parameters

frameID	an ID of the frame to be processed next

7.5.3.7 Stop()

```
MVX2_API bool Mvx2API::AutoSequentialGraphRunner::Stop ( )
```

Invokes stopping of the graph playback.

The function only invokes stopping of the graph playback, which means that the graph may not be stopped yet when the function returns (although in case of non-blocking playback, the playback will definitely be stopped when the function returns).

Returns

true if the graph playback stopping successfully invoked

The documentation for this class was generated from the following file:

- public/Mvx2API/runners/AutoSequentialGraphRunner.h

7.6 Mvx2API::BlockFPSGraphNode Class Reference

A blocking graph node with an automatized framerate-based frames-pulling capability.

```
#include <BlockFPSGraphNode.h>
```

Inherits Mvx2API::BlockGraphNode.

Public Member Functions

- MVX2_API BlockFPSGraphNode (uint32_t bufferSize=3, float fps=-1.0f, FullBehaviour fullBehaviour=FB_DROP_FRAMES)

  A constructor.
- virtual MVX2_API ~BlockFPSGraphNode ()

  A destructor.
- MVX2_API void SetFPS (float fps)

  Sets a new framerate to follow with frames-pulling.

Static Public Attributes

- static const MVX2_API float FPS_MAX

  A special framerate value indicating that the maximal possible framerate shall be used.
- static const MVX2_API float FPS_FROM_SOURCE

  A special framerate value indicating that the framerate of an open source shall be used.
- static const MVX2_API float FPS_HALF_FROM_SOURCE

  A special framerate value indicating that the half of the framerate of an open source shall be used.
- static const MVX2_API float FPS_DOUBLE_FROM_SOURCE

  A special framerate value indicating that the double of the framerate of an open source shall be used.

Additional Inherited Members

7.6.1 Detailed Description

A blocking graph node with an automatized framerate-based frames-pulling capability.

Internally maintains a single blocking filter. The same filter is reused even when the graph node is added to multiple graphs.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 BlockFPSGraphNode()

```
MVX2_API Mvx2API::BlockFPSGraphNode::BlockFPSGraphNode (
            uint32_t bufferSize = 3,
            float fps = -1.0f,
            FullBehaviour fullBehaviour = FB_DROP_FRAMES )
```

A constructor.

Parameters

bufferSize	a size of internal frames buffer
fps	a framerate to follow with frames-pulling
fullBehaviour	an initial full-behaviour

Exceptions

std::runtime_error	raised in case the creation of the internal filter fails

7.6.3 Member Function Documentation

7.6.3.1 SetFPS()

```
MVX2_API void Mvx2API::BlockFPSGraphNode::SetFPS (
            float fps )
```

Sets a new framerate to follow with frames-pulling.

Parameters

fps	a framerate to follow

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/BlockFPSGraphNode.h

7.7 Mvx2API::BlockGraphNode Class Reference

A graph node with a buffering and execution-blocking capabilities.

`#include <BlockGraphNode.h>`

Inherits Mvx2API::GraphNode.

Inherited by Mvx2API::BlockFPSGraphNode, and Mvx2API::BlockManualGraphNode.

Public Types

- enum FullBehaviour { FB_DROP_FRAMES, FB_BLOCK_FRAMES }

  Enumeration of supported behaviours when the buffer of the node is full.

Public Member Functions

- MVX2_API void SetFullBehaviour (FullBehaviour fullBehaviour)

  Sets a full-behaviour - action to perform when the internal buffer of frames becomes full.
- MVX2_API uint64_t GetDroppedFramesCount () const

  Gets a value of internal counter of dropped frames.
- MVX2_API void ResetDroppedFramesCounter () const

  Resets the internal counter of dropped frames to zero.

7.7.1 Detailed Description

A graph node with a buffering and execution-blocking capabilities.

Internally maintains a single blocking filter. The same filter is reused even when the graph node is added to multiple graphs.

7.7.2 Member Enumeration Documentation

7.7.2.1 FullBehaviour

```
enum Mvx2API::BlockGraphNode::FullBehaviour
```

Enumeration of supported behaviours when the buffer of the node is full.

Enumerator

FB_DROP_FRAMES	Additional frames are dropped.
FB_BLOCK_FRAMES	Execution of additional frames is blocked.

7.7.3 Member Function Documentation

7.7.3.1 GetDroppedFramesCount()

```
MVX2_API uint64_t Mvx2API::BlockGraphNode::GetDroppedFramesCount ( ) const
```

Gets a value of internal counter of dropped frames.

Returns

dropped frames count

7.7.3.2 SetFullBehaviour()

```
MVX2_API void Mvx2API::BlockGraphNode::SetFullBehaviour (
            FullBehaviour fullBehaviour )
```

Sets a full-behaviour - action to perform when the internal buffer of frames becomes full.

Parameters

fullBehaviour	a behaviour to set

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/BlockGraphNode.h

7.8 Mvx2API::BlockManualGraphNode Class Reference

A blocking graph node with a manual frames-pulling capability.

`#include <BlockManualGraphNode.h>`

Inherits Mvx2API::BlockGraphNode.

Public Member Functions

- MVX2_API BlockManualGraphNode (uint32_t bufferSize=3, FullBehaviour fullBehaviour=FB_DROP_FRAMES)
  A constructor.
- virtual MVX2_API ~BlockManualGraphNode ()
  A destructor.
- MVX2_API void PullNextProcessedFrame ()
  Releases the oldest of the buffered frames for further processing.

Additional Inherited Members

7.8.1 Detailed Description

A blocking graph node with a manual frames-pulling capability.

Internally maintains a single blocking filter. The same filter is reused even when the graph node is added to multiple graphs.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 BlockManualGraphNode()

```
MVX2_API Mvx2API::BlockManualGraphNode::BlockManualGraphNode (
            uint32_t bufferSize = 3,
            FullBehaviour fullBehaviour = FB_DROP_FRAMES )
```

A constructor.

Parameters

bufferSize	a size of internal frames buffer
fullBehaviour	an initial full-behaviour

Exceptions

std::runtime_error	raised in case the creation of the internal filter fails

7.8.3 Member Function Documentation

7.8.3.1 PullNextProcessedFrame()

```
MVX2_API void Mvx2API::BlockManualGraphNode::PullNextProcessedFrame ( )
```

Releases the oldest of the buffered frames for further processing.

Effectively makes a space for another processed frame.

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/BlockManualGraphNode.h

7.9 Mvx2API::ColRGBAData Struct Reference

A structure containing color data.

```
#include <MeshDataTypes.h>
```

Data Fields

- uint8_t r

  A red color component.
- uint8_t g

  A green color component.
- uint8_t b

  A blue color component.
- uint8_t a

  An alpha color component.

7.9.1 Detailed Description

A structure containing color data.

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/mesh/MeshDataTypes.h

7.10 MVX::DataLayerClassInfo Class Reference

An information data about a data layer class.

`#include <DataLayerClassInfo.h>`

Public Member Functions

- MVX2_API DataLayerClassInfo ()
  A default constructor.
- MVX2_API DataLayerClassInfo (MVCommon::String const &className)
  A constructor.
- MVX2_API ~DataLayerClassInfo ()
  A destructor.
- MVX2_API MVCommon::String GetClassName () const
  Returns data layer's class name.
- MVX2_API MVCommon::String GetNiceClassName () const
  Returns data layer's nice class name.

Static Public Member Functions

- static MVX2_API MVCommon::String NicifyDataLayerClassName (MVCommon::String const &dataLayer↩ ClassName)
  Transforms a technical data layer class name into a nice human-readable name.

7.10.1 Detailed Description

An information data about a data layer class.

Provides basic information about a data layer class.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 DataLayerClassInfo()

```
MVX2_API MVX::DataLayerClassInfo::DataLayerClassInfo (
            MVCommon::String const & className )
```

A constructor.

Parameters

className	a name of the data layer class

7.10.3 Member Function Documentation

7.10.3.1 GetClassName()

```
MVX2_API MVCommon::String MVX::DataLayerClassInfo::GetClassName ( ) const
```

Returns data layer's class name.

Returns

data layer's class name

7.10.3.2 GetNiceClassName()

```
MVX2_API MVCommon::String MVX::DataLayerClassInfo::GetNiceClassName ( ) const
```

Returns data layer's nice class name.

Returns

data layer's nice class name

7.10.3.3 NicifyDataLayerClassName()

```
static MVX2_API MVCommon::String MVX::DataLayerClassInfo::NicifyDataLayerClassName (
            MVCommon::String const & dataLayerClassName )  [static]
```

Transforms a technical data layer class name into a nice human-readable name.

Parameters

dataLayerClassName	a class name to transform

Returns

a nice class name

The documentation for this class was generated from the following file:

- public/Mvx2/core/datalayers/DataLayerClassInfo.h

7.11 MVX::DataLayerFactoryIterator Class Reference

An iterator over elements of DataLayerFactory collection.

`#include <DataLayerFactoryIterator.h>`

Public Types

- using ValueType = MVCommon::Pair< MVCommon::Guid, DataLayerClassInfo const >
  A type of iterated-over elements of DataLayerFactory collection.

Public Member Functions

- MVX2_API DataLayerFactoryIterator (DataLayerFactoryIterator const &other)
  A copy constructor.
- MVX2_API DataLayerFactoryIterator (DataLayerFactoryIterator &&other)
  A move constructor.
- virtual MVX2_API ~DataLayerFactoryIterator ()
  A destructor.
- MVX2_API DataLayerFactoryIterator & operator++ ()
  A prefix incrementation operator.
- MVX2_API DataLayerFactoryIterator operator++ (int)
  A postfix incrementation operator.
- MVX2_API ValueType operator* () const
  Dereferences the iterator.

7.11.1 Detailed Description

An iterator over elements of DataLayerFactory collection.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 DataLayerFactoryIterator() [1/2]

```
MVX2_API MVX::DataLayerFactoryIterator::DataLayerFactoryIterator (
            DataLayerFactoryIterator const & other )
```

A copy constructor.

Parameters

other	an iterator to make a copy of

7.11.2.2 DataLayerFactoryIterator() [2/2]

```
MVX2_API MVX::DataLayerFactoryIterator::DataLayerFactoryIterator (
            DataLayerFactoryIterator && other )
```

A move constructor.

Parameters

other	an iterator to move

7.11.3 Member Function Documentation

7.11.3.1 operator∗()

```
MVX2_API ValueType MVX::DataLayerFactoryIterator::operator* ( ) const
```

Dereferences the iterator.

Returns

a pair of Guid and its data layer class info

7.11.3.2 operator++() [1/2]

```
MVX2_API DataLayerFactoryIterator& MVX::DataLayerFactoryIterator::operator++ ( )
```

A prefix incrementation operator.

Moves the iterator to the next element and returns this updated iterator.

Returns

this iterator after it was updated

7.11.3.3 operator++() [2/2]

```
MVX2_API DataLayerFactoryIterator MVX::DataLayerFactoryIterator::operator++ (
            int  )
```

A postfix incrementation operator.

Moves the iterator to the next element, but returns the original iterator.

Returns

the original unupdated iterator

The documentation for this class was generated from the following file:

- public/Mvx2/core/datalayers/DataLayerFactoryIterator.h

7.12 Mvx2API::DataProfile Class Reference

A profile of a single data item.

```
#include <DataProfile.h>
```

Public Member Functions

- MVX2_API DataProfile (MVCommon::Guid const &typeGuid, MVCommon::Guid const &compressedType↩ Guid, MVCommon::Guid const &purposeGuid)

  A constructor.
- MVX2_API DataProfile (DataProfile const &other)

  A copy constructor.
- MVX2_API DataProfile (DataProfile &&other)

  A move constructor.
- virtual MVX2_API ∼DataProfile ()

  A destructor.
- MVX2_API const MVCommon::Guid & GetTypeGuid () const

  A getter of the data type guid.
- MVX2_API const MVCommon::Guid & GetCompressedTypeGuid () const

  A getter of the compressed data type guid.
- MVX2_API const MVCommon::Guid & GetPurposeGuid () const

  A getter of the purpose guid.

7.12.1 Detailed Description

A profile of a single data item.

A data profile is represented as an MVCommon::Guid triplet:

- a data type guid (mandatory),
- a compressed data type guid (optional - in case the data is a 'wrapper' over actual compressed data),
- a purpose guid (mandatory).

7.12.2 Constructor & Destructor Documentation

7.12.2.1 DataProfile() [1/3]

```
MVX2_API Mvx2API::DataProfile::DataProfile (
            MVCommon::Guid const & typeGuid,
            MVCommon::Guid const & compressedTypeGuid,
            MVCommon::Guid const & purposeGuid )
```

A constructor.

Parameters

typeGuid	a data type guid
compressedTypeGuid	a compressed data type guid
purposeGuid	a purpose guid

7.12.2.2 DataProfile() [2/3]

```
MVX2_API Mvx2API::DataProfile::DataProfile (
            DataProfile const & other )
```

A copy constructor.

Parameters

other	an instance to make a copy of

7.12.2.3 DataProfile() [3/3]

```
MVX2_API Mvx2API::DataProfile::DataProfile (
            DataProfile && other )
```

A move constructor.

Parameters

other	an instance to move

7.12.3 Member Function Documentation

7.12.3.1 GetCompressedTypeGuid()

```
MVX2_API const MVCommon::Guid& Mvx2API::DataProfile::GetCompressedTypeGuid ( ) const
```

A getter of the compressed data type guid.

Returns

compressed data type guid

7.12.3.2 GetPurposeGuid()

```
MVX2_API const MVCommon::Guid& Mvx2API::DataProfile::GetPurposeGuid ( ) const
```

A getter of the purpose guid.

Returns

purpose guid

7.12.3.3 GetTypeGuid()

```
MVX2_API const MVCommon::Guid& Mvx2API::DataProfile::GetTypeGuid ( ) const
```

A getter of the data type guid.

Returns

data type guid

The documentation for this class was generated from the following file:

- public/Mvx2API/data/dataprofiles/DataProfile.h

7.13 Mvx2API::DataProfileHasher Struct Reference

A hasher for DataProfile objects so they can be used in unordered collections.

```
#include <DataProfile.h>
```

Public Member Functions

- MVX2_API size_t operator() (DataProfile const &dataProfile) const

  Calculates a hash value from the object.

7.13.1 Detailed Description

A hasher for DataProfile objects so they can be used in unordered collections.

7.13.2 Member Function Documentation

7.13.2.1 operator()()

```
MVX2_API size_t Mvx2API::DataProfileHasher::operator() (
            DataProfile const & dataProfile ) const
```

Calculates a hash value from the object.

Parameters

dataProfile	an object to calculate the hash value of

Returns

hash value of the object

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/dataprofiles/DataProfile.h

7.14 Mvx2API::DataProfileIterator Class Reference

An iterator over profiles of data contained in a frame.

```
#include <DataProfileIterator.h>
```

Public Types

- using ValueType = DataProfile

  A type of iterated-over elements.

Public Member Functions

- MVX2_API DataProfileIterator (DataProfileIterator const &other)

  A copy constructor.
- MVX2_API DataProfileIterator (DataProfileIterator &&other)

  A move constructor.
- virtual MVX2_API ~DataProfileIterator ()

  A destructor.
- MVX2_API DataProfileIterator & operator++ ()

  A prefix incrementation operator.
- MVX2_API DataProfileIterator operator++ (int)

  A postfix incrementation operator.
- MVX2_API ValueType operator* () const

  Dereferences the iterator.

7.14.1 Detailed Description

An iterator over profiles of data contained in a frame.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 DataProfileIterator() [1/2]

```
MVX2_API Mvx2API::DataProfileIterator::DataProfileIterator (
            DataProfileIterator const & other )
```

A copy constructor.

Parameters

other	an iterator to make a copy of

7.14.2.2 DataProfileIterator() [2/2]

```
MVX2_API Mvx2API::DataProfileIterator::DataProfileIterator (
            DataProfileIterator && other )
```

A move constructor.

Parameters

other	an iterator to move

7.14.3 Member Function Documentation

7.14.3.1 operator∗()

```
MVX2_API ValueType Mvx2API::DataProfileIterator::operator* ( ) const
```

Dereferences the iterator.

Returns

a name of the filter parameter

Exceptions

std::runtime_error	raised when the iterator is invalid

7.14.3.2 operator++() [1/2]

```
MVX2_API DataProfileIterator& Mvx2API::DataProfileIterator::operator++ ( )
```

A prefix incrementation operator.

Moves the iterator to the next element and returns this updated iterator.

Returns

this iterator after it was updated

7.14.3.3 operator++() [2/2]

```
MVX2_API DataProfileIterator Mvx2API::DataProfileIterator::operator++ (
            int  )
```

A postfix incrementation operator.

Moves the iterator to the next element, but returns the original iterator.

Returns

the original unupdated iterator

The documentation for this class was generated from the following file:

- public/Mvx2API/data/dataprofiles/DataProfileIterator.h

7.15 MVX::ErrorHolder Class Reference

A holder of a human readable error string.

`#include <ErrorHolder.h>`

Public Member Functions

- MVX2_API MVCommon::String GetLastError (bool reset=true) const

  Returns a human readable string describing the most recent error.
- MVX2_API void ResetError () const

  Resets the last error to an empty string.

Protected Member Functions

- MVX2_API ErrorHolder ()

  A constructor.
- MVX2_API void SetError (MVCommon::String const &error) const

  Sets the human readable string describing the most recent error.

7.15.1 Detailed Description

A holder of a human readable error string.

7.15.2 Member Function Documentation

7.15.2.1 GetLastError()

```
MVX2_API MVCommon::String MVX::ErrorHolder::GetLastError (
            bool reset = true ) const
```

Returns a human readable string describing the most recent error.

Parameters

reset	indicates whether the error string should be reset afterwards

Returns

the last error string or an empty string if no errors have occured since the last reset

7.15.2.2 SetError()

```
MVX2_API void MVX::ErrorHolder::SetError (
            MVCommon::String const & error ) const  [protected]
```

Sets the human readable string describing the most recent error.

Parameters

error	error string

The documentation for this class was generated from the following file:

- public/Mvx2/core/ErrorHolder.h

7.16 MVX::FilterClassInfo Class Reference

An information data about a filter class.

```
#include <FilterClassInfo.h>
```

Public Member Functions

- MVX2_API FilterClassInfo ()
  A default constructor.
- MVX2_API FilterClassInfo (MVCommon::String const &className, FilterCategory category)
  A constructor.
- MVX2_API ~FilterClassInfo ()
  A destructor.
- MVX2_API MVCommon::String GetClassName () const
  Returns filter's class name.
- MVX2_API MVCommon::String GetNiceClassName () const
  Returns filter's nice class name.
- MVX2_API FilterCategory GetCategory () const
  Returns filter's category.

Static Public Member Functions

- static MVX2_API MVCommon::String NicifyFilterClassName (MVCommon::String const &filterClassName)
  Transforms a technical filter class name into a nice human-readable name.

7.16.1 Detailed Description

An information data about a filter class.

Provides basic information about a filter class.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 FilterClassInfo()

```
MVX2_API MVX::FilterClassInfo::FilterClassInfo (
            MVCommon::String const & className,
            FilterCategory category )
```

A constructor.

Parameters

className	a name of the filter class
category	a category of the filter class

7.16.3 Member Function Documentation

7.16.3.1 GetCategory()

```
MVX2_API FilterCategory MVX::FilterClassInfo::GetCategory ( ) const
```

Returns filter's category.

Returns

filter's category

7.16.3.2 GetClassName()

```
MVX2_API MVCommon::String MVX::FilterClassInfo::GetClassName ( ) const
```

Returns filter's class name.

Returns

filter's class name

7.16.3.3 GetNiceClassName()

```
MVX2_API MVCommon::String MVX::FilterClassInfo::GetNiceClassName ( ) const
```

Returns filter's nice class name.

Returns

filter's nice class name

7.16.3.4 NicifyFilterClassName()

```
static MVX2_API MVCommon::String MVX::FilterClassInfo::NicifyFilterClassName (
            MVCommon::String const & filterClassName )  [static]
```

Transforms a technical filter class name into a nice human-readable name.

Parameters

filterClassName	a class name to transform

Returns

a nice class name

The documentation for this class was generated from the following file:

- public/Mvx2/core/filters/FilterClassInfo.h

7.17 MVX::FilterFactoryIterator Class Reference

An iterator over elements of FilterFactory collection.

```
#include <FilterFactoryIterator.h>
```

Public Types

- using ValueType = MVCommon::Pair< MVCommon::Guid, FilterClassInfo const >

  A type of iterated-over elements of FilterFactory collection.

Public Member Functions

- MVX2_API FilterFactoryIterator (FilterFactoryIterator const &other)

  A copy constructor.
- MVX2_API FilterFactoryIterator (FilterFactoryIterator &&other)

  A move constructor.
- virtual MVX2_API ~FilterFactoryIterator ()

  A destructor.
- MVX2_API FilterFactoryIterator & operator++ ()

  A prefix incrementation operator.
- MVX2_API FilterFactoryIterator operator++ (int)

  A postfix incrementation operator.
- MVX2_API ValueType operator* () const

  Dereferences the iterator.

7.17.1 Detailed Description

An iterator over elements of FilterFactory collection.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 FilterFactoryIterator() [1/2]

```
MVX2_API MVX::FilterFactoryIterator::FilterFactoryIterator (
            FilterFactoryIterator const & other )
```

A copy constructor.

Parameters

other	an iterator to make a copy of

7.17.2.2 FilterFactoryIterator() [2/2]

```
MVX2_API MVX::FilterFactoryIterator::FilterFactoryIterator (
            FilterFactoryIterator && other )
```

A move constructor.

Parameters

other	an iterator to move

7.17.3 Member Function Documentation

7.17.3.1 operator*()

```
MVX2_API ValueType MVX::FilterFactoryIterator::operator* ( ) const
```

Dereferences the iterator.

Returns

a pair of Guid and its filter class info

7.17.3.2 operator++() [1/2]

```
MVX2_API FilterFactoryIterator& MVX::FilterFactoryIterator::operator++ ( )
```

A prefix incrementation operator.

Moves the iterator to the next element and returns this updated iterator.

Returns

this iterator after it was updated

7.17.3.3 operator++() [2/2]

```
MVX2_API FilterFactoryIterator MVX::FilterFactoryIterator::operator++ (
            int  )
```

A postfix incrementation operator.

Moves the iterator to the next element, but returns the original iterator.

Returns

the original unupdated iterator

The documentation for this class was generated from the following file:

- public/Mvx2/core/filters/FilterFactoryIterator.h

7.18 Mvx2API::FilterList Class Reference

A collection of filters.

`#include <FilterList.h>`

Public Member Functions

- MVX2_API FilterList ()
  A constructor.
- MVX2_API FilterList (FilterList const &other)
  A copy-constructor.
- MVX2_API ~FilterList ()
  A destructor.
- MVX2_API void PushBack (SharedFilterPtr const &filter)
  Pushes a filter to the collection.
- MVX2_API SharedFilterPtr & operator[ ] (uint32_t pos)
  Returns a filter at a given index in the collection.
- MVX2_API const SharedFilterPtr & operator[ ] (uint32_t pos) const
  Returns a filter at a given index in the collection.
- MVX2_API uint32_t Count () const
  Returns a count of filters in the collection.
- MVX2_API void Clear ()
  Empties the collection.

7.18.1 Detailed Description

A collection of filters.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 FilterList()

```
MVX2_API Mvx2API::FilterList::FilterList (
            FilterList const & other )
```

A copy-constructor.

Parameters

other	another collection to copy filters from

7.18.3 Member Function Documentation

7.18.3.1 Count()

```
MVX2_API uint32_t Mvx2API::FilterList::Count ( ) const
```

Returns a count of filters in the collection.

Returns

filters count

7.18.3.2 operator[]() [1/2]

```
MVX2_API SharedFilterPtr& Mvx2API::FilterList::operator[] (
            uint32_t pos )
```

Returns a filter at a given index in the collection.

Parameters

pos	an index of index to return

Returns

a filter at the index

7.18.3.3 operator[]() [2/2]

```
MVX2_API const SharedFilterPtr& Mvx2API::FilterList::operator[] (
            uint32_t pos ) const
```

Returns a filter at a given index in the collection.

Parameters

pos	an index of index to return

Returns

a filter at the index

7.18.3.4 PushBack()

```
MVX2_API void Mvx2API::FilterList::PushBack (
            SharedFilterPtr const & filter )
```

Pushes a filter to the collection.

Parameters

filter	a filter to push

The documentation for this class was generated from the following file:

- public/Mvx2API/filters/FilterList.h

7.19 Mvx2API::FilterParameterNameIterator Class Reference

An iterator over names of filter parameters of a SingleFilterGraphNode.

```
#include <FilterParameterNameIterator.h>
```

Public Types

- using ValueType = MVCommon::String
  A type of iterated-over elements.

Public Member Functions

- MVX2_API FilterParameterNameIterator (FilterParameterNameIterator const &other)
  A copy constructor.
- MVX2_API FilterParameterNameIterator (FilterParameterNameIterator &&other)
  A move constructor.
- virtual MVX2_API ~FilterParameterNameIterator ()
  A destructor.
- MVX2_API FilterParameterNameIterator & operator++ ()
  A prefix incrementation operator.
- MVX2_API FilterParameterNameIterator operator++ (int)
  A postfix incrementation operator.
- MVX2_API ValueType operator* () const
  Dereferences the iterator.

7.19.1 Detailed Description

An iterator over names of filter parameters of a SingleFilterGraphNode.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 FilterParameterNameIterator() [1/2]

```
MVX2_API Mvx2API::FilterParameterNameIterator::FilterParameterNameIterator (
            FilterParameterNameIterator const & other )
```

A copy constructor.

Parameters

other	an iterator to make a copy of

7.19.2.2 FilterParameterNameIterator() [2/2]

```
MVX2_API Mvx2API::FilterParameterNameIterator::FilterParameterNameIterator (
            FilterParameterNameIterator && other )
```

A move constructor.

Parameters

other	an iterator to move

7.19.3 Member Function Documentation

7.19.3.1 operator∗()

```
MVX2_API ValueType Mvx2API::FilterParameterNameIterator::operator* ( ) const
```

Dereferences the iterator.

Returns

a name of the filter parameter

7.19.3.2 operator++() [1/2]

```
MVX2_API FilterParameterNameIterator& Mvx2API::FilterParameterNameIterator::operator++ ( )
```

A prefix incrementation operator.

Moves the iterator to the next element and returns this updated iterator.

Returns

this iterator after it was updated

7.19.3.3 operator++() [2/2]

```
MVX2_API FilterParameterNameIterator Mvx2API::FilterParameterNameIterator::operator++ (
            int  )
```

A postfix incrementation operator.

Moves the iterator to the next element, but returns the original iterator.

Returns

the original unupdated iterator

The documentation for this class was generated from the following file:

- public/Mvx2API/filters/FilterParameterNameIterator.h

7.20 Mvx2API::Frame Class Reference

A frame of data.

```
#include <Frame.h>
```

Inherits NonAssignable.

Public Types

- using Iterator = DataProfileIterator

  An alternative type name declaration for DataProfileIterator.

Public Member Functions

- MVX2_API Frame (AtomList const &streams)

  A constructor.
- MVX2_API ~Frame ()

  A destructor.
- MVX2_API uint32_t GetNumStreams () const

  Returns streams count of the frame.
- MVX2_API bool ActivateStreamWithIndex (uint32_t activeStreamIndex)

  Sets a stream of the frame to be active.
- MVX2_API uint32_t GetActiveStreamIndex () const

  Returns index of the currently active stream of the frame.
- MVX2_API uint16_t GetStreamId () const

  Returns ID of the currently active stream of the frame.
- MVX2_API uint32_t GetStreamAtomNr () const

  Returns the atom number in the currently active stream of the frame.
- MVX2_API uint64_t GetStreamAtomTimestamp () const

  Returns the atom timestamp in the currently active stream of the frame.
- MVX2_API bool StreamContainsDataLayer (MVCommon::Guid const &dataLayerGuid, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil(), bool checkCompressedDataLayersToo=true)

  Checks whether the currently active stream contains a data layer with a given guid.
- MVX2_API Iterator DataProfilesBegin () const

  Returns an iterator to the first data profile entry of the active stream.
- MVX2_API Iterator DataProfilesEnd () const

  Returns an iterator to the last data profile entry of the active stream.
- MVX2_API AtomList GetStreams () const

  Returns the collection of streams the frame is composed of.
- MVX2_API SharedAtomPtr GetActiveStream () const

  Returns currently active stream.

7.20.1 Detailed Description

A frame of data.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Frame()

```
MVX2_API Mvx2API::Frame::Frame (
            AtomList const & streams )
```

A constructor.

Parameters

streams	a collection of streams to compose the frame from

7.20.3 Member Function Documentation

7.20.3.1 ActivateStreamWithIndex()

```
MVX2_API bool Mvx2API::Frame::ActivateStreamWithIndex (
            uint32_t activeStreamIndex )
```

Sets a stream of the frame to be active.

Parameters

activeStreamIndex	an index of the stream to activate

Returns

true if the stream was successfully activated

7.20.3.2 DataProfilesBegin()

```
MVX2_API Iterator Mvx2API::Frame::DataProfilesBegin ( ) const
```

Returns an iterator to the first data profile entry of the active stream.

The returned iterator is equal to DataProfilesEnd() iterator when the active stream does not have any data.

Returns

an iterator

7.20.3.3 DataProfilesEnd()

```
MVX2_API Iterator Mvx2API::Frame::DataProfilesEnd ( ) const
```

Returns an iterator to the last data profile entry of the active stream.

Returns

an iterator

7.20.3.4 GetActiveStream()

```
MVX2_API SharedAtomPtr Mvx2API::Frame::GetActiveStream ( ) const
```

Returns currently active stream.

Returns

active stream

7.20.3.5 GetActiveStreamIndex()

```
MVX2_API uint32_t Mvx2API::Frame::GetActiveStreamIndex ( ) const
```

Returns index of the currently active stream of the frame.

Returns

currently active stream's index

7.20.3.6 GetNumStreams()

```
MVX2_API uint32_t Mvx2API::Frame::GetNumStreams ( ) const
```

Returns streams count of the frame.

Returns

streams count

7.20.3.7 GetStreamAtomNr()

```
MVX2_API uint32_t Mvx2API::Frame::GetStreamAtomNr ( ) const
```

Returns the atom number in the currently active stream of the frame.

Returns

atom number in the currently active stream

7.20.3.8 GetStreamAtomTimestamp()

```
MVX2_API uint64_t Mvx2API::Frame::GetStreamAtomTimestamp ( ) const
```

Returns the atom timestamp in the currently active stream of the frame.

Returns

atom timestamp in the currently active stream

7.20.3.9 GetStreamId()

```
MVX2_API uint16_t Mvx2API::Frame::GetStreamId ( ) const
```

Returns ID of the currently active stream of the frame.

Returns

currently active stream's ID

7.20.3.10 GetStreams()

```
MVX2_API AtomList Mvx2API::Frame::GetStreams ( ) const
```

Returns the collection of streams the frame is composed of.

Returns

the streams collection

7.20.3.11 StreamContainsDataLayer()

```
MVX2_API bool Mvx2API::Frame::StreamContainsDataLayer (
            MVCommon::Guid const & dataLayerGuid,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil(),
            bool checkCompressedDataLayersToo = true )
```

Checks whether the currently active stream contains a data layer with a given guid.

Parameters

dataLayerGuid	a guid of the data layer to check
purposeGuid	a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid)
checkCompressedDataLayersToo	an indication whether to check also compressed data layers

Returns

true in case the data layer (compressed and/or uncompressed) is present in the stream

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/Frame.h

7.21 Mvx2API::FrameAccessGraphNode Class Reference

A graph node for direct access to processed MVX frames.

`#include <FrameAccessGraphNode.h>`

Inherits Mvx2API::SingleFilterGraphNode.

Public Member Functions

- MVX2_API FrameAccessGraphNode ()
  A constructor.
- virtual MVX2_API ~FrameAccessGraphNode ()
  A destructor.
- MVX2_API Frame * GetRecentProcessedFrame ()
  Returns the most recent frame processed by a containing graph.

Additional Inherited Members

7.21.1 Detailed Description

A graph node for direct access to processed MVX frames.

Internally maintains a single filter for synchronous access to frames. The same filter is reused even when the graph node is added to multiple graphs.

7.21.2 Member Function Documentation

7.21.2.1 GetRecentProcessedFrame()

```
MVX2_API Frame* Mvx2API::FrameAccessGraphNode::GetRecentProcessedFrame ( )
```

Returns the most recent frame processed by a containing graph.

It is a responsibility of the client to dispose the returned frame.

Returns

the most recent processed frame (may be null, e.g. when MVX stream is over or there was no frame processed in the recent update)

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/FrameAccessGraphNode.h

7.22 Mvx2API::FrameListener Class Reference

A listener for asynchronous reception of frames.

`#include <FrameListener.h>`

Inherits NonAssignable.

Public Member Functions

- virtual MVX2_API ~FrameListener ()
  A destructor.
- virtual MVX2_API void OnFrameProcessed (Frame *pFrame)=0
  A callback executed when a new frame is processed.

7.22.1 Detailed Description

A listener for asynchronous reception of frames.

7.22.2 Member Function Documentation

7.22.2.1 OnFrameProcessed()

```
virtual MVX2_API void Mvx2API::FrameListener::OnFrameProcessed (
            Frame * pFrame )  [pure virtual]
```

A callback executed when a new frame is processed.

Parameters

pFrame	a new frame (it is a responsibility of the client to dispose it)

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/FrameListener.h

7.23 MVX::GenericSharedDataLayerPtr< TDataLayerClass > Class Template Reference

A shared generic smart-pointer to a data layer of a specific data layer class.

`#include <GenericSharedDataLayerPtr.h>`

Public Member Functions

- GenericSharedDataLayerPtr ()

  A constructor.
- GenericSharedDataLayerPtr (TDataLayerClass ∗pDataLayer)

  A constructor.
- GenericSharedDataLayerPtr (SharedDataLayerPtr spDataLayer)

  A constructor.
- GenericSharedDataLayerPtr & operator= (TDataLayerClass ∗pDataLayer)

  Makes the pointer point to a data layer.
- GenericSharedDataLayerPtr & operator= (SharedDataLayerPtr spDataLayer)

  Makes the pointer point to a data layer.
- operator bool () const

  Converts the pointer to a boolean value.
- TDataLayerClass & operator∗ () const

  'Indirection' operator.
- TDataLayerClass ∗ operator-> () const

  'Dereference' operator.
- TDataLayerClass ∗ Get () const

  Returns a raw pointer to the pointed-to data layer.
- operator SharedDataLayerPtr () const

  Converts the generic shared pointer to a non-generic pointer.

7.23.1 Detailed Description

template<typename TDataLayerClass>
class MVX::GenericSharedDataLayerPtr< TDataLayerClass >

A shared generic smart-pointer to a data layer of a specific data layer class.

Allows sharing of the same data layer object by multiple owners and automatically destroys data layer objects when no more pointers point to them.

Template Parameters

TDataLayerClass	a data layer class the shared data layer is of

7.23.2 Constructor & Destructor Documentation

7.23.2.1 GenericSharedDataLayerPtr() [1/3]

```
template<typename TDataLayerClass>
MVX::GenericSharedDataLayerPtr< TDataLayerClass >::GenericSharedDataLayerPtr ( )  [inline]
```

A constructor.

Initializes the pointer with nullptr.

7.23.2.2 GenericSharedDataLayerPtr() [2/3]

```
template<typename TDataLayerClass>
MVX::GenericSharedDataLayerPtr< TDataLayerClass >::GenericSharedDataLayerPtr (
            TDataLayerClass * pDataLayer )  [inline]
```

A constructor.

Parameters

pDataLayer	a data layer to share a pointer to

7.23.2.3 GenericSharedDataLayerPtr() [3/3]

```
template<typename TDataLayerClass>
MVX::GenericSharedDataLayerPtr< TDataLayerClass >::GenericSharedDataLayerPtr (
            SharedDataLayerPtr spDataLayer )  [inline]
```

A constructor.

Parameters

spDataLayer	a non-generic shared data layer pointer to create a generic shared pointer from

In case the data layer pointed-to by the non-generic pointer can not be statically cast to the data layer class of this pointer, this pointer will be set to nullptr value.

7.23.3 Member Function Documentation

7.23.3.1 Get()

```
template<typename TDataLayerClass>
TDataLayerClass* MVX::GenericSharedDataLayerPtr< TDataLayerClass >::Get ( ) const  [inline]
```

Returns a raw pointer to the pointed-to data layer.

Returns

a raw pointer to the pointed-to data layer

7.23.3.2 operator bool()

```
template<typename TDataLayerClass>
MVX::GenericSharedDataLayerPtr< TDataLayerClass >::operator bool ( ) const  [inline]
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to data layer is not null

7.23.3.3 operator SharedDataLayerPtr()

```
template<typename TDataLayerClass>
MVX::GenericSharedDataLayerPtr< TDataLayerClass >::operator SharedDataLayerPtr ( ) const
[inline]
```

Converts the generic shared pointer to a non-generic pointer.

Returns

non-generic shared pointer to a data layer

7.23.3.4 operator*()

```
template<typename TDataLayerClass>
TDataLayerClass& MVX::GenericSharedDataLayerPtr< TDataLayerClass >::operator* ( ) const  [inline]
```

'Indirection' operator.

Returns

a reference to the pointed-to data layer

7.23.3.5 operator->()

```
template<typename TDataLayerClass>
TDataLayerClass* MVX::GenericSharedDataLayerPtr< TDataLayerClass >::operator-> ( ) const
[inline]
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to data layer

7.23.3.6 operator=() [1/2]

```
template<typename TDataLayerClass>
GenericSharedDataLayerPtr& MVX::GenericSharedDataLayerPtr< TDataLayerClass >::operator= (
            SharedDataLayerPtr spDataLayer )  [inline]
```

Makes the pointer point to a data layer.

Destroys previously pointed-to data layer if this was the last pointer pointing to it.

In case the data layer pointed-to by the non-generic pointer can not be statically cast to the data layer class of this pointer, this pointer will be set to nullptr value.

Parameters

spDataLayer	a data layer to point to

Returns

the pointer itself

7.23.3.7 operator=() [2/2]

```
template<typename TDataLayerClass>
GenericSharedDataLayerPtr& MVX::GenericSharedDataLayerPtr< TDataLayerClass >::operator= (
            TDataLayerClass * pDataLayer )  [inline]
```

Makes the pointer point to a data layer.

Destroys previously pointed-to data layer if this was the last pointer pointing to it.

Parameters

pDataLayer	a data layer to point to

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2/core/datalayers/GenericSharedDataLayerPtr.h

7.24 MVX::GenericSharedFilterPtr< TFilterClass > Class Template Reference

A shared generic smart-pointer to a filter of a specific filter class.

`#include <GenericSharedFilterPtr.h>`

Public Member Functions

- GenericSharedFilterPtr ()
  A constructor.
- GenericSharedFilterPtr (TFilterClass *pFilter)
  A constructor.
- GenericSharedFilterPtr (SharedFilterPtr spFilter)
  A constructor.
- GenericSharedFilterPtr & operator= (TFilterClass *pFilter)
  Makes the pointer point to a filter.
- GenericSharedFilterPtr & operator= (SharedFilterPtr spFilter)
  Makes the pointer point to a filter.
- operator bool () const
  Converts the pointer to a boolean value.
- TFilterClass & operator* () const
  'Indirection' operator.
- TFilterClass * operator-> () const
  'Dereference' operator.
- TFilterClass * Get () const
  Returns a raw pointer to the pointed-to filter.
- operator SharedFilterPtr () const
  Converts the generic shared pointer to a non-generic pointer.

7.24.1 Detailed Description

template<typename TFilterClass>
class MVX::GenericSharedFilterPtr< TFilterClass >

A shared generic smart-pointer to a filter of a specific filter class.

Allows sharing of the same filter object by multiple owners and automatically destroys filter objects when no more pointers point to them.

Template Parameters

TFilterClass	a filter class the shared filter is of

7.24.2 Constructor & Destructor Documentation

7.24.2.1 GenericSharedFilterPtr() [1/3]

```
template<typename TFilterClass>
MVX::GenericSharedFilterPtr< TFilterClass >::GenericSharedFilterPtr ( )  [inline]
```

A constructor.

Initializes the pointer with nullptr.

7.24.2.2 GenericSharedFilterPtr() [2/3]

```
template<typename TFilterClass>
MVX::GenericSharedFilterPtr< TFilterClass >::GenericSharedFilterPtr (
            TFilterClass * pFilter )  [inline]
```

A constructor.

Parameters

pFilter	a filter to share a pointer to

7.24.2.3 GenericSharedFilterPtr() [3/3]

```
template<typename TFilterClass>
MVX::GenericSharedFilterPtr< TFilterClass >::GenericSharedFilterPtr (
            SharedFilterPtr spFilter )  [inline]
```

A constructor.

Parameters

spFilter	a non-generic shared filter pointer to create a generic shared pointer from

In case the filter pointed-to by the non-generic pointer can not be statically cast to the filter class of this pointer, this pointer will be set to nullptr value.

7.24.3 Member Function Documentation

7.24.3.1 Get()

```
template<typename TFilterClass>
TFilterClass* MVX::GenericSharedFilterPtr< TFilterClass >::Get ( ) const  [inline]
```

Returns a raw pointer to the pointed-to filter.

Returns

a raw pointer to the pointed-to filter

7.24.3.2 operator bool()

```
template<typename TFilterClass>
MVX::GenericSharedFilterPtr< TFilterClass >::operator bool ( ) const  [inline]
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to filter is not null

7.24.3.3 operator SharedFilterPtr()

```
template<typename TFilterClass>
MVX::GenericSharedFilterPtr< TFilterClass >::operator SharedFilterPtr ( ) const  [inline]
```

Converts the generic shared pointer to a non-generic pointer.

Returns

non-generic shared pointer to a filter

7.24.3.4 operator∗()

```
template<typename TFilterClass>
TFilterClass& MVX::GenericSharedFilterPtr< TFilterClass >::operator* ( ) const  [inline]
```

'Indirection' operator.

Returns

a reference to the pointed-to filter

7.24.3.5 operator->()

```
template<typename TFilterClass>
TFilterClass* MVX::GenericSharedFilterPtr< TFilterClass >::operator-> ( ) const  [inline]
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to filter

7.24.3.6 operator=() [1/2]

```
template<typename TFilterClass>
GenericSharedFilterPtr& MVX::GenericSharedFilterPtr< TFilterClass >::operator= (
            SharedFilterPtr spFilter )  [inline]
```

Makes the pointer point to a filter.

Destroys previously pointed-to filter if this was the last pointer pointing to it.

In case the filter pointed-to by the non-generic pointer can not be statically cast to the filter class of this pointer, this pointer will be set to nullptr value.

Parameters

spFilter	a filter to point to

Returns

the pointer itself

7.24.3.7 operator=() [2/2]

```
template<typename TFilterClass>
GenericSharedFilterPtr& MVX::GenericSharedFilterPtr< TFilterClass >::operator= (
            TFilterClass * pFilter )  [inline]
```

Makes the pointer point to a filter.

Destroys previously pointed-to filter if this was the last pointer pointing to it.

Parameters

pFilter	a filter to point to

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2/core/filters/GenericSharedFilterPtr.h

7.25 Mvx2API::Graph Class Reference

A graph of data-processing nodes.

`#include <Graph.h>`

Inherits NonAssignable.

Public Member Functions

- virtual MVX2_API ~Graph ()
  A destructor.
- MVX2_API bool Reinitialize ()
  Reinitializes the graph.

7.25.1 Detailed Description

A graph of data-processing nodes.

7.25.2 Member Function Documentation

7.25.2.1 Reinitialize()

`MVX2_API bool Mvx2API::Graph::Reinitialize ( )`

Reinitializes the graph.

Fails if the graph is currently in a running state. Otherwise all filters of the graph are deinitialized, removed from it, reinitialized and readded to the graph. If any of the actions on any of the filters fails, the graph may remain in an invalid state and may not be usable anymore.

The purpose of the function is to allow modification of 'hard' parameters of filters, which normally have no impact on the graph once they have been initialized. These parameters may significantly change behaviour of filters and the whole graph.

Returns

true if the reinitialization succeeds

The documentation for this class was generated from the following file:

- public/Mvx2API/core/Graph.h

7.26 Mvx2API::GraphBuilder Class Reference

A builder of data-processing graphs.

`#include <GraphBuilder.h>`

Inherits NonAssignable.

Inherited by Mvx2API::ManualGraphBuilder.

Public Types

- using Iterator = DataProfileIterator

  An alternative type name declaration for DataProfileIterator.

Public Member Functions

- virtual MVX2_API ~GraphBuilder ()

  A destructor.
- virtual MVX2_API Graph * CompileGraphAndReset ()=0

  Compiles a graph being built and resets the builder for another graph to be built.
- virtual MVX2_API void Reset ()=0

  Resets the builder by removing all already appended graph nodes.
- virtual MVX2_API bool Refresh ()=0

  Refreshes the builder.
- virtual MVX2_API bool ContainsDataProfile (MVCommon::Guid const &dataLayerGuid, MVCommon::Guid const &purposeGuid, bool checkCompressedDataLayersToo=true)=0

  Checks whether the graph being built in its current state contains a data profile with a given guid.
- virtual MVX2_API Iterator DataProfilesBegin () const =0

  Returns an iterator to the first data profile entry of the graph being built in its current state.
- virtual MVX2_API Iterator DataProfilesEnd () const =0

  Returns an iterator to the last data profile entry of the graph being built in its current state.

Protected Member Functions

- MVX2_API GraphBuilder ()

  A constructor.

7.26.1 Detailed Description

A builder of data-processing graphs.

7.26.2 Member Function Documentation

7.26.2.1 CompileGraphAndReset()

```
virtual MVX2_API Graph* Mvx2API::GraphBuilder::CompileGraphAndReset ( )  [pure virtual]
```

Compiles a graph being built and resets the builder for another graph to be built.

The graph is being reinitialized during the compilation so filter parameter changes which would potentially modify its behaviour can take effect. However, since the reinitialization of the graph may fail, the compilation of the graph may fail as well. In such case the graph being built is not replaced by a new graph in the builder and after fixing the filter parameters, the graph compilation may be attempted again.

Returns

a compiled graph or nullptr if the graph reinitialization fails

Implemented in Mvx2API::ManualGraphBuilder.

7.26.2.2 ContainsDataProfile()

```
virtual MVX2_API bool Mvx2API::GraphBuilder::ContainsDataProfile (
            MVCommon::Guid const & dataLayerGuid,
            MVCommon::Guid const & purposeGuid,
            bool checkCompressedDataLayersToo = true )  [pure virtual]
```

Checks whether the graph being built in its current state contains a data profile with a given guid.

Parameters

dataLayerGuid	a guid of the data layer to check
purposeGuid	a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid)
checkCompressedDataLayersToo	an indication whether to check also compressed data layers

Returns

true in case the data profile (compressed and/or uncompressed data layer) is present in the graph

Implemented in Mvx2API::ManualGraphBuilder.

7.26.2.3 DataProfilesBegin()

```
virtual MVX2_API Iterator Mvx2API::GraphBuilder::DataProfilesBegin ( ) const  [pure virtual]
```

Returns an iterator to the first data profile entry of the graph being built in its current state.

The returned iterator is equal to DataProfilesEnd() iterator when the graph is empty or in an error state.

Returns

an iterator

Implemented in Mvx2API::ManualGraphBuilder.

7.26.2.4 DataProfilesEnd()

```
virtual MVX2_API Iterator Mvx2API::GraphBuilder::DataProfilesEnd ( ) const  [pure virtual]
```

Returns an iterator to the last data profile entry of the graph being built in its current state.

Returns

an iterator

Implemented in Mvx2API::ManualGraphBuilder.

7.26.2.5 Refresh()

```
virtual MVX2_API bool Mvx2API::GraphBuilder::Refresh ( )  [pure virtual]
```

Refreshes the builder.

Restarts creation of the graph being built and re-adds all already appended graph nodes to it.

Returns

true in case the graph creation was successfully refreshed, false otherwise

Implemented in Mvx2API::ManualGraphBuilder.

The documentation for this class was generated from the following file:

- public/Mvx2API/core/GraphBuilder.h

7.27 Mvx2API::GraphNode Class Reference

A processing node.

```
#include <GraphNode.h>
```

Inherits NonAssignable.

Inherited by Mvx2API::AutoCompressorGraphNode, Mvx2API::AutoDecompressorGraphNode, Mvx2API::BlockGraphNode, Mvx2API::InjectMemoryDataGraphNode, Mvx2API::ManualLiveFrameSourceGraphNode, Mvx2API::ManualOfflineFrameSourceGrap and Mvx2API::SingleFilterGraphNode.

Public Member Functions

- MVX2_API GraphNode ()

  A constructor.
- virtual MVX2_API ~GraphNode ()

  A destructor.
- virtual MVX2_API void GetFilters (SharedFilterPtr spPrecedingFilter, FilterList &targetFilterList)=0

  A getter of all MVX filters created and managed internally by the node.

7.27.1 Detailed Description

A processing node.

Each node can be added to multiple graphs as long as at any point in time it only is added to only one. A graph that the graph node is currently in must first be completely destroyed before the graph node can be added to another graph. Attempts to add the same graph node to multiple graphs at the same time will end with a failure.

What happens when a graph node was in a graph and is then added to another graph depends on its implementation. Some graph nodes may permanently keep the same collection of processing filters, reusing them this way effectively in multiple graphs. Other implementations may create a new collection of filters each time they are added to a graph.

7.27.2 Member Function Documentation

7.27.2.1 GetFilters()

```
virtual MVX2_API void Mvx2API::GraphNode::GetFilters (
            SharedFilterPtr spPrecedingFilter,
            FilterList & targetFilterList )  [pure virtual]
```

A getter of all MVX filters created and managed internally by the node.

Parameters

spPrecedingFilter	a pointer to the last filter preceding the filters to be added by the graph node, so the added filters can be initialized with it in case they need to be
targetFilterList	a collection to add filters to

Exceptions

std::runtime_error	raised when there is an error getting filters from the graph node

The documentation for this class was generated from the following file:

- public/Mvx2API/core/GraphNode.h

7.28 Mvx2API::GraphRunner Class Reference

A runner of data-processing graphs.

`#include <GraphRunner.h>`

Inherits NonAssignable.

Inherited by Mvx2API::AutoSequentialGraphRunner, Mvx2API::ManualSequentialGraphRunner, and Mvx2API::RandomAccessGraph

Public Member Functions

- virtual MVX2_API ~GraphRunner ()
  A destructor.
- virtual MVX2_API SourceInfo * GetSourceInfo () const =0
  Retrieves source information about the currently open MVX source.

7.28.1 Detailed Description

A runner of data-processing graphs.

7.28.2 Member Function Documentation

7.28.2.1 GetSourceInfo()

```
virtual MVX2_API SourceInfo* Mvx2API::GraphRunner::GetSourceInfo ( ) const  [pure virtual]
```

Retrieves source information about the currently open MVX source.

Returns

information about the current MVX source or null if no source is open

Implemented in Mvx2API::AutoSequentialGraphRunner, Mvx2API::ManualSequentialGraphRunner, and Mvx2API::RandomAccessGr

The documentation for this class was generated from the following file:

- public/Mvx2API/core/GraphRunner.h

7.29 MVX::IMVXLoggerInstanceListener Class Reference

An interface of listeners to MVX logger instance changes.

`#include <Logger.h>`

Public Member Functions

- virtual MVX2_API ~IMVXLoggerInstanceListener ()
  A virtual destructor.
- virtual MVX2_API void OnMVXLoggerInstanceChanged (MVCommon::WeakLoggerPtr wpLogger)=0
  A callback executed when MVX logger instance changes.

7.29.1 Detailed Description

An interface of listeners to MVX logger instance changes.

7.29.2 Member Function Documentation

7.29.2.1 OnMVXLoggerInstanceChanged()

```
virtual MVX2_API void MVX::IMVXLoggerInstanceListener::OnMVXLoggerInstanceChanged (
            MVCommon::WeakLoggerPtr wpLogger )  [pure virtual]
```

A callback executed when MVX logger instance changes.

Parameters

wpLogger	a weak pointer to the new logger instance

The documentation for this class was generated from the following file:

- public/Mvx2/utils/Logger.h

7.30 Mvx2API::InjectFileDataGraphNode Class Reference

A graph node for injecting binary data from files to frames.

`#include <InjectFileDataGraphNode.h>`

Inherits Mvx2API::SingleFilterGraphNode.

Public Member Functions

- MVX2_API InjectFileDataGraphNode (MVCommon::Guid const &dataPurposeGuid)

  A constructor.
- MVX2_API ~InjectFileDataGraphNode ()

  A destructor.
- MVX2_API void SetFile (MVCommon::String const &filePath)

  Sets a new file to inject the binary content of to frames.

Additional Inherited Members

7.30.1 Detailed Description

A graph node for injecting binary data from files to frames.

Internally maintains a single data-injecting filter. The same filter is reused even when the graph node is added to multiple graphs.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 InjectFileDataGraphNode()

```
MVX2_API Mvx2API::InjectFileDataGraphNode::InjectFileDataGraphNode (
            MVCommon::Guid const & dataPurposeGuid )
```

A constructor.

Parameters

dataPurposeGuid	purpose guid of the injected data

7.30.3 Member Function Documentation

7.30.3.1 SetFile()

```
MVX2_API void Mvx2API::InjectFileDataGraphNode::SetFile (
            MVCommon::String const & filePath )
```

Sets a new file to inject the binary content of to frames.

Parameters

filePath	a path of the file

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/InjectFileDataGraphNode.h

7.31 Mvx2API::InjectMemoryDataGraphNode Class Reference

A graph node for injecting binary data from memory to frames.

`#include <InjectMemoryDataGraphNode.h>`

Inherits Mvx2API::GraphNode.

Public Member Functions

- MVX2_API InjectMemoryDataGraphNode (MVCommon::Guid const &dataPurposeGuid)
  A constructor.
- MVX2_API ~InjectMemoryDataGraphNode ()
  A destructor.
- MVX2_API void SetData (MVCommon::ByteArray const &data)
  Sets a new data to inject to frames.

7.31.1 Detailed Description

A graph node for injecting binary data from memory to frames.

Internally maintains a single data-injecting filter. The same filter is reused even when the graph node is added to multiple graphs.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 InjectMemoryDataGraphNode()

```
MVX2_API Mvx2API::InjectMemoryDataGraphNode::InjectMemoryDataGraphNode (
            MVCommon::Guid const & dataPurposeGuid )
```

A constructor.

Parameters

dataPurposeGuid	purpose guid of the injected data

Exceptions

std::runtime_error	raised in case the creation of the internal filter fails

7.31.3 Member Function Documentation

7.31.3.1 SetData()

```
MVX2_API void Mvx2API::InjectMemoryDataGraphNode::SetData (
            MVCommon::ByteArray const & data )
```

Sets a new data to inject to frames.

Parameters

data	a data

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/InjectMemoryDataGraphNode.h

7.32 Mvx2API::InputEvent Struct Reference

An input event structure.

```
#include <InputEvent.h>
```

Inherited by Mvx2API::KeyDownEvent, Mvx2API::KeyUpEvent, Mvx2API::MouseDoubleClickEvent, Mvx2API::MouseDownEvent, Mvx2API::MouseMoveEvent, Mvx2API::MouseUpEvent, and Mvx2API::MouseWheelEvent.

Public Member Functions

- virtual MVX2_API ~InputEvent ()

  A destructor.

Protected Member Functions

- MVX2_API InputEvent ()

  A constructor.

7.32.1 Detailed Description

An input event structure.

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/InputEvent.h

7.33 Mvx2API::IParameterValueChangedListener Class Reference

A listener for changes of graph nodes' parameters.

`#include <IParameterValueChangedListener.h>`

Inherits NonAssignable.

Public Member Functions

- virtual MVX2_API ~IParameterValueChangedListener ()

  A destructor.
- virtual MVX2_API void OnParameterValueChanged (GraphNode *pGraphNode, MVCommon::String const &parameterName, MVCommon::String const &parameterValueStr)=0

  A callback executed when a parameter of a graph node changes its value.

7.33.1 Detailed Description

A listener for changes of graph nodes' parameters.

7.33.2 Member Function Documentation

7.33.2.1 OnParameterValueChanged()

```
virtual MVX2_API void Mvx2API::IParameterValueChangedListener::OnParameterValueChanged (
            GraphNode * pGraphNode,
            MVCommon::String const & parameterName,
            MVCommon::String const & parameterValueStr )  [pure virtual]
```

A callback executed when a parameter of a graph node changes its value.

Parameters

pGraphNode	a graph node containing the changed parameter
parameterName	name of the changed parameter
parameterValueStr	parameter's new value in a string form

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/IParameterValueChangedListener.h

7.34 Mvx2API::KeyDownEvent Struct Reference

A 'key down' event.

`#include <KeyDownEvent.h>`

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API KeyDownEvent (int32_t key)
  A constructor.
- MVX2_API KeyDownEvent (KeyDownEvent const &other)
  A copy constructor.
- MVX2_API KeyDownEvent (KeyDownEvent &&other)
  A move constructor.
- virtual MVX2_API ~KeyDownEvent ()
  A destructor.

Additional Inherited Members

7.34.1 Detailed Description

A 'key down' event.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 KeyDownEvent() [1/3]

```
MVX2_API Mvx2API::KeyDownEvent::KeyDownEvent (
            int32_t key )
```

A constructor.

Parameters

key	a value of key pressed down

7.34.2.2 KeyDownEvent() [2/3]

```
MVX2_API Mvx2API::KeyDownEvent::KeyDownEvent (
            KeyDownEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.34.2.3 KeyDownEvent() [3/3]

```
MVX2_API Mvx2API::KeyDownEvent::KeyDownEvent (
            KeyDownEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/KeyDownEvent.h

7.35 Mvx2API::KeyUpEvent Struct Reference

A 'key up' event.

`#include <KeyUpEvent.h>`

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API KeyUpEvent (int32_t key)
  A constructor.
- MVX2_API KeyUpEvent (KeyUpEvent const &other)
  A copy constructor.
- MVX2_API KeyUpEvent (KeyUpEvent &&other)
  A move constructor.
- virtual MVX2_API ~KeyUpEvent ()
  A destructor.

Additional Inherited Members

7.35.1 Detailed Description

A 'key up' event.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 KeyUpEvent() [1/3]

```
MVX2_API Mvx2API::KeyUpEvent::KeyUpEvent (
            int32_t key )
```

A constructor.

Parameters

key	a value of key released

7.35.2.2 KeyUpEvent() [2/3]

```
MVX2_API Mvx2API::KeyUpEvent::KeyUpEvent (
            KeyUpEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.35.2.3 KeyUpEvent() [3/3]

```
MVX2_API Mvx2API::KeyUpEvent::KeyUpEvent (
            KeyUpEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/KeyUpEvent.h

7.36 Mvx2API::ManualGraphBuilder Class Reference

A manual builder of data-processing graphs.

`#include <ManualGraphBuilder.h>`

Inherits Mvx2API::GraphBuilder.

Public Member Functions

- MVX2_API ManualGraphBuilder ()
  A constructor.
- MVX2_API ~ManualGraphBuilder ()
  A destructor.
- MVX2_API ManualGraphBuilder & operator<< (GraphNode &graphNode)
  Appends a graph node to the graph being built.
- MVX2_API ManualGraphBuilder & operator<< (GraphNode &&graphNode)
  Appends a graph node to the graph being built.
- MVX2_API void AppendGraphNode (GraphNode &graphNode)
  Appends a graph node to the graph being built.
- virtual MVX2_API Graph * CompileGraphAndReset () override
  Compiles a graph being built and resets the builder for another graph to be built.
- virtual MVX2_API void Reset () override
  Resets the builder by removing all already appended graph nodes.
- virtual MVX2_API bool Refresh () override
  Refreshes the builder.
- virtual MVX2_API bool ContainsDataProfile (MVCommon::Guid const &dataLayerGuid, MVCommon::Guid const &purposeGuid, bool checkCompressedDataLayersToo=true) override
  Checks whether the graph being built in its current state contains a data profile with a given guid.
- virtual MVX2_API Iterator DataProfilesBegin () const override
  Returns an iterator to the first data profile entry of the graph being built in its current state.
- virtual MVX2_API Iterator DataProfilesEnd () const override
  Returns an iterator to the last data profile entry of the graph being built in its current state.

Additional Inherited Members

7.36.1 Detailed Description

A manual builder of data-processing graphs.

7.36.2 Member Function Documentation

7.36.2.1 AppendGraphNode()

```
MVX2_API void Mvx2API::ManualGraphBuilder::AppendGraphNode (
            GraphNode & graphNode )
```

Appends a graph node to the graph being built.

Parameters

graphNode	a graph node to append

Exceptions

std::runtime_error	raised when the graph builder fails to append the graph node to the graph

7.36.2.2 CompileGraphAndReset()

```
virtual MVX2_API Graph* Mvx2API::ManualGraphBuilder::CompileGraphAndReset ( )  [override],
[virtual]
```

Compiles a graph being built and resets the builder for another graph to be built.

The graph is being reinitialized during the compilation so filter parameter changes which would potentially modify its behaviour can take effect. However, since the reinitialization of the graph may fail, the compilation of the graph may fail as well. In such case the graph being built is not replaced by a new graph in the builder and after fixing the filter parameters, the graph compilation may be attempted again.

Returns

a compiled graph or nullptr if the graph reinitialization fails

Implements Mvx2API::GraphBuilder.

7.36.2.3 ContainsDataProfile()

```
virtual MVX2_API bool Mvx2API::ManualGraphBuilder::ContainsDataProfile (
            MVCommon::Guid const & dataLayerGuid,
            MVCommon::Guid const & purposeGuid,
            bool checkCompressedDataLayersToo = true )  [override], [virtual]
```

Checks whether the graph being built in its current state contains a data profile with a given guid.

Parameters

dataLayerGuid	a guid of the data layer to check
purposeGuid	a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid)
checkCompressedDataLayersToo	an indication whether to check also compressed data layers

Returns

true in case the data profile (compressed and/or uncompressed data layer) is present in the graph

Implements Mvx2API::GraphBuilder.

7.36.2.4 DataProfilesBegin()

```
virtual MVX2_API Iterator Mvx2API::ManualGraphBuilder::DataProfilesBegin ( ) const  [override],
[virtual]
```

Returns an iterator to the first data profile entry of the graph being built in its current state.

The returned iterator is equal to DataProfilesEnd() iterator when the graph is empty or in an error state.

Returns

an iterator

Implements Mvx2API::GraphBuilder.

7.36.2.5 DataProfilesEnd()

```
virtual MVX2_API Iterator Mvx2API::ManualGraphBuilder::DataProfilesEnd ( ) const  [override],
[virtual]
```

Returns an iterator to the last data profile entry of the graph being built in its current state.

Returns

an iterator

Implements Mvx2API::GraphBuilder.

7.36.2.6 operator<<() [1/2]

```
MVX2_API ManualGraphBuilder& Mvx2API::ManualGraphBuilder::operator<< (
            GraphNode && graphNode )
```

Appends a graph node to the graph being built.

Parameters

graphNode	a graph node to append

Returns

the builder itself

Exceptions

std::runtime_error	raised when the graph builder fails to append the graph node to the graph

7.36.2.7 operator<<() [2/2]

```
MVX2_API ManualGraphBuilder& Mvx2API::ManualGraphBuilder::operator<< (
            GraphNode & graphNode )
```

Appends a graph node to the graph being built.

Parameters

graphNode	a graph node to append

Returns

the builder itself

Exceptions

std::runtime_error	raised when the graph builder fails to append the graph node to the graph

7.36.2.8 Refresh()

```
virtual MVX2_API bool Mvx2API::ManualGraphBuilder::Refresh ( )  [override], [virtual]
```

Refreshes the builder.

Restarts creation of the graph being built and re-adds all already appended graph nodes to it.

Returns

true in case the graph creation was successfully refreshed, false otherwise

Implements Mvx2API::GraphBuilder.

The documentation for this class was generated from the following file:

- public/Mvx2API/core/ManualGraphBuilder.h

7.37 Mvx2API::ManualLiveFrameSourceGraphNode Class Reference

A source graph node for manual production of MVX frames.

`#include <ManualLiveFrameSourceGraphNode.h>`

Inherits Mvx2API::GraphNode.

Public Member Functions

- MVX2_API ManualLiveFrameSourceGraphNode ()

  A constructor.
- virtual MVX2_API ~ManualLiveFrameSourceGraphNode ()

  A destructor.
- MVX2_API bool ClearCacheAndReinitializeProperties (Frame const *pFrame, float declaredFPS, bool reassignSequentialFrameNumbers=true)

  Clears the queue of frames and reinitializes the internal filter's properties based on the first stream of a provided frame.
- MVX2_API bool PropertiesAreInitialized () const

  Checks whether the internal filter's properties have been initialized already.
- MVX2_API void ClearCache (bool revertReassignedFrameNumbers=true)

  Clears the queue of frames.
- MVX2_API bool PushFrame (Frame const *pFrame)

  Pushes another frame to the queue.

7.37.1 Detailed Description

A source graph node for manual production of MVX frames.

Allows to add frames on the fly, while the graph node is in a running graph.

Internally maintains a single filter for synchronous access to frames. The same filter is reused even when the graph node is added to multiple graphs.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 ManualLiveFrameSourceGraphNode()

`MVX2_API Mvx2API::ManualLiveFrameSourceGraphNode::ManualLiveFrameSourceGraphNode ( )`

A constructor.

Exceptions

std::runtime_error	raised in case the creation of the internal filter fails

7.37.3 Member Function Documentation

7.37.3.1 ClearCache()

```
MVX2_API void Mvx2API::ManualLiveFrameSourceGraphNode::ClearCache (
            bool revertReassignedFrameNumbers = true )
```

Clears the queue of frames.

Parameters

revertReassignedFrameNumbers	in case the reassignment of frame numbers is enabled, determines whether frame numbers assigned to the to-be removed frames shall be reused for potential new frames pushed to the filter afterwards

7.37.3.2 ClearCacheAndReinitializeProperties()

```
MVX2_API bool Mvx2API::ManualLiveFrameSourceGraphNode::ClearCacheAndReinitializeProperties (
            Frame const * pFrame,
            float declaredFPS,
            bool reassignSequentialFrameNumbers = true )
```

Clears the queue of frames and reinitializes the internal filter's properties based on the first stream of a provided frame.

Graph node can only be reinitialized while it was not yet added to a graph. Reinitialization causes the remaining cached frames to be destroyed, since they may not be valid after the reinitialization.

The properties of the filter that are initialized include the filter's output profile and stream information.

Parameters

pFrame	a frame to reinitialize the internal filter's properties with
declaredFPS	declared rate of frames production
reassignSequentialFrameNumbers	determines whether the graph node should assign new (sequential) numbers to frames pushed to its output, or leave original numbers in place

Returns

true if the reinitialization was successful

7.37.3.3 PropertiesAreInitialized()

```
MVX2_API bool Mvx2API::ManualLiveFrameSourceGraphNode::PropertiesAreInitialized ( ) const
```

Checks whether the internal filter's properties have been initialized already.

Returns

true if the properties have been already initialized

7.37.3.4 PushFrame()

```
MVX2_API bool Mvx2API::ManualLiveFrameSourceGraphNode::PushFrame (
            Frame const * pFrame )
```

Pushes another frame to the queue.

A frame will only be pushed if it has exactly the same number of streams as was declared during the initialization of the graph node, and if data layers of all its streams satisfy the output profile of the internal filter (i.e. the filter's properties must be initialized already).

Parameters

pFrame	a frame to push

Returns

true if the frame was pushed to the queue

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/ManualLiveFrameSourceGraphNode.h

7.38 Mvx2API::ManualOfflineFrameSourceGraphNode Class Reference

A source graph node for manual production of MVX frames.

```
#include <ManualOfflineFrameSourceGraphNode.h>
```

Inherits Mvx2API::GraphNode.

Public Member Functions

- MVX2_API ManualOfflineFrameSourceGraphNode ()

  A constructor.
- virtual MVX2_API ~ManualOfflineFrameSourceGraphNode ()

  A destructor.
- MVX2_API bool ClearCacheAndReinitializeProperties (Frame const *pFrame, float declaredFPS, bool reassignSequentialFrameNumbers=true)

  Clears the collection of frames and reinitializes the internal filter's properties based on the first stream of a provided frame.
- MVX2_API bool PropertiesAreInitialized () const

  Checks whether the internal filter's properties have been initialized already.
- MVX2_API bool ClearCache ()

  Clears the collection of frames.
- MVX2_API bool PushFrame (Frame const *pFrame)

  Pushes another frame to the collection.

7.38.1 Detailed Description

A source graph node for manual production of MVX frames.

Its internal queue of frames must be prepared before the graph node is added to a graph and can not be changed afterwards.

Internally maintains a single filter for synchronous access to frames. The same filter is reused even when the graph node is added to multiple graphs.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 ManualOfflineFrameSourceGraphNode()

```
MVX2_API Mvx2API::ManualOfflineFrameSourceGraphNode::ManualOfflineFrameSourceGraphNode ( )
```

A constructor.

Exceptions

std::runtime_error	raised in case the creation of the internal filter fails

7.38.3 Member Function Documentation

7.38.3.1 ClearCache()

```
MVX2_API bool Mvx2API::ManualOfflineFrameSourceGraphNode::ClearCache ( )
```

Clears the collection of frames.

Collection of frames can only be cleared while the graph node was not yet added to a graph.

Returns

true if cache was cleared, false otherwise

7.38.3.2 ClearCacheAndReinitializeProperties()

```
MVX2_API bool Mvx2API::ManualOfflineFrameSourceGraphNode::ClearCacheAndReinitializeProperties
(
            Frame const * pFrame,
            float declaredFPS,
            bool reassignSequentialFrameNumbers = true )
```

Clears the collection of frames and reinitializes the internal filter's properties based on the first stream of a provided frame.

Graph node can only be reinitialized while it was not yet added to a graph. Reinitialization causes cached frames to be destroyed, since they may not be valid after the reinitialization.

The properties of the filter that are initialized include the filter's output profile and stream information.

Parameters

pFrame	a frame to reinitialize the internal filter's properties with
declaredFPS	declared rate of frames production
reassignSequentialFrameNumbers	determines whether the filter should assign new (sequential) numbers to frames pushed to its output, or leave original numbers in place

Returns

true if the reinitialization was successful

7.38.3.3 PropertiesAreInitialized()

```
MVX2_API bool Mvx2API::ManualOfflineFrameSourceGraphNode::PropertiesAreInitialized ( ) const
```

Checks whether the internal filter's properties have been initialized already.

Returns

true if the properties have been already initialized

7.38.3.4 PushFrame()

```
MVX2_API bool Mvx2API::ManualOfflineFrameSourceGraphNode::PushFrame (
            Frame const * pFrame )
```

Pushes another frame to the collection.

Frames can only be pushed to the collection while the graph node was not yet added to a graph.

A frame will only be pushed if it has exactly the same number of streams as was declared during the initialization of the graph node, and if data layers of all its streams satisfy the output profile of the internal filter (i.e. the filter's properties must be initialized already).

Parameters

pFrame	a frame to push

Returns

true if the frame was pushed to the collection

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/ManualOfflineFrameSourceGraphNode.h

7.39 Mvx2API::ManualSequentialGraphRunner Class Reference

A sequential runner of data-processing graphs with manual updates-invocation.

```
#include <ManualSequentialGraphRunner.h>
```

Inherits Mvx2API::GraphRunner.

Public Member Functions

- MVX2_API ManualSequentialGraphRunner (Graph *graph)

  A constructor.
- virtual MVX2_API ~ManualSequentialGraphRunner ()

  A destructor.
- MVX2_API bool RestartWithPlaybackMode (RunnerPlaybackMode playbackMode)

  Restarts the runner with a new playback mode.
- MVX2_API bool ProcessNextFrame ()

  Processes a subsequent frame (depending on the current playback mode).
- MVX2_API void SeekFrame (uint32_t frameID)

  Sets a frame with a given ID as the next to be processed.
- virtual MVX2_API SourceInfo * GetSourceInfo () const override

  Retrieves source information about the currently open MVX source.

7.39.1 Detailed Description

A sequential runner of data-processing graphs with manual updates-invocation.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 ManualSequentialGraphRunner()

```
MVX2_API Mvx2API::ManualSequentialGraphRunner::ManualSequentialGraphRunner (
            Graph * graph )
```

A constructor.

Parameters

graph	a graph to create the runner for

7.39.3 Member Function Documentation

7.39.3.1 GetSourceInfo()

```
virtual MVX2_API SourceInfo* Mvx2API::ManualSequentialGraphRunner::GetSourceInfo ( ) const
[override], [virtual]
```

Retrieves source information about the currently open MVX source.

Returns

information about the current MVX source or null if no source is open

Implements Mvx2API::GraphRunner.

7.39.3.2 ProcessNextFrame()

```
MVX2_API bool Mvx2API::ManualSequentialGraphRunner::ProcessNextFrame ( )
```

Processes a subsequent frame (depending on the current playback mode).

Returns

true if no error occured during the processing

7.39.3.3 RestartWithPlaybackMode()

```
MVX2_API bool Mvx2API::ManualSequentialGraphRunner::RestartWithPlaybackMode (
            RunnerPlaybackMode playbackMode )
```

Restarts the runner with a new playback mode.

Parameters

playbackMode	a playback mode to restart with

Returns

true if the playback mode was successfully changed

7.39.3.4 SeekFrame()

```
MVX2_API void Mvx2API::ManualSequentialGraphRunner::SeekFrame (
            uint32_t frameID )
```

Sets a frame with a given ID as the next to be processed.

Parameters

frameID	an ID of the frame to be processed next

The documentation for this class was generated from the following file:

- public/Mvx2API/runners/ManualSequentialGraphRunner.h

7.40 Mvx2API::MeshData Class Reference

A class containing data of a single mesh.

`#include <MeshData.h>`

Inherits NonAssignable.

Public Member Functions

- virtual MVX2_API ~MeshData ()
  A destructor.
- MVX2_API uint32_t GetNumVertices () const
  A getter of the vertices count.
- MVX2_API const float * GetVertices () const
  A getter of the raw pointer to vertices collection.
- MVX2_API bool CopyVertices (float *targetVertices) const
  Copies vertices collection to the target array.
- MVX2_API bool CopyVerticesVec3 (Vec3Data *targetVertices) const
  Copies vertices collection to the target array.
- MVX2_API uint32_t GetNumNormals () const
  A getter of the normals count.
- MVX2_API const float * GetNormals () const
  A getter of the raw pointer to normals collection.
- MVX2_API bool CopyNormals (float *targetNormals) const
  Copies vertex normals collection to the target array.
- MVX2_API bool CopyNormalsVec3 (Vec3Data *targetNormals) const
  Copies vertex normals collection to the target array.
- MVX2_API uint32_t GetNumColors () const
  A getter of the colors count.
- MVX2_API const uint8_t * GetColorsRGB () const
  A getter of the raw pointer to RGB colors collection.
- MVX2_API bool CopyColorsRGB (uint8_t *targetColors) const
  Copies vertex RGB colors collection to the target array.
- MVX2_API bool CopyColorsColRGBA (ColRGBAData *targetColors) const
  Copies vertex RGBA colors collection to the target array.
- MVX2_API uint32_t GetNumUVs () const
  A getter of the UVs count.
- MVX2_API const float * GetUVs () const
  A getter of the raw pointer to UVs collection.
- MVX2_API bool CopyUVs (float *targetUVs) const
  Copies vertex UVs collection to the target array.
- MVX2_API bool CopyUVsVec2 (Vec2Data *targetUVs) const
  Copies vertex UVs collection to the target array.
- MVX2_API uint32_t GetNumIndices () const

A getter of the indices count.

- MVX2_API const uint32_t * GetIndices () const

  A getter of the raw pointer to indices collection.

- MVX2_API bool CopyIndices (uint32_t *targetIndices) const

  Copies vertex indices collection to the target array.

- MVX2_API const float * GetBoundingBox () const

  A getter of the raw pointer to bounding box data.

- MVX2_API bool CopyBoundingBox (float *targetBoundingBox) const

  Copies bounding box data to the target array.

7.40.1 Detailed Description

A class containing data of a single mesh.

7.40.2 Member Function Documentation

7.40.2.1 CopyBoundingBox()

```
MVX2_API bool Mvx2API::MeshData::CopyBoundingBox (
            float * targetBoundingBox ) const
```

Copies bounding box data to the target array.

Parameters

targetBoundingBox	an outputted bounding box data array (must be pre-allocated with 6 elements)

Returns

true if the bounding box was successfully copied

7.40.2.2 CopyColorsColRGBA()

```
MVX2_API bool Mvx2API::MeshData::CopyColorsColRGBA (
            ColRGBAData * targetColors ) const
```

Copies vertex RGBA colors collection to the target array.

Parameters

targetColors	an outputted vertex RGBA colors array (must be pre-allocated with (colors count) elements)

Returns

true if the vertex RGBA colors were successfully copied

7.40.2.3 CopyColorsRGB()

```
MVX2_API bool Mvx2API::MeshData::CopyColorsRGB (
            uint8_t * targetColors ) const
```

Copies vertex RGB colors collection to the target array.

Parameters

targetColors	an outputted vertex RGB colors array (must be pre-allocated with (3 ∗ colors count) elements)

Returns

true if the vertex RGB colors were successfully copied

7.40.2.4 CopyIndices()

```
MVX2_API bool Mvx2API::MeshData::CopyIndices (
            uint32_t * targetIndices ) const
```

Copies vertex indices collection to the target array.

Parameters

targetIndices	an outputted vertex indices array (must be pre-allocated with (indices count) elements)

Returns

true if the vertex indices were successfully copied

7.40.2.5 CopyNormals()

```
MVX2_API bool Mvx2API::MeshData::CopyNormals (
            float * targetNormals ) const
```

Copies vertex normals collection to the target array.

Parameters

targetNormals	an outputted vertex normals array (must be pre-allocated with (3 ∗ normals count) elements)

Returns

true if the vertex normals were successfully copied

7.40.2.6 CopyNormalsVec3()

```
MVX2_API bool Mvx2API::MeshData::CopyNormalsVec3 (
            Vec3Data * targetNormals ) const
```

Copies vertex normals collection to the target array.

Parameters

targetNormals	an outputted vertex normals array (must be pre-allocated with (normals count) elements)

Returns

true if the vertex normals were successfully copied

7.40.2.7 CopyUVs()

```
MVX2_API bool Mvx2API::MeshData::CopyUVs (
            float * targetUVs ) const
```

Copies vertex UVs collection to the target array.

Parameters

targetUVs	an outputted vertex UVs array (must be pre-allocated with (2 ∗ UVs count) elements)

Returns

true if the vertex UVs were successfully copied

7.40.2.8 CopyUVsVec2()

```
MVX2_API bool Mvx2API::MeshData::CopyUVsVec2 (
            Vec2Data * targetUVs ) const
```

Copies vertex UVs collection to the target array.

Parameters

targetUVs	an outputted vertex UVs array (must be pre-allocated with (UVs count) elements)

Returns

true if the vertex UVs were successfully copied

7.40.2.9 CopyVertices()

```
MVX2_API bool Mvx2API::MeshData::CopyVertices (
            float * targetVertices ) const
```

Copies vertices collection to the target array.

Parameters

targetVertices	an outputted vertex positions array (must be pre-allocated with (3 * vertices count) elements)

Returns

true if the vertex positions were successfully copied

7.40.2.10 CopyVerticesVec3()

```
MVX2_API bool Mvx2API::MeshData::CopyVerticesVec3 (
            Vec3Data * targetVertices ) const
```

Copies vertices collection to the target array.

Parameters

targetVertices	an outputted vertex positions array (must be pre-allocated with (vertices count) elements)

Returns

true if the vertex positions were successfully copied

7.40.2.11 GetBoundingBox()

```
MVX2_API const float* Mvx2API::MeshData::GetBoundingBox ( ) const
```

A getter of the raw pointer to bounding box data.

Returns

bounding box (array of 6 values)

7.40.2.12 GetColorsRGB()

```
MVX2_API const uint8_t* Mvx2API::MeshData::GetColorsRGB ( ) const
```

A getter of the raw pointer to RGB colors collection.

Returns

mesh RGB colors

7.40.2.13 GetIndices()

```
MVX2_API const uint32_t* Mvx2API::MeshData::GetIndices ( ) const
```

A getter of the raw pointer to indices collection.

Returns

mesh indices

7.40.2.14 GetNormals()

```
MVX2_API const float* Mvx2API::MeshData::GetNormals ( ) const
```

A getter of the raw pointer to normals collection.

Returns

mesh normals

7.40.2.15 GetNumColors()

```
MVX2_API uint32_t Mvx2API::MeshData::GetNumColors ( ) const
```

A getter of the colors count.

Returns

count of mesh colors

7.40.2.16 GetNumIndices()

```
MVX2_API uint32_t Mvx2API::MeshData::GetNumIndices ( ) const
```

A getter of the indices count.

Returns

count of mesh indices

7.40.2.17 GetNumNormals()

```
MVX2_API uint32_t Mvx2API::MeshData::GetNumNormals ( ) const
```

A getter of the normals count.

Returns

count of mesh normals

7.40.2.18 GetNumUVs()

```
MVX2_API uint32_t Mvx2API::MeshData::GetNumUVs ( ) const
```

A getter of the UVs count.

Returns

count of mesh UVs

7.40.2.19 GetNumVertices()

```
MVX2_API uint32_t Mvx2API::MeshData::GetNumVertices ( ) const
```

A getter of the vertices count.

Returns

count of mesh vertices

7.40.2.20 GetUVs()

```
MVX2_API const float* Mvx2API::MeshData::GetUVs ( ) const
```

A getter of the raw pointer to UVs collection.

Returns

mesh UVs

7.40.2.21 GetVertices()

```
MVX2_API const float* Mvx2API::MeshData::GetVertices ( ) const
```

A getter of the raw pointer to vertices collection.

Returns

mesh vertices

The documentation for this class was generated from the following file:

- public/Mvx2API/data/mesh/MeshData.h

7.41 Mvx2API::MeshSplitter Class Reference

A helper class for splitting provided mesh data into multiple meshes, depending on the maximal count of vertices the resulting meshes are allowed to contain. The splitting is based on indices collection, so in case there are none, there will be no meshes in the result.

```
#include <MeshSplitter.h>
```

Inherits NonAssignable.

Public Member Functions

- MVX2_API MeshSplitter (uint32_t maxVerticesCount)
  A constructor.
- virtual MVX2_API ~MeshSplitter ()
  A destructor.
- MVX2_API void ClearResults ()
  Clears results of the previous mesh splitting.
- MVX2_API void SplitMesh (MeshData const *mesh, MeshIndicesMode indicesMode, bool includeNormals=true, bool includeColors=true, bool includeUVs=true)
  Splits a given mesh into submeshes, so each contains only given maximal count of vertices at most.
- MVX2_API uint32_t GetSplitMeshesCount () const
  A getter of split meshes count.
- MVX2_API MeshData * GetSplitMeshData (uint32_t meshIndex) const
  Returns a split submesh with a given index.

7.41.1 Detailed Description

A helper class for splitting provided mesh data into multiple meshes, depending on the maximal count of vertices the resulting meshes are allowed to contain. The splitting is based on indices collection, so in case there are none, there will be no meshes in the result.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 MeshSplitter()

```
MVX2_API Mvx2API::MeshSplitter::MeshSplitter (
            uint32_t maxVerticesCount )
```

A constructor.

Parameters

maxVerticesCount	a maximal count of vertices contained in the resulting split meshes

7.41.3 Member Function Documentation

7.41.3.1 GetSplitMeshData()

```
MVX2_API MeshData* Mvx2API::MeshSplitter::GetSplitMeshData (
            uint32_t meshIndex ) const
```

Returns a split submesh with a given index.

Parameters

meshIndex	an index of the submesh to return

Returns

a split submesh at the given index or null in case the index is out of bounds

7.41.3.2 GetSplitMeshesCount()

```
MVX2_API uint32_t Mvx2API::MeshSplitter::GetSplitMeshesCount ( ) const
```

A getter of split meshes count.

Returns

count of meshes

7.41.3.3 SplitMesh()

```
MVX2_API void Mvx2API::MeshSplitter::SplitMesh (
            MeshData const * mesh,
            MeshIndicesMode indicesMode,
            bool includeNormals = true,
            bool includeColors = true,
            bool includeUVs = true )
```

Splits a given mesh into submeshes, so each contains only given maximal count of vertices at most.

Resulting submeshes are stored in the collection.

Parameters

mesh	a mesh to split
indicesMode	an interpretation of indices collection (will be preserved in split meshes)
includeNormals	indication whether normals of the mesh shall be included in the splitting process and thus in the resulting submeshes
includeColors	indication whether colors of the mesh shall be included in the splitting process and thus in the resulting submeshes
includeUVs	indication whether texture UVs of the mesh shall be included in the splitting process and thus in the resulting submeshes

The documentation for this class was generated from the following file:

- public/Mvx2API/data/mesh/MeshSplitter.h

7.42 Mvx2API::MouseDoubleClickEvent Struct Reference

A 'mouse double-click' event.

`#include <MouseDoubleClickEvent.h>`

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API MouseDoubleClickEvent (int32_t button, int32_t x, int32_t y)
  A constructor.
- MVX2_API MouseDoubleClickEvent (MouseDoubleClickEvent const &other)
  A copy constructor.
- MVX2_API MouseDoubleClickEvent (MouseDoubleClickEvent &&other)
  A move constructor.
- virtual MVX2_API ~MouseDoubleClickEvent ()
  A destructor.

Additional Inherited Members

7.42.1 Detailed Description

A 'mouse double-click' event.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 MouseDoubleClickEvent() [1/3]

```
MVX2_API Mvx2API::MouseDoubleClickEvent::MouseDoubleClickEvent (
            int32_t button,
            int32_t x,
            int32_t y )
```

A constructor.

Parameters

button	a mouse button double-clicked
x	an x-coordinate of mouse during the event
y	an y-coordinate of mouse during the event

7.42.2.2 MouseDoubleClickEvent() [2/3]

```
MVX2_API Mvx2API::MouseDoubleClickEvent::MouseDoubleClickEvent (
            MouseDoubleClickEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.42.2.3 MouseDoubleClickEvent() [3/3]

```
MVX2_API Mvx2API::MouseDoubleClickEvent::MouseDoubleClickEvent (
            MouseDoubleClickEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/MouseDoubleClickEvent.h

7.43 Mvx2API::MouseDownEvent Struct Reference

A 'mouse down' event.

`#include <MouseDownEvent.h>`

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API MouseDownEvent (int32_t button, int32_t x, int32_t y)
  A constructor.
- MVX2_API MouseDownEvent (MouseDownEvent const &other)
  A copy constructor.
- MVX2_API MouseDownEvent (MouseDownEvent &&other)
  A move constructor.
- virtual MVX2_API ~MouseDownEvent ()
  A destructor.

Additional Inherited Members

7.43.1 Detailed Description

A 'mouse down' event.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 MouseDownEvent() [1/3]

```
MVX2_API Mvx2API::MouseDownEvent::MouseDownEvent (
            int32_t button,
            int32_t x,
            int32_t y )
```

A constructor.

Parameters

button	a mouse button pressed down
x	an x-coordinate of mouse during the event
y	an y-coordinate of mouse during the event

7.43.2.2 MouseDownEvent() [2/3]

```
MVX2_API Mvx2API::MouseDownEvent::MouseDownEvent (
            MouseDownEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.43.2.3 MouseDownEvent() [3/3]

```
MVX2_API Mvx2API::MouseDownEvent::MouseDownEvent (
            MouseDownEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/MouseDownEvent.h

7.44 Mvx2API::MouseMoveEvent Struct Reference

A 'mouse move' event.

```
#include <MouseMoveEvent.h>
```

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API MouseMoveEvent (int32_t x, int32_t y)

  A constructor.
- MVX2_API MouseMoveEvent (MouseMoveEvent const &other)

  A copy constructor.
- MVX2_API MouseMoveEvent (MouseMoveEvent &&other)

  A move constructor.
- virtual MVX2_API ~MouseMoveEvent ()

  A destructor.

Additional Inherited Members

7.44.1 Detailed Description

A 'mouse move' event.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 MouseMoveEvent() [1/3]

```
MVX2_API Mvx2API::MouseMoveEvent::MouseMoveEvent (
            int32_t x,
            int32_t y )
```

A constructor.

Parameters

x	an x-coordinate of mouse during the event
y	an y-coordinate of mouse during the event

7.44.2.2 MouseMoveEvent() [2/3]

```
MVX2_API Mvx2API::MouseMoveEvent::MouseMoveEvent (
            MouseMoveEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.44.2.3 MouseMoveEvent() [3/3]

```
MVX2_API Mvx2API::MouseMoveEvent::MouseMoveEvent (
            MouseMoveEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/MouseMoveEvent.h

7.45 Mvx2API::MouseUpEvent Struct Reference

A 'mouse up' event.

`#include <MouseUpEvent.h>`

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API MouseUpEvent (int32_t button, int32_t x, int32_t y)
  A constructor.
- MVX2_API MouseUpEvent (MouseUpEvent const &other)
  A copy constructor.
- MVX2_API MouseUpEvent (MouseUpEvent &&other)
  A move constructor.
- virtual MVX2_API ~MouseUpEvent ()
  A destructor.

Additional Inherited Members

7.45.1 Detailed Description

A 'mouse up' event.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 MouseUpEvent() [1/3]

```
MVX2_API Mvx2API::MouseUpEvent::MouseUpEvent (
            int32_t button,
            int32_t x,
            int32_t y )
```

A constructor.

Parameters

button	a mouse button released
x	an x-coordinate of mouse during the event
y	an y-coordinate of mouse during the event

7.45.2.2 MouseUpEvent() [2/3]

```
MVX2_API Mvx2API::MouseUpEvent::MouseUpEvent (
            MouseUpEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.45.2.3 MouseUpEvent() [3/3]

```
MVX2_API Mvx2API::MouseUpEvent::MouseUpEvent (
            MouseUpEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/MouseUpEvent.h

7.46 Mvx2API::MouseWheelEvent Struct Reference

A 'mouse wheel' event.

```
#include <MouseWheelEvent.h>
```

Inherits Mvx2API::InputEvent.

Public Member Functions

- MVX2_API MouseWheelEvent (float delta, int32_t x, int32_t y)

  A constructor.
- MVX2_API MouseWheelEvent (MouseWheelEvent const &other)

  A copy constructor.
- MVX2_API MouseWheelEvent (MouseWheelEvent &&other)

  A move constructor.
- virtual MVX2_API ~MouseWheelEvent ()

  A destructor.

Additional Inherited Members

7.46.1 Detailed Description

A 'mouse wheel' event.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 MouseWheelEvent() [1/3]

```
MVX2_API Mvx2API::MouseWheelEvent::MouseWheelEvent (
            float delta,
            int32_t x,
            int32_t y )
```

A constructor.

Parameters

delta	a delta value representing mouse wheel movement
x	an x-coordinate of mouse during the event
y	an y-coordinate of mouse during the event

7.46.2.2 MouseWheelEvent() [2/3]

```
MVX2_API Mvx2API::MouseWheelEvent::MouseWheelEvent (
            MouseWheelEvent const & other )
```

A copy constructor.

Parameters

other	an event to make a copy of

7.46.2.3 MouseWheelEvent() [3/3]

```
MVX2_API Mvx2API::MouseWheelEvent::MouseWheelEvent (
            MouseWheelEvent && other )
```

A move constructor.

Parameters

other	an event to move

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/events/MouseWheelEvent.h

7.47 MVX::PluginInfo Struct Reference

A plugin info data structure.

```
#include <PluginInfo.h>
```

Data Fields

- MVCommon::String pluginName
  A plugin name.
- MVCommon::String pluginVersion
  A plugin version.
- MVCommon::VersionInfo mvxVersion
  A version-stamp of the Mvx2 framework the plugin module was compiled with.
- MVCommon::String mvxVersionStr
  A version-stamp string of the Mvx2 framework the plugin module was compiled with.

7.47.1 Detailed Description

A plugin info data structure.

The documentation for this struct was generated from the following file:

- public/Mvx2/plugins/PluginInfo.h

7.48 Mvx2API::RandomAccessGraphRunner Class Reference

A random-access runner of data-processing graphs.

`#include <RandomAccessGraphRunner.h>`

Inherits Mvx2API::GraphRunner.

Public Member Functions

- MVX2_API RandomAccessGraphRunner (Graph *graph)
  A constructor.
- virtual MVX2_API ~RandomAccessGraphRunner ()
  A destructor.
- MVX2_API bool ProcessFrame (uint32_t frameID)
  Processes a frame with the given ID.
- virtual MVX2_API SourceInfo * GetSourceInfo () const override
  Retrieves source information about the currently open MVX source.

7.48.1 Detailed Description

A random-access runner of data-processing graphs.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 RandomAccessGraphRunner()

```
MVX2_API Mvx2API::RandomAccessGraphRunner::RandomAccessGraphRunner (
            Graph * graph )
```

A constructor.

Parameters

graph	a graph to create the runner for

7.48.3 Member Function Documentation

7.48.3.1 GetSourceInfo()

```
virtual MVX2_API SourceInfo* Mvx2API::RandomAccessGraphRunner::GetSourceInfo ( ) const  [override],
[virtual]
```

Retrieves source information about the currently open MVX source.

Returns

information about the current MVX source or null if no source is open

Implements Mvx2API::GraphRunner.

7.48.3.2 ProcessFrame()

```
MVX2_API bool Mvx2API::RandomAccessGraphRunner::ProcessFrame (
            uint32_t frameID )
```

Processes a frame with the given ID.

Parameters

frameID	an ID of the frame to process

Returns

true if no error occured during the processing

The documentation for this class was generated from the following file:

- public/Mvx2API/runners/RandomAccessGraphRunner.h

7.49 Mvx2API::Experimental::RendererGraphNode Class Reference

A graph node for rendering visual Mvx2 data.

`#include <RendererGraphNode.h>`

Inherits Mvx2API::SingleFilterGraphNode.

Public Member Functions

- MVX2_API RendererGraphNode (MVCommon::Guid const &rendererGuid)
  A constructor.
- virtual MVX2_API ~RendererGraphNode ()
  A destructor.
- MVX2_API void Render (int32_t width, int32_t height, bool reinit, int32_t fbo=0)
  Invokes rendering of cached data using internal rendering facility.
- MVX2_API void DestroyRenderer ()
  Destroys internal rendering facility (e.g. resources).
- MVX2_API void HandleInputEvent (InputEvent const &evt)
  Gives internal rendering facility an opportunity to handle input events and customize its behaviour.

Additional Inherited Members

7.49.1 Detailed Description

A graph node for rendering visual Mvx2 data.

The rendering algorithm of rendering filters is not executed from the pipeline processing thread - instead it is invoked manually whenever rendering is appropriate and requested from a client's code. During the pipeline execution the rendering filters only 'cache' visual data they work with.

Internally maintains a single rendering filter. The same filter is reused even when the graph node is added to multiple graphs.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 RendererGraphNode()

```
MVX2_API Mvx2API::Experimental::RendererGraphNode::RendererGraphNode (
            MVCommon::Guid const & rendererGuid )
```

A constructor.

Parameters

rendererGuid	a Guid of renderer filter to instantiate

Exceptions

std::invalid_argument	raised when a filter with the given Guid is not registered or it is not a renderer

7.49.3 Member Function Documentation

7.49.3.1 DestroyRenderer()

```
MVX2_API void Mvx2API::Experimental::RendererGraphNode::DestroyRenderer ( )
```

Destroys internal rendering facility (e.g. resources).

Exceptions

std::runtime_error	raised when internal filter does not exist yet or it is not a renderer

7.49.3.2 HandleInputEvent()

```
MVX2_API void Mvx2API::Experimental::RendererGraphNode::HandleInputEvent (
            InputEvent const & evt )
```

Gives internal rendering facility an opportunity to handle input events and customize its behaviour.

Parameters

evt	an input event

Exceptions

std::runtime_error	raised when internal filter does not exist yet or it is not a renderer

7.49.3.3 Render()

```
MVX2_API void Mvx2API::Experimental::RendererGraphNode::Render (
            int32_t width,
            int32_t height,
            bool reinit,
            int32_t fbo = 0 )
```

Invokes rendering of cached data using internal rendering facility.

Parameters

width	a width of the frame buffer object (or screen) to render into
height	a height of the frame buffer object (or screen) to render into
reinit	forces reinitialization of the internal rendering facility (e.g. resources, shaders) if it was initialized already
fbo	a frame buffer object to render into (0 to render to default buffer object)

Exceptions

std::runtime_error	raised when internal filter does not exist yet or it is not a renderer

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/RendererGraphNode.h

7.50 Mvx2API::SharedAtomPtr Class Reference

A shared smart-pointer to a stream.

`#include <SharedAtomPtr.h>`

Public Member Functions

- MVX2_API SharedAtomPtr ()
  A constructor.
- MVX2_API SharedAtomPtr (MVX::Atom *pAtom)
  A constructor.
- MVX2_API SharedAtomPtr (SharedAtomPtr const &other)
  A copy-constructor.
- MVX2_API ~SharedAtomPtr ()
  A destructor.
- MVX2_API SharedAtomPtr & operator= (SharedAtomPtr const &other)
  Makes the pointer point to a stream pointed to by the `other` pointer.
- MVX2_API SharedAtomPtr & operator= (MVX::Atom *pAtom)
  Makes the pointer point to a stream.
- MVX2_API operator bool () const
  Converts the pointer to a boolean value.
- MVX2_API MVX::Atom & operator* () const
  'Indirection' operator.
- MVX2_API MVX::Atom * operator-> () const
  'Dereference' operator.
- MVX2_API MVX::Atom * Get () const
  Returns a raw pointer to the pointed-to stream.

7.50.1 Detailed Description

A shared smart-pointer to a stream.

Allows sharing of the same stream object by multiple owners and automatically destroys stream objects when no more pointers point to them.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 SharedAtomPtr() [1/3]

```
MVX2_API Mvx2API::SharedAtomPtr::SharedAtomPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

7.50.2.2 SharedAtomPtr() [2/3]

```
MVX2_API Mvx2API::SharedAtomPtr::SharedAtomPtr (
            MVX::Atom * pAtom )
```

A constructor.

Parameters

pAtom	a stream to share a pointer to

7.50.2.3 SharedAtomPtr() [3/3]

```
MVX2_API Mvx2API::SharedAtomPtr::SharedAtomPtr (
            SharedAtomPtr const & other )
```

A copy-constructor.

Parameters

other	other pointer to share a pointed-to stream with

7.50.2.4 ~SharedAtomPtr()

```
MVX2_API Mvx2API::SharedAtomPtr::~SharedAtomPtr ( )
```

A destructor.

Destroys the pointed-to stream if this was the last pointer pointing to it.

7.50.3 Member Function Documentation

7.50.3.1 Get()

```
MVX2_API MVX::Atom* Mvx2API::SharedAtomPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to stream.

Returns

a raw pointer to the pointed-to stream

7.50.3.2 operator bool()

```
MVX2_API Mvx2API::SharedAtomPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to stream is not null

7.50.3.3 operator∗()

```
MVX2_API MVX::Atom& Mvx2API::SharedAtomPtr::operator* ( ) const
```

'Indirection' operator.

Returns

a reference to the pointed-to stream

7.50.3.4 operator->()

```
MVX2_API MVX::Atom* Mvx2API::SharedAtomPtr::operator-> ( ) const
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to stream

7.50.3.5 operator=() [1/2]

```
MVX2_API SharedAtomPtr& Mvx2API::SharedAtomPtr::operator= (
            MVX::Atom * pAtom )
```

Makes the pointer point to a stream.

Destroys previously pointed-to stream if this was the last pointer pointing to it.

Parameters

pAtom	a stream to point to

Returns

the pointer itself

7.50.3.6 operator=() [2/2]

```
MVX2_API SharedAtomPtr& Mvx2API::SharedAtomPtr::operator= (
            SharedAtomPtr const & other )
```

Makes the pointer point to a stream pointed to by the `other` pointer.

Destroys previously pointed-to stream if this was the last pointer pointing to it.

Parameters

other	other pointer to share a pointed-to stream with

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2API/frameaccess/SharedAtomPtr.h

7.51 MVX::SharedDataLayerPtr Class Reference

A shared smart-pointer to a data layer.

`#include <SharedDataLayerPtr.h>`

Public Member Functions

- MVX2_API SharedDataLayerPtr ()
  A constructor.
- MVX2_API SharedDataLayerPtr (DataLayer *pDataLayer)
  A constructor.
- MVX2_API SharedDataLayerPtr (SharedDataLayerPtr const &other)
  A copy-constructor.
- MVX2_API ~SharedDataLayerPtr ()
  A destructor.
- MVX2_API SharedDataLayerPtr & operator= (SharedDataLayerPtr const &other)
  Makes the pointer point to a data layer pointed to by the `other` pointer.
- MVX2_API SharedDataLayerPtr & operator= (DataLayer *pDataLayer)
  Makes the pointer point to a data layer.
- MVX2_API operator bool () const

Converts the pointer to a boolean value.

- MVX2_API DataLayer & operator∗ () const

  'Indirection' operator.

- MVX2_API DataLayer ∗ operator-> () const

  'Dereference' operator.

- MVX2_API MVX::DataLayer ∗ Get () const

  Returns a raw pointer to the pointed-to data layer.

7.51.1 Detailed Description

A shared smart-pointer to a data layer.

Allows sharing of the same data layer object by multiple owners and automatically destroys data layer objects when no more pointers point to them.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 SharedDataLayerPtr() [1/3]

```
MVX2_API MVX::SharedDataLayerPtr::SharedDataLayerPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

7.51.2.2 SharedDataLayerPtr() [2/3]

```
MVX2_API MVX::SharedDataLayerPtr::SharedDataLayerPtr (
            DataLayer * pDataLayer )
```

A constructor.

Parameters

pDataLayer	a data layer to share a pointer to

7.51.2.3 SharedDataLayerPtr() [3/3]

```
MVX2_API MVX::SharedDataLayerPtr::SharedDataLayerPtr (
            SharedDataLayerPtr const & other )
```

A copy-constructor.

Parameters

other	other pointer to share a pointed-to data layer with

7.51.2.4 ~SharedDataLayerPtr()

```
MVX2_API MVX::SharedDataLayerPtr::~SharedDataLayerPtr ( )
```

A destructor.

Destroys the pointed-to data layer if this was the last pointer pointing to it.

7.51.3 Member Function Documentation

7.51.3.1 Get()

```
MVX2_API MVX::DataLayer* MVX::SharedDataLayerPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to data layer.

Returns

a raw pointer to the pointed-to data layer

7.51.3.2 operator bool()

```
MVX2_API MVX::SharedDataLayerPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to data layer is not null

7.51.3.3 operator∗()

```
MVX2_API DataLayer& MVX::SharedDataLayerPtr::operator* ( ) const
```

'Indirection' operator.

Returns

a reference to the pointed-to data layer

7.51.3.4 operator->()

```
MVX2_API DataLayer* MVX::SharedDataLayerPtr::operator-> ( ) const
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to data layer

7.51.3.5 operator=() [1/2]

```
MVX2_API SharedDataLayerPtr& MVX::SharedDataLayerPtr::operator= (
            DataLayer * pDataLayer )
```

Makes the pointer point to a data layer.

Destroys previously pointed-to data layer if this was the last pointer pointing to it.

Parameters

pDataLayer	a data layer to point to

Returns

the pointer itself

7.51.3.6 operator=() [2/2]

```
MVX2_API SharedDataLayerPtr& MVX::SharedDataLayerPtr::operator= (
            SharedDataLayerPtr const & other )
```

Makes the pointer point to a data layer pointed to by the `other` pointer.

Destroys previously pointed-to data layer if this was the last pointer pointing to it.

Parameters

other	other pointer to share a pointed-to data layer with

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2/core/datalayers/SharedDataLayerPtr.h

7.52 MVX::SharedFilterPtr Class Reference

A shared smart-pointer to a filter.

`#include <SharedFilterPtr.h>`

Public Member Functions

- MVX2_API SharedFilterPtr ()
  A constructor.
- MVX2_API SharedFilterPtr (Filter *pFilter)
  A constructor.
- MVX2_API SharedFilterPtr (SharedFilterPtr const &other)
  A copy-constructor.
- MVX2_API ~SharedFilterPtr ()
  A destructor.
- MVX2_API SharedFilterPtr & operator= (SharedFilterPtr const &other)
  Makes the pointer point to a filter pointed to by the `other` pointer.
- MVX2_API SharedFilterPtr & operator= (Filter *pFilter)
  Makes the pointer point to a filter.
- MVX2_API operator bool () const
  Converts the pointer to a boolean value.
- MVX2_API Filter & operator* () const
  'Indirection' operator.
- MVX2_API Filter * operator-> () const
  'Dereference' operator.
- MVX2_API Filter * Get () const
  Returns a raw pointer to the pointed-to filter.

7.52.1 Detailed Description

A shared smart-pointer to a filter.

Allows sharing of the same filter object by multiple owners and automatically destroys filter objects when no more pointers point to them.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 SharedFilterPtr() [1/3]

```
MVX2_API MVX::SharedFilterPtr::SharedFilterPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

7.52.2.2 SharedFilterPtr() [2/3]

```
MVX2_API MVX::SharedFilterPtr::SharedFilterPtr (
            Filter * pFilter )
```

A constructor.

Parameters

pFilter	a filter to share a pointer to

7.52.2.3 SharedFilterPtr() [3/3]

```
MVX2_API MVX::SharedFilterPtr::SharedFilterPtr (
            SharedFilterPtr const & other )
```

A copy-constructor.

Parameters

other	other pointer to share a pointed-to filter with

7.52.2.4 ~SharedFilterPtr()

```
MVX2_API MVX::SharedFilterPtr::~SharedFilterPtr ( )
```

A destructor.

Destroys the pointed-to filter if this was the last pointer pointing to it.

7.52.3 Member Function Documentation

7.52.3.1 Get()

```
MVX2_API Filter* MVX::SharedFilterPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to filter.

Returns

a raw pointer to the pointed-to filter

7.52.3.2 operator bool()

```
MVX2_API MVX::SharedFilterPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to filter is not null

7.52.3.3 operator*()

```
MVX2_API Filter& MVX::SharedFilterPtr::operator* ( ) const
```

'Indirection' operator.

Returns

a reference to the pointed-to filter

7.52.3.4 operator->()

```
MVX2_API Filter* MVX::SharedFilterPtr::operator-> ( ) const
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to filter

7.52.3.5 operator=() [1/2]

```
MVX2_API SharedFilterPtr& MVX::SharedFilterPtr::operator= (
            Filter * pFilter )
```

Makes the pointer point to a filter.

Destroys previously pointed-to filter if this was the last pointer pointing to it.

Parameters

pFilter	a filter to point to

Returns

the pointer itself

7.52.3.6 operator=() [2/2]

```
MVX2_API SharedFilterPtr& MVX::SharedFilterPtr::operator= (
            SharedFilterPtr const & other )
```

Makes the pointer point to a filter pointed to by the `other` pointer.

Destroys previously pointed-to filter if this was the last pointer pointing to it.

Parameters

other	other pointer to share a pointed-to filter with

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2/core/filters/SharedFilterPtr.h

7.53 Mvx2API::SharedFilterPtr Class Reference

A shared smart-pointer to a filter.

```
#include <SharedFilterPtr.h>
```

Public Member Functions

- MVX2_API SharedFilterPtr ()
  A constructor.
- MVX2_API SharedFilterPtr (MVX::Filter *pFilter)
  A constructor.
- MVX2_API SharedFilterPtr (SharedFilterPtr const &other)
  A copy-constructor.
- MVX2_API ~SharedFilterPtr ()

A destructor.

- MVX2_API SharedFilterPtr & operator= (SharedFilterPtr const &other)

  Makes the pointer point to a filter pointed to by the `other` pointer.
- MVX2_API SharedFilterPtr & operator= (MVX::Filter *pFilter)

  Makes the pointer point to a filter.
- MVX2_API operator bool () const

  Converts the pointer to a boolean value.
- MVX2_API MVX::Filter & operator* () const

  'Indirection' operator.
- MVX2_API MVX::Filter * operator-> () const

  'Dereference' operator.
- MVX2_API MVX::Filter * Get () const

  Returns a raw pointer to the pointed-to filter.

7.53.1 Detailed Description

A shared smart-pointer to a filter.

Allows sharing of the same filter object by multiple owners and automatically destroys filter objects when no more pointers point to them.

7.53.2 Constructor & Destructor Documentation

7.53.2.1 SharedFilterPtr() [1/3]

```
MVX2_API Mvx2API::SharedFilterPtr::SharedFilterPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

7.53.2.2 SharedFilterPtr() [2/3]

```
MVX2_API Mvx2API::SharedFilterPtr::SharedFilterPtr (
            MVX::Filter * pFilter )
```

A constructor.

Parameters

pFilter	a filter to share a pointer to

7.53.2.3 SharedFilterPtr() [3/3]

```
MVX2_API Mvx2API::SharedFilterPtr::SharedFilterPtr (
            SharedFilterPtr const & other )
```

A copy-constructor.

Parameters

other	other pointer to share a pointed-to filter with

7.53.2.4 ~SharedFilterPtr()

```
MVX2_API Mvx2API::SharedFilterPtr::~SharedFilterPtr ( )
```

A destructor.

Destroys the pointed-to filter if this was the last pointer pointing to it.

7.53.3 Member Function Documentation

7.53.3.1 Get()

```
MVX2_API MVX::Filter* Mvx2API::SharedFilterPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to filter.

Returns

a raw pointer to the pointed-to filter

7.53.3.2 operator bool()

```
MVX2_API Mvx2API::SharedFilterPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to filter is not null

7.53.3.3 operator*()

```
MVX2_API MVX::Filter& Mvx2API::SharedFilterPtr::operator* ( ) const
```

'Indirection' operator.

Returns

a reference to the pointed-to filter

7.53.3.4 operator->()

```
MVX2_API MVX::Filter* Mvx2API::SharedFilterPtr::operator-> ( ) const
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to filter

7.53.3.5 operator=() [1/2]

```
MVX2_API SharedFilterPtr& Mvx2API::SharedFilterPtr::operator= (
            MVX::Filter * pFilter )
```

Makes the pointer point to a filter.

Destroys previously pointed-to filter if this was the last pointer pointing to it.

Parameters

pFilter	a filter to point to

Returns

the pointer itself

7.53.3.6 operator=() [2/2]

```
MVX2_API SharedFilterPtr& Mvx2API::SharedFilterPtr::operator= (
            SharedFilterPtr const & other )
```

Makes the pointer point to a filter pointed to by the `other` pointer.

Destroys previously pointed-to filter if this was the last pointer pointing to it.

Parameters

other	other pointer to share a pointed-to filter with

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2API/filters/SharedFilterPtr.h

7.54 MVX::SharedGraphPtr Class Reference

A shared smart-pointer to a graph.

`#include <SharedGraphPtr.h>`

Public Member Functions

- MVX2_API SharedGraphPtr ()
  A constructor.
- MVX2_API SharedGraphPtr (Graph *pGraph)
  A constructor.
- MVX2_API SharedGraphPtr (SharedGraphPtr const &other)
  A copy-constructor.
- MVX2_API ~SharedGraphPtr ()
  A destructor.
- MVX2_API SharedGraphPtr & operator= (SharedGraphPtr const &other)
  Makes the pointer point to a graph pointed to by the `other` pointer.
- MVX2_API SharedGraphPtr & operator= (Graph *pGraph)
  Makes the pointer point to a graph.
- MVX2_API operator bool () const
  Converts the pointer to a boolean value.
- MVX2_API Graph & operator* () const
  'Indirection' operator.
- MVX2_API Graph * operator-> () const
  'Dereference' operator.
- MVX2_API MVX::Graph * Get () const
  Returns a raw pointer to the pointed-to graph.

7.54.1 Detailed Description

A shared smart-pointer to a graph.

Allows sharing of the same graph object by multiple owners and automatically destroys graph objects when no more pointers point to them.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 SharedGraphPtr() [1/3]

```
MVX2_API MVX::SharedGraphPtr::SharedGraphPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

7.54.2.2 SharedGraphPtr() [2/3]

```
MVX2_API MVX::SharedGraphPtr::SharedGraphPtr (
            Graph * pGraph )
```

A constructor.

Parameters

pGraph	a graph to share a pointer to

7.54.2.3 SharedGraphPtr() [3/3]

```
MVX2_API MVX::SharedGraphPtr::SharedGraphPtr (
            SharedGraphPtr const & other )
```

A copy-constructor.

Parameters

other	other pointer to share a pointed-to graph with

7.54.2.4 ~SharedGraphPtr()

```
MVX2_API MVX::SharedGraphPtr::~SharedGraphPtr ( )
```

A destructor.

Destroys the pointed-to graph if this was the last pointer pointing to it.

7.54.3 Member Function Documentation

7.54.3.1 Get()

```
MVX2_API MVX::Graph* MVX::SharedGraphPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to graph.

Returns

a raw pointer to the pointed-to graph

7.54.3.2 operator bool()

```
MVX2_API MVX::SharedGraphPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

Returns

true in case the pointed-to graph is not null

7.54.3.3 operator*()

```
MVX2_API Graph& MVX::SharedGraphPtr::operator* ( ) const
```

'Indirection' operator.

Returns

a reference to the pointed-to graph

7.54.3.4 operator->()

```
MVX2_API Graph* MVX::SharedGraphPtr::operator-> ( ) const
```

'Dereference' operator.

Returns

a raw pointer to the pointed-to graph

7.54.3.5 operator=() [1/2]

```
MVX2_API SharedGraphPtr& MVX::SharedGraphPtr::operator= (
            Graph * pGraph )
```

Makes the pointer point to a graph.

Destroys previously pointed-to graph if this was the last pointer pointing to it.

Parameters

pGraph	a graph to point to

Returns

the pointer itself

7.54.3.6 operator=() [2/2]

```
MVX2_API SharedGraphPtr& MVX::SharedGraphPtr::operator= (
            SharedGraphPtr const & other )
```

Makes the pointer point to a graph pointed to by the `other` pointer.

Destroys previously pointed-to graph if this was the last pointer pointing to it.

Parameters

other	other pointer to share a pointed-to graph with

Returns

the pointer itself

The documentation for this class was generated from the following file:

- public/Mvx2/core/SharedGraphPtr.h

7.55 Mvx2API::SingleFilterGraphNode Class Reference

A graph node with a single custom, explicitly specified, processing filter.

`#include <SingleFilterGraphNode.h>`

Inherits Mvx2API::GraphNode.

Inherited by Mvx2API::AsyncFrameAccessGraphNode, Mvx2API::Experimental::RendererGraphNode, Mvx2API::FrameAccessGraph and Mvx2API::InjectFileDataGraphNode.

Public Types

- typedef FilterParameterNameIterator Iterator

  An alternative type name declaration for FilterParameterNameIterator.

Public Member Functions

- MVX2_API SingleFilterGraphNode (MVCommon::Guid const &filterGuid, bool singleFilterInstance=false, MVCommon::String const &filterName="")

  A constructor.
- virtual MVX2_API ~SingleFilterGraphNode ()

  A destructor.
- MVX2_API bool SetFilterParameterValue (MVCommon::String const &paramName, MVCommon::String const &value) const

  Sets a value of the filter's parameter.
- MVX2_API bool TryGetFilterParameterValue (MVCommon::String const &paramName, MVCommon::String &value) const

  Returns a value of the filter's parameter.
- MVX2_API bool RegisterParameterValueChangedListener (MVCommon::String const &paramName, IParameterValueChangedListener *pParameterValueChangedListener)

  Registers a listener for a parameter value changed event.
- MVX2_API void UnregisterParameterValueChangedListener (MVCommon::String const &paramName, IParameterValueChangedListener *pParameterValueChangedListener)

  Unregisters a listener for a parameter value changed event.
- MVX2_API void UnregisterAllParameterValueChangedListeners ()

  Unregisters all registered listeners for any parameter value changed events.
- MVX2_API FilterParameterNameIterator ParameterNamesBegin () const

  Returns an iterator to the first entry of the internal filter's parameters collection.
- MVX2_API FilterParameterNameIterator ParameterNamesEnd () const

  Returns an iterator to the last entry of the internal filter's parameters collection.
- MVX2_API bool ContainsDataProfile (MVCommon::Guid const &dataLayerGuid, MVCommon::Guid const &purposeGuid, bool checkCompressedDataLayersToo=true)

  Checks whether the internal filter (assuming it exists already) contains a data profile with a given guid on its output.
- MVX2_API DataProfileIterator DataProfilesBegin () const

  Returns an iterator to the first data profile entry of the internal filter (assuming it exists already).
- MVX2_API DataProfileIterator DataProfilesEnd () const

  Returns an iterator to the last data profile entry of the internal filter (assuming it exists already).

7.55.1 Detailed Description

A graph node with a single custom, explicitly specified, processing filter.

Allows to maintain internally a single filter reused when the graph node is added to multiple graphs, or to create a new filter every time the graph node is added to a graph.

7.55.2 Constructor & Destructor Documentation

7.55.2.1 SingleFilterGraphNode()

```
MVX2_API Mvx2API::SingleFilterGraphNode::SingleFilterGraphNode (
            MVCommon::Guid const & filterGuid,
            bool singleFilterInstance = false,
            MVCommon::String const & filterName = "" )
```

A constructor.

Parameters

filterGuid	a GUID of filter
singleFilterInstance	determines whether a single instance of the internal filter shall be created and reused, or a new instance shall be created whenever the graph node is added to a graph
filterName	a custom name of the filter

7.55.3 Member Function Documentation

7.55.3.1 ContainsDataProfile()

```
MVX2_API bool Mvx2API::SingleFilterGraphNode::ContainsDataProfile (
            MVCommon::Guid const & dataLayerGuid,
            MVCommon::Guid const & purposeGuid,
            bool checkCompressedDataLayersToo = true )
```

Checks whether the internal filter (assuming it exists already) contains a data profile with a given guid on its output.

The collection of the same filter's data profiles may vary depending on its current internal state and on the state of its preceeding filters in a graph. Data profiles are generally determined when the graph node is added to a graph via a graph builder, so enumerating them before that may result in an empty collection. Even further modifications of graph nodes after they were added to a graph may cause changes in the collection of data profiles - especially when hard parameters of the graph node or its predecessors are modified and followed by the graph reinitialization.

Parameters

dataLayerGuid	a guid of the data layer to check
purposeGuid	a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid)
checkCompressedDataLayersToo	an indication whether to check also compressed data layers

Returns

true in case the data profile (compressed and/or uncompressed data layer) is present on the output

Exceptions

std::runtime_error	raised in case the internal filter does not exist yet

7.55.3.2 DataProfilesBegin()

```
MVX2_API DataProfileIterator Mvx2API::SingleFilterGraphNode::DataProfilesBegin ( ) const
```

Returns an iterator to the first data profile entry of the internal filter (assuming it exists already).

The collection of the same filter's data profiles may vary depending on its current internal state and on the state of its preceeding filters in a graph. Data profiles are generally determined when the graph node is added to a graph via a graph builder, so enumerating them before that may result in an empty collection. Even further modifications of graph nodes after they were added to a graph may cause changes in the collection of data profiles - especially when hard parameters of the graph node or its predecessors are modified and followed by the graph reinitialization.

The returned iterator is equal to DataProfilesEnd() iterator when the filter does not provide any data profiles on its output.

Returns

an iterator

Exceptions

std::runtime_error	raised in case the internal filter does not exist yet

7.55.3.3 DataProfilesEnd()

```
MVX2_API DataProfileIterator Mvx2API::SingleFilterGraphNode::DataProfilesEnd ( ) const
```

Returns an iterator to the last data profile entry of the internal filter (assuming it exists already).

Returns

an iterator

Exceptions

std::runtime_error	raised in case the internal filter does not exist yet

7.55.3.4 ParameterNamesBegin()

```
MVX2_API FilterParameterNameIterator Mvx2API::SingleFilterGraphNode::ParameterNamesBegin ( )
const
```

Returns an iterator to the first entry of the internal filter's parameters collection.

Returns

an iterator

The collection of the same filter's parameters may vary depending on its current internal state and on the state of its preceeding filters in a graph. Filter parameters are generally created when the graph node is added to a graph via a graph builder, so enumerating them before that may result in an empty collection. Even further modifications of graph nodes after they were added to a graph may cause changes in the collection of parameters - especially when hard parameters are modified and followed by the graph reinitialization.

The returned iterator is equal to ParameterNamesEnd() iterator when the filter does not have any filter parameters.

Exceptions

std::runtime_error	raised in case the internal filter does not exist yet

7.55.3.5 ParameterNamesEnd()

```
MVX2_API FilterParameterNameIterator Mvx2API::SingleFilterGraphNode::ParameterNamesEnd ( )
const
```

Returns an iterator to the last entry of the internal filter's parameters collection.

Returns

an iterator

Exceptions

std::runtime_error	raised in case the internal filter does not exist yet

7.55.3.6 RegisterParameterValueChangedListener()

```
MVX2_API bool Mvx2API::SingleFilterGraphNode::RegisterParameterValueChangedListener (
            MVCommon::String const & paramName,
            IParameterValueChangedListener * pParameterValueChangedListener )
```

Registers a listener for a parameter value changed event.

Parameters

paramName	a name of the parameter to listen to changes of
pParameterValueChangedListener	a listener for the value change event

Returns

true if the parameter exists and the listener was successfully attached to its changes

Exceptions

std::runtime_error	raised in case the internal filter is supposed to exist already but does not

7.55.3.7 SetFilterParameterValue()

```
MVX2_API bool Mvx2API::SingleFilterGraphNode::SetFilterParameterValue (
            MVCommon::String const & paramName,
            MVCommon::String const & value ) const
```

Sets a value of the filter's parameter.

Parameters

paramName	a name of the parameter to set
value	a string representation of the value to set

Returns

true if the parameter exists and its value was set, false otherwise

Exceptions

std::runtime_error	raised in case the internal filter is supposed to exist already but does not

Parameters are set to the latest created filter in case a new filter instance is supposed to be created for each graph. Before the creation of the first filter, the parameters are cached and set when the filter is created.

7.55.3.8 TryGetFilterParameterValue()

```
MVX2_API bool Mvx2API::SingleFilterGraphNode::TryGetFilterParameterValue (
            MVCommon::String const & paramName,
            MVCommon::String & value ) const
```

Returns a value of the filter's parameter.

Parameters

paramName	a name of the parameter to get
value	a resulting parameter value after the call in case true is returned

Returns

true if the parameter exists and its value was retrieved

Exceptions

std::runtime_error	raised in case the internal filter does not exist yet

7.55.3.9 UnregisterParameterValueChangedListener()

```
MVX2_API void Mvx2API::SingleFilterGraphNode::UnregisterParameterValueChangedListener (
            MVCommon::String const & paramName,
            IParameterValueChangedListener * pParameterValueChangedListener )
```

Unregisters a listener for a parameter value changed event.

Parameters

paramName	a name of the parameter to stop listening to changes of
pParameterValueChangedListener	a listener to unregister

The documentation for this class was generated from the following file:

- public/Mvx2API/graphnodes/SingleFilterGraphNode.h

7.56 Mvx2API::SourceInfo Class Reference

An information provider about an MVX source.

```
#include <SourceInfo.h>
```

Inherits NonAssignable.

Public Types

- using Iterator = DataProfileIterator

  An alternative type name declaration for DataProfileIterator.

Public Member Functions

- virtual MVX2_API ~SourceInfo ()

  A destructor.
- MVX2_API uint32_t GetNumFrames () const

  Returns number of frames in the source.
- MVX2_API float GetFPS () const

  Returns source's framerate.
- MVX2_API bool ContainsDataLayer (MVCommon::Guid const &dataLayerGuid, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil(), bool checkCompressedDataLayersToo=true) const

  Checks whether the source contains a data layer with a given guid.
- MVX2_API Iterator DataProfilesBegin () const

  Returns an iterator to the first data profile entry of the source.
- MVX2_API Iterator DataProfilesEnd () const

  Returns an iterator to the last data profile entry of the source.

7.56.1 Detailed Description

An information provider about an MVX source.

7.56.2 Member Function Documentation

7.56.2.1 ContainsDataLayer()

```
MVX2_API bool Mvx2API::SourceInfo::ContainsDataLayer (
            MVCommon::Guid const & dataLayerGuid,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil(),
            bool checkCompressedDataLayersToo = true ) const
```

Checks whether the source contains a data layer with a given guid.

Parameters

dataLayerGuid	a guid of the data layer to check
purposeGuid	a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid)
checkCompressedDataLayersToo	an indication whether to check also compressed data layers

Returns

true in case the data layer (compressed and/or uncompressed) is present in the source

7.56.2.2 DataProfilesBegin()

```
MVX2_API Iterator Mvx2API::SourceInfo::DataProfilesBegin ( ) const
```

Returns an iterator to the first data profile entry of the source.

The returned iterator is equal to DataProfilesEnd() iterator when the source does not have any data.

Returns

an iterator

7.56.2.3 DataProfilesEnd()

```
MVX2_API Iterator Mvx2API::SourceInfo::DataProfilesEnd ( ) const
```

Returns an iterator to the last data profile entry of the source.

Returns

an iterator

7.56.2.4 GetFPS()

```
MVX2_API float Mvx2API::SourceInfo::GetFPS ( ) const
```

Returns source's framerate.

Returns

framerate

7.56.2.5 GetNumFrames()

```
MVX2_API uint32_t Mvx2API::SourceInfo::GetNumFrames ( ) const
```

Returns number of frames in the source.

Returns

frames count

The documentation for this class was generated from the following file:

- public/Mvx2API/core/SourceInfo.h

7.57 Mvx2API::Vec2Data Struct Reference

A structure containing 2D position data.

`#include <MeshDataTypes.h>`

Data Fields

- float x
  A x-coordinate.
- float y
  A y-coordinate.

7.57.1 Detailed Description

A structure containing 2D position data.

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/mesh/MeshDataTypes.h

7.58 Mvx2API::Vec3Data Struct Reference

A structure containing 3D position data.

`#include <MeshDataTypes.h>`

Data Fields

- float x
  A x-coordinate.
- float y
  A y-coordinate.
- float z
  A z-coordinate.

7.58.1 Detailed Description

A structure containing 3D position data.

The documentation for this struct was generated from the following file:

- public/Mvx2API/data/mesh/MeshDataTypes.h

Chapter 8

File Documentation

8.1 public/Mvx2/core/ActionResult.h File Reference

Enumerations

- enum MVX::ActionResult { MVX::AR_FAILURE = false, MVX::AR_SUCCESS = true }

  An enumeration of action results.

8.1.1 Enumeration Type Documentation

8.1.1.1 ActionResult

`enum MVX::ActionResult`

An enumeration of action results.

Enumerator

AR_FAILURE	A result indicating a failure.
AR_SUCCESS	A result indication a success.

8.2 public/Mvx2/core/datalayers/DataLayerCreator.h File Reference

Typedefs

- typedef DataLayer ∗(∗ MVX::DataLayerCreator) ()

  A type of data layer-creating functions.

8.3 public/Mvx2/core/datalayers/DataLayerDefinition.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include "DataLayerFactory.h"
```

Macros

- #define DATALAYER_DECL

  A declarator of a data layer with no exported symbols.
- #define DATALAYER_DECL_EXPORT(EXPORT_MACRO)

  A declarator of a data layer with explicit specification of export macro.

8.3.1 Macro Definition Documentation

8.3.1.1 DATALAYER_DECL

```
#define DATALAYER_DECL
```

Value:

```
DATALAYER_DECL_COMMON(NO_EXPORT_API)                \
  public:                                                                    \
    DECLARE_CLASS_SPECIFIC_NEW_DELETE(NO_EXPORT_API)  \
  private:
```

A declarator of a data layer with no exported symbols.

It shall be specified inside a data layer's class declaration.

8.3.1.2 DATALAYER_DECL_EXPORT

```
#define DATALAYER_DECL_EXPORT(
              EXPORT_MACRO )
```

Value:

```
DATALAYER_DECL_COMMON(EXPORT_MACRO)                \
  public:                                                                    \
    DECLARE_CLASS_SPECIFIC_NEW_DELETE(EXPORT_MACRO)   \
  private:
```

A declarator of a data layer with explicit specification of export macro.

It shall be specified inside a data layer's class declaration.

8.4 public/Mvx2/core/datalayers/DataLayerFactory.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include "DataLayerClassInfo.h"
#include "DataLayerFactoryIterator.h"
#include "DataLayerCreator.h"
#include "SharedDataLayerPtr.h"
#include "GenericSharedDataLayerPtr.h"
```

Typedefs

- typedef DataLayerFactoryIterator MVX::DataLayerFactory::Iterator

  An alternative type name declaration for DataLayerFactoryIterator.

Functions

- MVX2_API void * MVX::DataLayerFactory::RegisterDataLayerClass (MVCommon::Guid const &dataLayer↩ ClassGuid, DataLayerClassInfo const &dataLayerClassInfo, DataLayerCreator dataLayerCreator)

  Registers a data layer class to the factory.
- MVX2_API bool MVX::DataLayerFactory::TryGetDataLayerClassInfo (MVCommon::Guid const &dataLayer↩ ClassGuid, DataLayerClassInfo &dataLayerClassInfo)

  Tries to get a data layer class info registered with a given data layer class guid.
- MVX2_API DataLayerClassInfo MVX::DataLayerFactory::GetDataLayerClassInfo (MVCommon::Guid const &dataLayerClassGuid)

  Gets a data layer class info registered with a given data layer class guid.
- MVX2_API Iterator MVX::DataLayerFactory::Begin ()

  Returns an iterator to the first data layer class info of the factory.
- MVX2_API Iterator MVX::DataLayerFactory::End ()

  Returns an iterator to the last data layer class info of the factory.
- MVX2_API SharedDataLayerPtr MVX::DataLayerFactory::CreateDataLayer (MVCommon::Guid const &dataLayerClassGuid, MVCommon::Guid const &purposeGuid)

  Creates a data layer instance.
- MVX2_API SharedDataLayerPtr MVX::DataLayerFactory::CreateDataLayer (MVCommon::Guid const &dataLayerClassGuid, MVCommon::Guid const &purposeGuid, int versionOverride)

  Creates a data layer instance.
- template<class TDataLayerClass > GenericSharedDataLayerPtr< TDataLayerClass > MVX::DataLayerFactory::CreateDataLayer (MV↩ Common::Guid const &purposeGuid)

  Creates a data layer instance.
- template<class TDataLayerClass > GenericSharedDataLayerPtr< TDataLayerClass > MVX::DataLayerFactory::CreateDataLayer (MV↩ Common::Guid const &purposeGuid, int versionOverride)

  Creates a data layer instance.

8.4.1 Function Documentation

8.4.1.1 Begin()

```
MVX2_API Iterator MVX::DataLayerFactory::Begin ( )
```

Returns an iterator to the first data layer class info of the factory.

Returns

an iterator

The returned iterator is equal to End() iterator when the factory is empty.

8.4.1.2 CreateDataLayer() [1/4]

```
MVX2_API SharedDataLayerPtr MVX::DataLayerFactory::CreateDataLayer (
            MVCommon::Guid const & dataLayerClassGuid,
            MVCommon::Guid const & purposeGuid )
```

Creates a data layer instance.

Parameters

dataLayerClassGuid	a guid of a data layer class to create an instance of
purposeGuid	a purpose guid of the new data layer instance

Returns

a new data layer instance or nullptr if no data layer class with the given guid is registered

8.4.1.3 CreateDataLayer() [2/4]

```
MVX2_API SharedDataLayerPtr MVX::DataLayerFactory::CreateDataLayer (
            MVCommon::Guid const & dataLayerClassGuid,
            MVCommon::Guid const & purposeGuid,
            int versionOverride )
```

Creates a data layer instance.

Parameters

dataLayerClassGuid	a guid of a data layer class to create an instance of
purposeGuid	a purpose guid of the new data layer instance
versionOverride	a version-override of the new data layer instance (the instance will not be of the latest data layer class version)

Returns

a new data layer instance or nullptr if no data layer class with the given guid is registered

8.4.1.4 CreateDataLayer() [3/4]

```
template<class TDataLayerClass >
GenericSharedDataLayerPtr<TDataLayerClass> MVX::DataLayerFactory::CreateDataLayer (
            MVCommon::Guid const & purposeGuid )
```

Creates a data layer instance.

Template Parameters

TDataLayerClass	a data layer class to create an instance of

Parameters

purposeGuid	a purpose guid of the new data layer instance

Returns

a new data layer instance or nullptr if the data layer class is not registered in the factory

8.4.1.5 CreateDataLayer() [4/4]

```
template<class TDataLayerClass >
GenericSharedDataLayerPtr<TDataLayerClass> MVX::DataLayerFactory::CreateDataLayer (
            MVCommon::Guid const & purposeGuid,
            int versionOverride )
```

Creates a data layer instance.

Template Parameters

TDataLayerClass	a data layer class to create an instance of

Parameters

purposeGuid	a purpose guid of the new data layer instance
versionOverride	a version-override of the new data layer instance (the instance will not be of the latest data layer class version)

Returns

a new data layer instance or nullptr if the data layer class is not registered in the factory

8.4.1.6 End()

```
MVX2_API Iterator MVX::DataLayerFactory::End ( )
```

Returns an iterator to the last data layer class info of the factory.

Returns

an iterator

8.4.1.7 GetDataLayerClassInfo()

```
MVX2_API DataLayerClassInfo MVX::DataLayerFactory::GetDataLayerClassInfo (
            MVCommon::Guid const & dataLayerClassGuid )
```

Gets a data layer class info registered with a given data layer class guid.

Parameters

dataLayerClassGuid	a guid of a data layer class to get info about

Returns

the data layer class info with a given guid

Exceptions

std::invalid_argument	raised when there is no data layer class registered with the given guid

8.4.1.8 RegisterDataLayerClass()

```
MVX2_API void* MVX::DataLayerFactory::RegisterDataLayerClass (
            MVCommon::Guid const & dataLayerClassGuid,
            DataLayerClassInfo const & dataLayerClassInfo,
            DataLayerCreator dataLayerCreator )
```

Registers a data layer class to the factory.

Parameters

dataLayerClassGuid	a guid of the data layer class
dataLayerClassInfo	information about the data layer class
dataLayerCreator	a creator of the data layer instances

Returns

always nullptr

8.4.1.9 TryGetDataLayerClassInfo()

```
MVX2_API bool MVX::DataLayerFactory::TryGetDataLayerClassInfo (
            MVCommon::Guid const & dataLayerClassGuid,
            DataLayerClassInfo & dataLayerClassInfo )
```

Tries to get a data layer class info registered with a given data layer class guid.

Parameters

dataLayerClassGuid	a guid of a data layer class to get info about
dataLayerClassInfo	a target to store the data layer class info to

Returns

true in case there is a data layer class info registered for the guid

8.5 public/Mvx2/core/datalayers/DataLayerFactoryIterator.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/Memory.h>
#include <MVCommon/utils/Pair.h>
#include <MVCommon/guid/Guid.h>
#include "DataLayerClassInfo.h"
```

Data Structures

- class MVX::DataLayerFactoryIterator

  An iterator over elements of DataLayerFactory collection.

8.6 public/Mvx2/core/filters/FilterCategory.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/utils/String.h>
#include <type_traits>
```

Enumerations

- enum MVX::FilterCategory {
  MVX::FC_UNKNOWN, MVX::FC_SOURCE, MVX::FC_TRANSFORM, MVX::FC_TRANSFORM_TEXTURECONVERSION,
  MVX::FC_TRANSFORM_TEXTURECOLOR, MVX::FC_TARGET, MVX::FC_RENDERER, MVX::FC_TRANSFORM_COMPRE
  MVX::FC_TRANSFORM_DECOMPRESSOR }

  An enumeration of filter categories.

Functions

- MVX2_API MVCommon::String MVX::FilterCategoryDeterminer::GetFilterCategoryName (FilterCategory filterCategory)

  Converts a filter category to a category name.
- template<class TFilterClass >
  FilterCategory MVX::FilterCategoryDeterminer::DetermineFilterCategory ()

  Determines category of a filter class.

8.6.1 Enumeration Type Documentation

8.6.1.1 FilterCategory

```
enum MVX::FilterCategory
```

An enumeration of filter categories.

Enumerator

FC_UNKNOWN	Unknown category indication.
FC_SOURCE	A category of Source filters.
FC_TRANSFORM	A category of Transform filters.
FC_TRANSFORM_TEXTURECONVERSION	A category of texture-converting Transform filters.
FC_TRANSFORM_TEXTURECOLOR	A category of texture-color Transform filters.
FC_TARGET	A category of Target filters.
FC_RENDERER	A category of rendering Target (Renderer) filters.
FC_TRANSFORM_COMPRESSOR	A category of data-compressing Transform filters.
FC_TRANSFORM_DECOMPRESSOR	A category of data-decompressing Transform filters.

8.6.2 Function Documentation

8.6.2.1 DetermineFilterCategory()

```
template<class TFilterClass >
FilterCategory MVX::FilterCategoryDeterminer::DetermineFilterCategory ( )
```

Determines category of a filter class.

Template Parameters

TFilterClass	a filter class to determine the category of

Category of a filter class is determined based on its class inheritance.

Returns

category of the filter class

8.6.2.2 GetFilterCategoryName()

```
MVX2_API MVCommon::String MVX::FilterCategoryDeterminer::GetFilterCategoryName (
            FilterCategory filterCategory )
```

Converts a filter category to a category name.

Parameters

filterCategory	a filter category to get the name of

Returns

category's name

8.7 public/Mvx2/core/filters/FilterCreator.h File Reference

Typedefs

- typedef Filter ∗(∗ MVX::FilterCreator) ()

  A type of filter-creating functions.

8.8 public/Mvx2/core/filters/FilterDefinition.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include "FilterFactory.h"
```

Macros

- #define FILTER_DECL(BASECLASSNAME)

  A declarator of a filter with no exported symbols.
- #define FILTER_DECL_EXPORT(BASECLASSNAME, EXPORT_MACRO)

  A declarator of a filter with explicit specification of export macro.

8.8.1 Macro Definition Documentation

8.8.1.1 FILTER_DECL

```
#define FILTER_DECL(
              BASECLASSNAME )
```

Value:

```
FILTER_DECL_COMMON(BASECLASSNAME, NO_EXPORT_API)     \
  public:                                                                    \
    DECLARE_CLASS_SPECIFIC_NEW_DELETE(NO_EXPORT_API)   \
  private:
```

A declarator of a filter with no exported symbols.

It shall be specified inside a filter's class declaration.

8.8.1.2 FILTER_DECL_EXPORT

```
#define FILTER_DECL_EXPORT(
              BASECLASSNAME,
              EXPORT_MACRO )
```

Value:

```
FILTER_DECL_COMMON(BASECLASSNAME, EXPORT_MACRO)         \
  public:                                                                    \
    DECLARE_CLASS_SPECIFIC_NEW_DELETE(EXPORT_MACRO)       \
  private:
```

A declarator of a filter with explicit specification of export macro.

It shall be specified inside a filter's class declaration.

8.9 public/Mvx2/core/filters/FilterFactory.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include "FilterClassInfo.h"
#include "FilterFactoryIterator.h"
#include "SharedFilterPtr.h"
#include "GenericSharedFilterPtr.h"
#include <Mvx2/core/SharedGraphPtr.h>
#include "FilterCreator.h"
```

Typedefs

- typedef FilterFactoryIterator MVX::FilterFactory::Iterator

  An alternative type name declaration for FilterFactoryIterator.

Functions

- MVX2_API void * MVX::FilterFactory::RegisterFilterClass (MVCommon::Guid const &filterClassGuid, FilterClassInfo const &filterClassInfo, FilterCreator filterCreator)

  Registers a filter class to the factory.
- MVX2_API bool MVX::FilterFactory::TryGetFilterClassInfo (MVCommon::Guid const &filterClassGuid, FilterClassInfo &filterClassInfo)

  Tries to get a filter class info registered with a given filter class guid.
- MVX2_API FilterClassInfo MVX::FilterFactory::GetFilterClassInfo (MVCommon::Guid const &filterClassGuid)

  Gets a filter class info registered with a given filter class guid.
- MVX2_API Iterator MVX::FilterFactory::Begin ()

  Returns an iterator to the first filter class info of the factory.
- MVX2_API Iterator MVX::FilterFactory::End ()

  Returns an iterator to the last filter class info of the factory.
- MVX2_API SharedFilterPtr MVX::FilterFactory::CreateFilter (MVCommon::Guid const &filterClassGuid, SharedFilterPtr spInputFilter=(Filter *) nullptr, SharedGraphPtr spGraph=(Graph *) nullptr)

  Creates a filter instance.
- template<class TFilterClass > GenericSharedFilterPtr< TFilterClass > MVX::FilterFactory::CreateFilter (SharedFilterPtr spInputFilter=(Filter *) nullptr, SharedGraphPtr spGraph=(Graph *) nullptr)

  Creates a filter instance.

8.9.1 Function Documentation

8.9.1.1 Begin()

```
MVX2_API Iterator MVX::FilterFactory::Begin ( )
```

Returns an iterator to the first filter class info of the factory.

Returns

an iterator

The returned iterator is equal to End() iterator when the factory is empty.

8.9.1.2 CreateFilter() [1/2]

```
MVX2_API SharedFilterPtr MVX::FilterFactory::CreateFilter (
            MVCommon::Guid const & filterClassGuid,
            SharedFilterPtr spInputFilter = (Filter *) nullptr,
            SharedGraphPtr spGraph = (Graph *) nullptr )
```

Creates a filter instance.

Parameters

filterClassGuid	a guid of a filter class to create an instance of
spInputFilter	a filter to initialize the new filter instance with (in a role of its input filter)
spGraph	a graph to initialize the new filter with

Returns

a new filter instance or nullptr if no filter class with the given guid is registered

8.9.1.3 CreateFilter() [2/2]

```
template<class TFilterClass >
GenericSharedFilterPtr<TFilterClass> MVX::FilterFactory::CreateFilter (
            SharedFilterPtr spInputFilter = (Filter *) nullptr,
            SharedGraphPtr spGraph = (Graph *) nullptr )
```

Creates a filter instance.

Template Parameters

TFilterClass	a filter class to create an instance of

Parameters

spInputFilter	a filter to initialize the new filter instance with (in a role of its input filter)
spGraph	a graph to initialize the new filter with

Returns

a new filter instance or nullptr if the filter class is not registered in the factory

8.9.1.4 End()

```
MVX2_API Iterator MVX::FilterFactory::End ( )
```

Returns an iterator to the last filter class info of the factory.

Returns

an iterator

8.9.1.5 GetFilterClassInfo()

```
MVX2_API FilterClassInfo MVX::FilterFactory::GetFilterClassInfo (
            MVCommon::Guid const & filterClassGuid )
```

Gets a filter class info registered with a given filter class guid.

Parameters

filterClassGuid	a guid of a filter class to get info about

Returns

the filter class info with a given guid

Exceptions

std::invalid_argument	raised when there is no filter class registered with the given guid

8.9.1.6 RegisterFilterClass()

```
MVX2_API void* MVX::FilterFactory::RegisterFilterClass (
```

```
            MVCommon::Guid const & filterClassGuid,
            FilterClassInfo const & filterClassInfo,
            FilterCreator filterCreator )
```

Registers a filter class to the factory.

Parameters

filterClassGuid	a guid of the filter class
filterClassInfo	information about the filter class
filterCreator	a creator of the filter instances

Returns

always nullptr

8.9.1.7 TryGetFilterClassInfo()

```
MVX2_API bool MVX::FilterFactory::TryGetFilterClassInfo (
            MVCommon::Guid const & filterClassGuid,
            FilterClassInfo & filterClassInfo )
```

Tries to get a filter class info registered with a given filter class guid.

Parameters

filterClassGuid	a guid of a filter class to get info about
filterClassInfo	a target to store the filter class info to

Returns

true in case there is a filter class info registered for the guid

8.10 public/Mvx2/core/filters/FilterFactoryIterator.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/Memory.h>
#include <MVCommon/utils/Pair.h>
#include <MVCommon/guid/Guid.h>
#include "FilterClassInfo.h"
```

Data Structures

- class MVX::FilterFactoryIterator

  An iterator over elements of FilterFactory collection.

8.11 public/Mvx2/core/MvxVersion.h File Reference

```
#include <MVCommon/utils/VersionInfo.h>
```

Macros

- #define MVX_VERSION_MAJOR 6

  Current value of the most-significant Mvx2 framework version component.
- #define MVX_VERSION_MINOR 3

  Current value of the medium-significant Mvx2 framework version component.
- #define MVX_VERSION_PATCH 1

  Current value of the least-significant Mvx2 framework version component.

Variables

- const MVCommon::VersionInfo MVX::MVX_COMPILE_VERSION = { 6 , 3 , 1 }

  A version of Mvx2 framework at compilation time of depending modules (e.g. plugins).
- const MVCommon::VersionInfo MVX::MVX_RUNTIME_VERSION

  A version of Mvx2 framework at runtime.

8.11.1 Variable Documentation

8.11.1.1 MVX_RUNTIME_VERSION

```
const MVCommon::VersionInfo MVX::MVX_RUNTIME_VERSION
```

A version of Mvx2 framework at runtime.

The version is the version compiled into the Mvx2 framework itself.

8.12 public/Mvx2/plugins/PluginDatabase.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/utils/String.h>
```

Functions

- MVX2_API void MVX::PluginDatabase::ScanFolderForPlugins (MVCommon::String const &pluginsFolder↩ Path, bool checkCacheFile=false, bool storeCacheFile=false, bool checkSubfolders=true)

  Scans and tries to load plugins in a given folder.
- MVX2_API bool MVX::PluginDatabase::AddPlugin (MVCommon::String const &pluginPath, MVCommon::↩ String *pFailReason=nullptr)

  Tries to load a plugin at a given path.
- MVX2_API void MVX::PluginDatabase::LoadPluginsFromCacheFile (MVCommon::String const &cacheFile↩ Path)

  Loads a plugins cache file.
- MVX2_API void MVX::PluginDatabase::SavePluginsToCacheFile (MVCommon::String const &cacheFile↩ Path)

  Stores the plugin database to a plugins cache file.

8.12.1 Function Documentation

8.12.1.1 AddPlugin()

```
MVX2_API bool MVX::PluginDatabase::AddPlugin (
            MVCommon::String const & pluginPath,
            MVCommon::String * pFailReason = nullptr )
```

Tries to load a plugin at a given path.

Parameters

pluginPath	a path of the plugin
pFailReason	an optional holder for failure reason message (when nullptr passed, no failure message is provided)

Returns

true in case the plugin was successfully loaded (if actual loading does not take place immediatelly because of existing cache, true is returned as well even if the plugin would not be successfully loadable)

8.12.1.2 LoadPluginsFromCacheFile()

```
MVX2_API void MVX::PluginDatabase::LoadPluginsFromCacheFile (
            MVCommon::String const & cacheFilePath )
```

Loads a plugins cache file.

Populates the plugins database with records about plugins stored in the cache file.

Parameters

cacheFilePath	a path of the cache file

8.12.1.3 SavePluginsToCacheFile()

```
MVX2_API void MVX::PluginDatabase::SavePluginsToCacheFile (
            MVCommon::String const & cacheFilePath )
```

Stores the plugin database to a plugins cache file.

Parameters

cacheFilePath	a desired path of the cache file

8.12.1.4 ScanFolderForPlugins()

```
MVX2_API void MVX::PluginDatabase::ScanFolderForPlugins (
            MVCommon::String const & pluginsFolderPath,
            bool checkCacheFile = false,
            bool storeCacheFile = false,
            bool checkSubfolders = true )
```

Scans and tries to load plugins in a given folder.

When plugins cache file is checked before scanning, the plugins which are cached in 'plugins.xml' file are not loaded unless the actual files of the plugins have a newer timestamp.

Only files with '_mvp.{extension}' names (e.g. '_mvp.dll') are scanned as potential plugin files.

Parameters

pluginsFolderPath	a folder to scan for plugins
checkCacheFile	if true, before actually scanning tries to locate and read plugins cache file in the folder
storeCacheFile	if true, after scanning stores plugins cache file inside the folder
checkSubfolders	if true, checks also subfolders of the folder

8.13 public/Mvx2/plugins/PluginInfo.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <Mvx2/core/MvxVersion.h>
#include <MVCommon/CUtil.h>
#include <MVCommon/utils/String.h>
```

Data Structures

- struct MVX::PluginInfo

  A plugin info data structure.

Macros

- #define MVX_PLUGIN_INFO_OBJECT_NAME mvx_plugin

  A name of the symbol that indicates a module is Mvx2 framework's plugin.
- #define MVX_PLUGIN_INFO_OBJECT_NAME_STR "mvx_plugin"

  A string literal containing name of the symbol that indicates a module is Mvx2 framework's plugin.
- #define MVX_PLUGIN_API

  Defines export macro for plugin-exported symbols.
- #define MVX_PLUGIN(pluginName, pluginVersion)

  Defines a symbol that is expected in all Mvx2 framework's plugin modules.

8.13.1 Macro Definition Documentation

8.13.1.1 MVX_PLUGIN

```
#define MVX_PLUGIN(
              pluginName,
              pluginVersion )
```

Value:

```
extern "C"
      \
    {
      \
      MVX_PLUGIN_API MVX::PluginInfo MVX_PLUGIN_INFO_OBJECT_NAME =
      \
      {
      \
        pluginName,
      \
        pluginVersion,
      \
        MVX::MVX_COMPILE_VERSION,
      \
        "mvxversion_" MV_CONSTANT_TO_STR(MVX_VERSION_MAJOR) "." MV_CONSTANT_TO_STR(MVX_VERSION_MINOR) "."
      MV_CONSTANT_TO_STR(MVX_VERSION_PATCH),  \
      };
      \
    }
```

Defines a symbol that is expected in all Mvx2 framework's plugin modules.

Without the symbol the module can not be loaded by framework as its plugin.

8.14 public/Mvx2/utils/Logger.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/logger/WeakLoggerPtr.h>
```

Data Structures

- class MVX::IMVXLoggerInstanceListener

  An interface of listeners to MVX logger instance changes.

Macros

- #define MVX2_TAG "MVX2"

  A tag used in log messages originating from Mvx2 framework.
- #define MVX2_SIMPLE_TAG "MVX2_SIMPLE"

  An alternative tag used in log messages originating from Mvx2 framework.
- #define MVX2_LOG_ERROR(fmt, ...) LOGGER_WP_LOG_E(MVX::GetMVXLoggerInstance(), MVX2_TAG, fmt, ##__VA_ARGS__)

  Logs an error message.
- #define MVX2_LOG_INFO(fmt, ...) LOGGER_WP_LOG_I(MVX::GetMVXLoggerInstance(), MVX2_TAG, fmt, ##__VA_ARGS__)

  Logs an info message.
- #define MVX2_LOG_WARN(fmt, ...) LOGGER_WP_LOG_W(MVX::GetMVXLoggerInstance(), MVX2_TAG, fmt, ##__VA_ARGS__)

  Logs a warning message.
- #define MVX2_LOG_VERB(fmt, ...) LOGGER_WP_LOG_V(MVX::GetMVXLoggerInstance(), MVX2_TAG, fmt, ##__VA_ARGS__)

  Logs a verbose message.
- #define MVX2_SIMPLE_LOG(fmt, ...) LOGGER_WP_LOG_V(MVX::GetMVXLoggerInstance(), MVX2_SIMPLE_TAG, fmt, ##__VA_ARGS__)

  Logs a verbose message with the alternative (MVX2_SIMPLE_TAG) tag.
- #define MVX2_LOG_DEBUG(fmt, ...)

  Logs a debug message in release configurations and nothing in debug configurations.

Functions

- MVX2_API void MVX::SetMVXLoggerInstance (MVCommon::WeakLoggerPtr wpLogger)

  Sets the MVX logger instance.
- MVX2_API void MVX::ResetMVXLoggerInstance ()

  Resets the MVX logger instance.
- MVX2_API MVCommon::WeakLoggerPtr MVX::GetMVXLoggerInstance ()

  Accesses current MVX logger instance.
- MVX2_API void MVX::RegisterMVXLoggerInstanceListener (IMVXLoggerInstanceListener *pListener)

  Registers a listener to MVX logger instance changes.
- MVX2_API void MVX::UnregisterMVXLoggerInstanceListener (IMVXLoggerInstanceListener *pListener)

  Unregisters a listener to MVX logger instance changes.

8.14.1 Function Documentation

8.14.1.1 GetMVXLoggerInstance()

```
MVX2_API MVCommon::WeakLoggerPtr MVX::GetMVXLoggerInstance ( )
```

Accesses current MVX logger instance.

Returns

weak pointer to the current MVX logger instance

8.14.1.2 RegisterMVXLoggerInstanceListener()

```
MVX2_API void MVX::RegisterMVXLoggerInstanceListener (
            IMVXLoggerInstanceListener * pListener )
```

Registers a listener to MVX logger instance changes.

Parameters

pListener	a listener to register

8.14.1.3 ResetMVXLoggerInstance()

```
MVX2_API void MVX::ResetMVXLoggerInstance ( )
```

Resets the MVX logger instance.

No logs will be generated.

8.14.1.4 SetMVXLoggerInstance()

```
MVX2_API void MVX::SetMVXLoggerInstance (
            MVCommon::WeakLoggerPtr wpLogger )
```

Sets the MVX logger instance.

Replaces the previous logger instance if there was any.

Parameters

wpLogger	a weak pointer to the new logger instance

8.14.1.5 UnregisterMVXLoggerInstanceListener()

```
MVX2_API void MVX::UnregisterMVXLoggerInstanceListener (
            IMVXLoggerInstanceListener * pListener )
```

Unregisters a listener to MVX logger instance changes.

Parameters

pListener	a listener to unregister

8.15 public/Mvx2/utils/MVXPurposeGuids.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/Guid.h>
#include <MVCommon/guid/SharedGuidAliasDatabasePtr.h>
```

Macros

- #define DECLARE_PURPOSE_GUID(alias, guid) static const MVCommon::Guid PurposeGuid_ ## alias = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(guid, #alias);

  Declares a purpose guid alias and registers it to the Mvx2 framework's internal database.

Functions

- MVX2_API MVCommon::Guid MVX::MVXPurposeGuids::RegisterPurposeGuidAlias (MVCommon::Guid const &guid, MVCommon::String const &alias)

  Registers a purpose guid alias in Mvx2 framework's internal database of guid aliases.

Variables

- static const MVCommon::Guid MVX::PurposeGuid_MV4D = MVX::MVXPurposeGuids::RegisterPurpose↩ GuidAlias(MVCommon::Guid::FromHexString("6F0022F4-A3D5-4CCA-8DAC-ADCC3B37C839") , "MV4D")

  MV4D purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_AFFINE = MVX::MVXPurposeGuids::RegisterPurpose↩ GuidAlias(MVCommon::Guid::FromHexString("CAAFFE19-6E28-48F8-966B-480C70D43C67") , "AFFINE")

  AFFINE purpose guid.

- static const MVCommon::Guid MVX::PurposeGuid_PHOTOGRAMMETRY = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("44AA3F2B-BD87-4B70-8BF7-7EF34BB97EF3") , "PHOTOGRAMMETRY")

  PHOTOGRAMMETRY purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_MINC = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("EE290172-596C-4A9A-9175-AC029038B812") , "MINC")

  MINC purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_MVX1 = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("F07C7A27-7417-4638-BA50-08C68942C112") , "MVX1")

  MVX1 purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_Network = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("3DF13A4F-31E9-4756-AE01-FFA8E1EB80A5") , "Network")

  Network purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_ASD = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("8A213444-B140-4EA0-B625-175865657949") , "ASD")

  ASD purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_CROP = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("2483D517-8097-4263-8F06-02468EBC0443") , "CROP")

  CROP purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_DUMMY1 = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("B25B7743-920F-4BB0-A650-D1087E87BB6F") , "DUMMY1")

  DUMMY1 purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_DUMMY2 = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("5F0A4F1B-F04B-481B-A97C-F23E6C2071A0") , "DUMMY2")

  DUMMY2 purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_DUMMY3 = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("EC67A072-E33B-4862-A39E-4278A539E810") , "DUMMY3")

  DUMMY3 purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_STATIC = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("80FE49AE-24BF-439E-B8D5-383E83D59D79") , "STATIC")

  STATIC purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_DYNAMIC = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("AC41E48E-3714-4809-AB28-2A2A8E9AE06F") , "DYNAMIC")

  DYNAMIC purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_REX = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("3C766FE2-02AB-4D55-A9F9-5C4396C287F8") , "REX")

  REX purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_UNDISTORTED_COLORSPACE = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("29D2DF13-AAC7-4927-943E-17AB2FACC390") , "UNDISTORTED_COLORSPACE")

  UNDISTORTED_COLORSPACE purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_MERGED_POISSON = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("AC29C86C-6729-456A-ADF5-DF1D14804716") , "MERGED_POISSON")

  MERGED_POISSON purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_MERGED_TSDF = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("A9065DBA-7A9E-4A2C-82F9-56D0872DD102") , "MERGED_TSDF")

  MERGED_TSDF purpose guid.

- static const MVCommon::Guid MVX::PurposeGuid_SEQUENTIAL_POISSON = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("77109EC1-8B96-4DD8-9035-19DF0C626F2D") , "SEQUENTIAL_POISSON")

  SEQUENTIAL_POISSON purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_DECIMATE = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("1DD9CC73-8C94-4A0F-A610-687E31A682E2") , "DECIMATE")

  DECIMATE purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_COLOR_QUANT = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("CABA6F7C-E11B-495E-BA47-6B5AF2F3500C") , "COLOR_QUANT")

  COLOR_QUANT purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_SEQUENTIAL_REGISTRATION = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("58DDE7D7-4E76-4775-9C6F-8D6EBA25CADC") , "SEQUENTIAL_REGISTRATION")

  SEQUENTIAL_REGISTRATION purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_DENOISE = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("F0C6FB6D-A803-45E9-8F76-539E0F9152DE") , "DENOISE")

  DENOISE purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_THUMBNAIL = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("64645F07-91BB-4DAB-A7AD-1B50F1C1D1F0") , "THUMBNAIL")

  THUMBNAIL purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_COLORCORRECTED = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("61EAF822-27C4-43ED-A8D7-CCB0EC14FFCC") , "COLORCORRECTED")

  COLORCORRECTED purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_CIRCLES = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("617B0F33-3DA0-46B5-B3F0-ECD44A848B1F") , "CIRCLES")

  CIRCLES purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_CAMERA_PRIMARY = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("29D21270-8E73-4284-A3F6-5A9D868C9D1B") , "CAMERA_PRIMARY")

  CAMERA_PRIMARY purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_CAMERA_SECONDARY = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("EA97C8D8-CFBC-4052-8223-AAE4DF2EC903") , "CAMERA_SECONDARY")

  CAMERA_SECONDARY purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_PRIMARY_IR = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("703A5C10-16D4-419B-956B-DD1A894D5E0C") , "PRIMARY_IR")

  PRIMARY_IR purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_PRIMARY_COLOR = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("947E9749-2931-4A11-B747-7FC8EE1BD887") , "PRIMARY_COLOR")

  PRIMARY_COLOR purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_SECONDARY_COLOR = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("188F7F1F-3264-4073-B166-6C1A3DE002A2") , "SECONDARY_COLOR")

  SECONDARY_COLOR purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_MULTIPATCH_COLOR = MVX::MVXPurposeGuids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("FE75E4E4-4DEF-43BF-97FD-6A0A4D39C08E") , "MULTIPATCH_COLOR")

MULTIPATCH_COLOR purpose guid.

- static const MVCommon::Guid MVX::PurposeGuid_MULTIPATCH_DEPTH = MVX::MVXPurposeGuids↩ ::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("0EEED5E0-3D99-494F-A3E6-F46880↩ A99116") , "MULTIPATCH_DEPTH")

  MULTIPATCH_DEPTH purpose guid.
- static const MVCommon::Guid MVX::PurposeGuid_MULTIPATCH_COMBINED = MVX::MVXPurpose↩ Guids::RegisterPurposeGuidAlias(MVCommon::Guid::FromHexString("0F7A4F52-D13A-4B77-B7A2-23636E4DB3F9") , "MULTIPATCH_COMBINED")

  MULTIPATCH_COMBINED purpose guid.

8.15.1 Function Documentation

8.15.1.1 RegisterPurposeGuidAlias()

```
MVX2_API MVCommon::Guid MVX::MVXPurposeGuids::RegisterPurposeGuidAlias (
            MVCommon::Guid const & guid,
            MVCommon::String const & alias )
```

Registers a purpose guid alias in Mvx2 framework's internal database of guid aliases.

Parameters

guid	a purpose guid
alias	a purpose guid alias

Returns

the same purpose guid

8.16 public/Mvx2/utils/Utils.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/SharedGuidAliasDatabasePtr.h>
```

Functions

- MVX2_API MVCommon::SharedGuidAliasDatabasePtr MVX::Utils::GetMVXGuidAliasDatabase ()

  Gets Mvx2 framework's internal database of guid-alias pairs.
- MVX2_API MVCommon::String MVX::Utils::GetGuidAlias (MVCommon::Guid const &guid)

  Retrieves a guid alias from Mvx2 framework's internal database of guid-alias pairs, or transforms the guid into its hexadecimal representation if there is no alias registered.

8.16.1 Function Documentation

8.16.1.1 GetGuidAlias()

```
MVX2_API MVCommon::String MVX::Utils::GetGuidAlias (
            MVCommon::Guid const & guid )
```

Retrieves a guid alias from Mvx2 framework's internal database of guid-alias pairs, or transforms the guid into its hexadecimal representation if there is no alias registered.

Parameters

guid	a guid

Returns

guid alias or the guid's hexadecimal representation

8.16.1.2 GetMVXGuidAliasDatabase()

```
MVX2_API MVCommon::SharedGuidAliasDatabasePtr MVX::Utils::GetMVXGuidAliasDatabase ( )
```

Gets Mvx2 framework's internal database of guid-alias pairs.

Returns

guid-alias database

8.17 public/Mvx2API/utils/Utils.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/utils/String.h>
#include <MVCommon/logger/WeakLoggerPtr.h>
#include <MVCommon/guid/SharedGuidAliasDatabasePtr.h>
```

Functions

- MVX2_API void Mvx2API::Utils::SetMVXLoggerInstance (MVCommon::WeakLoggerPtr wpLogger)
  Sets the MVX logger instance.
- MVX2_API void Mvx2API::Utils::ResetMVXLoggerInstance ()
  Resets the MVX logger instance.
- MVX2_API MVCommon::WeakLoggerPtr Mvx2API::Utils::GetMVXLoggerInstance ()
  Accesses current MVX logger instance.
- MVX2_API MVCommon::SharedGuidAliasDatabasePtr Mvx2API::Utils::GetMVXGuidAliasDatabase ()
  A getter of the database containing MVX2 framework's internal guids and their aliases.
- MVX2_API MVCommon::String Mvx2API::Utils::GetAppExeFilePath ()
  Returns path of the application's executable file.
- MVX2_API MVCommon::String Mvx2API::Utils::GetAppExeDirectory ()
  Returns directory of the application's executable file.

8.17.1 Function Documentation

8.17.1.1 GetAppExeDirectory()

```
MVX2_API MVCommon::String Mvx2API::Utils::GetAppExeDirectory ( )
```

Returns directory of the application's executable file.

Returns

executable directory

8.17.1.2 GetAppExeFilePath()

```
MVX2_API MVCommon::String Mvx2API::Utils::GetAppExeFilePath ( )
```

Returns path of the application's executable file.

Returns

executable file path

8.17.1.3 GetMVXGuidAliasDatabase()

```
MVX2_API MVCommon::SharedGuidAliasDatabasePtr Mvx2API::Utils::GetMVXGuidAliasDatabase ( )
```

A getter of the database containing MVX2 framework's internal guids and their aliases.

Returns

a database of guids and their aliases

8.17.1.4 GetMVXLoggerInstance()

```
MVX2_API MVCommon::WeakLoggerPtr Mvx2API::Utils::GetMVXLoggerInstance ( )
```

Accesses current MVX logger instance.

Returns

weak pointer to the current MVX logger instance

8.17.1.5 ResetMVXLoggerInstance()

```
MVX2_API void Mvx2API::Utils::ResetMVXLoggerInstance ( )
```

Resets the MVX logger instance.

No logs will be generated.

8.17.1.6 SetMVXLoggerInstance()

```
MVX2_API void Mvx2API::Utils::SetMVXLoggerInstance (
            MVCommon::WeakLoggerPtr wpLogger )
```

Sets the MVX logger instance.

Replaces the previous logger instance if there was any.

There is no logger instance set by default - it is a responsibility of the application to install one.

Parameters

wpLogger	a weak pointer to the new logger instance

8.18 public/Mvx2API/data/BasicDataLayersGuids.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/Guid.h>
```

Functions

- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::AUDIO_DATA_LAYER ()
  A getter of audio data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_POSITIONS_DATA_LAYER ()
  A getter of vertex positions data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_COLORS_DATA_LAYER ()
  A getter of vertex colors data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_NORMALS_DATA_LAYER ()
  A getter of vertex normals data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_UVS_DATA_LAYER ()
  A getter of vertex UVs data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_INDICES_DATA_LAYER ()
  A getter of vertex indices data layer GUID.

- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::CAMERA_PARAMS_DATA_LAYER ()

  A getter of camera params data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::TRANSFORM_DATA_LAYER ()

  A getter of transform data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::SEGMENT_INFO_DATA_LAYER ()

  A getter of segment info data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::BYTEARRAY_DATA_LAYER ()

  A getter of bytearray data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::DEPTHMAP_TEXTURE_DATA_LAYER ()

  A getter of depth map texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::IR_TEXTURE_DATA_LAYER ()

  A getter of IR texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::RGB_TEXTURE_DATA_LAYER ()

  A getter of RGB texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::NVX_TEXTURE_DATA_LAYER ()

  A getter of NVX texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::NV12_TEXTURE_DATA_LAYER ()

  A getter of NV12 texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::NV21_TEXTURE_DATA_LAYER ()

  A getter of NV21 texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::DXT5YCOCG_TEXTURE_DATA_LAYER ()

  A getter of DXT5YCOCG texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::DXT1_TEXTURE_DATA_LAYER ()

  A getter of DXT1 texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::ETC2_TEXTURE_DATA_LAYER ()

  A getter of ETC2 texture data layer GUID.
- MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::ASTC_TEXTURE_DATA_LAYER ()

  A getter of ASTC texture data layer GUID.

8.18.1 Function Documentation

8.18.1.1 ASTC_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::ASTC_TEXTURE_DATA_LAYER ( )
```

A getter of ASTC texture data layer GUID.

Returns

the data layer GUID

8.18.1.2 AUDIO_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::AUDIO_DATA_LAYER ( )
```

A getter of audio data layer GUID.

Returns

the data layer GUID

8.18.1.3 BYTEARRAY_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::BYTEARRAY_DATA_LAYER ( )
```

A getter of bytearray data layer GUID.

Returns

the data layer GUID

8.18.1.4 CAMERA_PARAMS_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::CAMERA_PARAMS_DATA_LAYER ( )
```

A getter of camera params data layer GUID.

Returns

the data layer GUID

8.18.1.5 DEPTHMAP_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::DEPTHMAP_TEXTURE_DATA_LAYER ( )
```

A getter of depth map texture data layer GUID.

Returns

the data layer GUID

8.18.1.6 DXT1_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::DXT1_TEXTURE_DATA_LAYER ( )
```

A getter of DXT1 texture data layer GUID.

Returns

the data layer GUID

8.18.1.7 DXT5YCOCG_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::DXT5YCOCG_TEXTURE_DATA_LAYER ( )
```

A getter of DXT5YCOCG texture data layer GUID.

Returns

the data layer GUID

8.18.1.8 ETC2_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::ETC2_TEXTURE_DATA_LAYER ( )
```

A getter of ETC2 texture data layer GUID.

Returns

the data layer GUID

8.18.1.9 IR_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::IR_TEXTURE_DATA_LAYER ( )
```

A getter of IR texture data layer GUID.

Returns

the data layer GUID

8.18.1.10 NV12_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::NV12_TEXTURE_DATA_LAYER ( )
```

A getter of NV12 texture data layer GUID.

Returns

the data layer GUID

8.18.1.11 NV21_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::NV21_TEXTURE_DATA_LAYER ( )
```

A getter of NV21 texture data layer GUID.

Returns

the data layer GUID

8.18.1.12 NVX_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::NVX_TEXTURE_DATA_LAYER ( )
```

A getter of NVX texture data layer GUID.

Returns

the data layer GUID

8.18.1.13 RGB_TEXTURE_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::RGB_TEXTURE_DATA_LAYER ( )
```

A getter of RGB texture data layer GUID.

Returns

the data layer GUID

8.18.1.14 SEGMENT_INFO_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::SEGMENT_INFO_DATA_LAYER ( )
```

A getter of segment info data layer GUID.

Returns

the data layer GUID

8.18.1.15 TRANSFORM_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::TRANSFORM_DATA_LAYER ( )
```

A getter of transform data layer GUID.

Returns

the data layer GUID

8.18.1.16 VERTEX_COLORS_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_COLORS_DATA_LAYER ( )
```

A getter of vertex colors data layer GUID.

Returns

the data layer GUID

8.18.1.17 VERTEX_INDICES_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_INDICES_DATA_LAYER ( )
```

A getter of vertex indices data layer GUID.

Returns

the data layer GUID

8.18.1.18 VERTEX_NORMALS_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_NORMALS_DATA_LAYER ( )
```

A getter of vertex normals data layer GUID.

Returns

the data layer GUID

8.18.1.19 VERTEX_POSITIONS_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_POSITIONS_DATA_LAYER ( )
```

A getter of vertex positions data layer GUID.

Returns

the data layer GUID

8.18.1.20 VERTEX_UVS_DATA_LAYER()

```
MVX2_API MVCommon::Guid Mvx2API::BasicDataLayersGuids::VERTEX_UVS_DATA_LAYER ( )
```

A getter of vertex UVs data layer GUID.

Returns

the data layer GUID

8.19 public/Mvx2API/data/mesh/MeshDataTypes.h File Reference

```
#include <cstdint>
```

Data Structures

- struct Mvx2API::Vec2Data

  A structure containing 2D position data.
- struct Mvx2API::Vec3Data

  A structure containing 3D position data.
- struct Mvx2API::ColRGBAData

  A structure containing color data.

8.20 public/Mvx2API/data/mesh/MeshIndicesMode.h File Reference

Enumerations

- enum Mvx2API::MeshIndicesMode { Mvx2API::MIM_PointList = 0, Mvx2API::MIM_LineList = 1, Mvx2API::MIM_TriangleList = 2, Mvx2API::MIM_QuadList = 3 }

  Enumeration of indices modes.

8.20.1 Enumeration Type Documentation

8.20.1.1 MeshIndicesMode

`enum Mvx2API::MeshIndicesMode`

Enumeration of indices modes.

Determines proper interpretation of indices sequence of a mesh.

Enumerator

MIM_PointList	Every index represents a single point primitive.
MIM_LineList	Pairs of indices represent line primitives.
MIM_TriangleList	Triplets of indices represent triangle primitives.
MIM_QuadList	Quartets of indices represent quad primitives.

8.21 public/Mvx2API/filters/FilterPtrCreator.h File Reference

```
#include "SharedFilterPtr.h"
```

Functions

- MVX2_API SharedFilterPtr Mvx2API::FilterPtrCreator::CreateFilter (MVCommon::Guid const &filterGuid)

  Creates a filter with a given GUID.
- MVX2_API SharedFilterPtr Mvx2API::FilterPtrCreator::CreateFilter (MVCommon::Guid const &filterGuid, SharedFilterPtr spPrecedingFilter)

  Creates a filter with a given GUID.

8.21.1 Function Documentation

8.21.1.1 CreateFilter() [1/2]

```
MVX2_API SharedFilterPtr Mvx2API::FilterPtrCreator::CreateFilter (
            MVCommon::Guid const & filterGuid )
```

Creates a filter with a given GUID.

Parameters

filterGuid	a GUID of filter to create

Returns

a pointer to a new filter or null pointer if the filter could not be created

8.21.1.2 CreateFilter() [2/2]

```
MVX2_API SharedFilterPtr Mvx2API::FilterPtrCreator::CreateFilter (
            MVCommon::Guid const & filterGuid,
            SharedFilterPtr spPrecedingFilter )
```

Creates a filter with a given GUID.

Parameters

filterGuid	a GUID of filter to create
spPrecedingFilter	a filter the new filter will be pre-initialized with

Returns

a pointer to a new filter or null pointer if the filter could not be created

8.22 public/Mvx2API/frameaccess/extractors/FrameAudioExtractor.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/Guid.h>
```

Functions

- MVX2_API bool Mvx2API::FrameAudioExtractor::GetAudioSamplingInfo (Frame *frame, uint32_t &num↩ Channels, uint32_t &bitsPerSample, uint32_t &numSamplesPerSec, MVCommon::Guid const &purpose↩ Guid=MVCommon::Guid::Nil())

  Returns a frame's audio sampling information.

- MVX2_API uint32_t Mvx2API::FrameAudioExtractor::GetPCMDataOffset (Frame *frame, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Returns a frame's audio pulse-code modulation (PCM) data offset.
- MVX2_API uint32_t Mvx2API::FrameAudioExtractor::GetPCMDataSize (Frame *frame, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Returns a frame's audio pulse-code modulation (PCM) data size (in bytes).
- MVX2_API const uint8_t * Mvx2API::FrameAudioExtractor::GetPCMData (Frame *frame, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  A getter of the raw pointer to audio pulse-code modulation (PCM) data.
- MVX2_API bool Mvx2API::FrameAudioExtractor::CopyPCMData (Frame *frame, uint8_t *targetData, MV↩ Common::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Copies a frame's audio pulse-code modulation (PCM) data.

8.22.1 Function Documentation

8.22.1.1 CopyPCMData()

```
MVX2_API bool Mvx2API::FrameAudioExtractor::CopyPCMData (
            Frame * frame,
            uint8_t * targetData,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Copies a frame's audio pulse-code modulation (PCM) data.

Parameters

frame	a frame
targetData	a target PCM data array (must be pre-allocated with (PCM data size) elements)
purposeGuid	a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the PCM data were successfully copied

8.22.1.2 GetAudioSamplingInfo()

```
MVX2_API bool Mvx2API::FrameAudioExtractor::GetAudioSamplingInfo (
            Frame * frame,
            uint32_t & numChannels,
            uint32_t & bitsPerSample,
            uint32_t & numSamplesPerSec,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns a frame's audio sampling information.

Parameters

frame	a frame
numChannels	an outputted count of audio channels
bitsPerSample	an outputted bits count per sample
numSamplesPerSec	an outputted count of samples per second
purposeGuid	a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the audio sampling information were successfully retrieved

8.22.1.3 GetPCMData()

```
MVX2_API const uint8_t* Mvx2API::FrameAudioExtractor::GetPCMData (
            Frame * frame,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

A getter of the raw pointer to audio pulse-code modulation (PCM) data.

Parameters

frame	a frame
purposeGuid	a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

PCM data

8.22.1.4 GetPCMDataOffset()

```
MVX2_API uint32_t Mvx2API::FrameAudioExtractor::GetPCMDataOffset (
            Frame * frame,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns a frame's audio pulse-code modulation (PCM) data offset.

Parameters

frame	a frame
purposeGuid	a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

PCM data offset

8.22.1.5 GetPCMDataSize()

```
MVX2_API uint32_t Mvx2API::FrameAudioExtractor::GetPCMDataSize (
            Frame * frame,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns a frame's audio pulse-code modulation (PCM) data size (in bytes).

Parameters

frame	a frame
purposeGuid	a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

PCM data size

8.23 public/Mvx2API/frameaccess/extractors/FrameMeshExtractor.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/Guid.h>
```

Functions

- MVX2_API MeshData * Mvx2API::FrameMeshExtractor::GetMeshData (Frame *frame, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Returns a frame's mesh data.

8.23.1 Function Documentation

8.23.1.1 GetMeshData()

```
MVX2_API MeshData* Mvx2API::FrameMeshExtractor::GetMeshData (
            Frame * frame,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns a frame's mesh data.

Parameters

frame	a frame
purposeGuid	a purpose guid of mesh data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

frame's mesh

8.24 public/Mvx2API/frameaccess/extractors/FrameMiscDataExtractor.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/Guid.h>
```

Functions

- MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetColorCameraParams (Frame *frame, MV↩ Common::CameraParams &cameraParams, MVCommon::Guid const &purposeGuid=MVCommon::Guid::↩ Nil())

  Gets color camera parameters of a frame.
- MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetIRCameraParams (Frame *frame, MVCommon::↩ CameraParams &cameraParams, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Gets IR camera parameters of a frame.
- MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetTransform (Frame *frame, MVCommon::Matrix4x4f &transform, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Gets transformation matrix of a frame.
- MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetSegmentID (Frame *frame, uint16_t &segmentID, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Gets an ID of a segment a frame belongs to.
- MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetByteArrayData (Frame *frame, MVCommon::Byte↩ Array &byteArray, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Gets a bytearray data of a frame.

8.24.1 Function Documentation

8.24.1.1 GetByteArrayData()

```
MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetByteArrayData (
            Frame * frame,
            MVCommon::ByteArray & byteArray,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Gets a bytearray data of a frame.

Parameters

frame	a frame
byteArray	a target to store the bytearray data in
purposeGuid	a purpose guid of data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the frame contains bytearray data and it was successfully extracted

8.24.1.2 GetColorCameraParams()

```
MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetColorCameraParams (
            Frame * frame,
            MVCommon::CameraParams & cameraParams,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Gets color camera parameters of a frame.

Parameters

frame	a frame
cameraParams	a target to store the camera parameters in
purposeGuid	a purpose guid of data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the frame contains color camera parameters data and they were successfully extracted

8.24.1.3 GetIRCameraParams()

```
MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetIRCameraParams (
            Frame * frame,
            MVCommon::CameraParams & cameraParams,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Gets IR camera parameters of a frame.

Parameters

frame	a frame
cameraParams	a target to store the camera parameters in
purposeGuid	a purpose guid of data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the frame contains IR camera parameters data and they were successfully extracted

8.24.1.4 GetSegmentID()

```
MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetSegmentID (
            Frame * frame,
            uint16_t & segmentID,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Gets an ID of a segment a frame belongs to.

Parameters

frame	a frame
segmentID	a target to store the segment ID in
purposeGuid	a purpose guid of data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the frame contains segment information data and it was successfully extracted

8.24.1.5 GetTransform()

```
MVX2_API bool Mvx2API::FrameMiscDataExtractor::GetTransform (
            Frame * frame,
            MVCommon::Matrix4x4f & transform,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Gets transformation matrix of a frame.

Parameters

frame	a frame
transform	a target to store the transformation matrix in
purposeGuid	a purpose guid of data to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the frame contains transformation data and it was successfully extracted

8.25 public/Mvx2API/frameaccess/extractors/FrameTextureExtractor.h File Reference

```
#include <Mvx2API/Mvx2API.h>
#include <MVCommon/guid/Guid.h>
```

Enumerations

- enum Mvx2API::FrameTextureExtractor::TextureType { Mvx2API::FrameTextureExtractor::TT_DEPTH = 0, Mvx2API::FrameTextureExtractor::TT_IR = 1, Mvx2API::FrameTextureExtrac = 2, Mvx2API::FrameTextureExtractor::TT_NVX = 3, Mvx2API::FrameTextureExtractor::TT_DXT5YCOCG = 4, Mvx2API::FrameTextureExtractor::TT_DXT1 = 5, Mvx2API::FrameTextureExtractor::TT_ETC2 = 6, Mvx2API::FrameTextureExtractor::TT_ASTC = 7, Mvx2API::FrameTextureExtractor::TT_NV12 = 8, Mvx2API::FrameTextureExtractor::TT_NV21 = 9 }

  An enumeration of texture types.

Functions

- MVX2_API bool Mvx2API::FrameTextureExtractor::GetTextureResolution (Frame *frame, TextureType textureType, uint16_t &width, uint16_t &height, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Returns resolution of a frame's texture.
- MVX2_API uint32_t Mvx2API::FrameTextureExtractor::GetTextureDataSizeInBytes (Frame *frame, TextureType textureType, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Returns size (in bytes) of a frame's texture data.
- MVX2_API const uint8_t * Mvx2API::FrameTextureExtractor::GetTextureData (Frame *frame, TextureType textureType, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Returns raw pointer to the texture data owned by a frame.
- MVX2_API bool Mvx2API::FrameTextureExtractor::CopyTextureData (Frame *frame, TextureType textureType, uint8_t *targetData, MVCommon::Guid const &purposeGuid=MVCommon::Guid::Nil())

  Copies a frame's texture data.

8.25.1 Enumeration Type Documentation

8.25.1.1 TextureType

```
enum Mvx2API::FrameTextureExtractor::TextureType
```

An enumeration of texture types.

Enumerator

TT_DEPTH	Depth map texture type.
TT_IR	IR texture type.
TT_RGB	RGB texture type.
TT_NVX	NVX texture type.
TT_DXT5YCOCG	DXT5YCOCG texture type.
TT_DXT1	DXT1 texture type.
TT_ETC2	ETC texture type.
TT_ASTC	ASTC texture type.
TT_NV12	NV12 texture type.
TT_NV21	NV21 texture type.

8.25.2 Function Documentation

8.25.2.1 CopyTextureData()

```
MVX2_API bool Mvx2API::FrameTextureExtractor::CopyTextureData (
            Frame * frame,
            TextureType textureType,
            uint8_t * targetData,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Copies a frame's texture data.

Parameters

frame	a frame
textureType	a type of the texture to extract
targetData	an outputted texture data array (must be pre-allocated with (texture data size) elements)
purposeGuid	a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the texture data were successfully copied

8.25.2.2 GetTextureData()

```
MVX2_API const uint8_t* Mvx2API::FrameTextureExtractor::GetTextureData (
            Frame * frame,
            TextureType textureType,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns raw pointer to the texture data owned by a frame.

Parameters

frame	a frame
textureType	a type of the texture to extract
purposeGuid	a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

texture data

8.25.2.3 GetTextureDataSizeInBytes()

```
MVX2_API uint32_t Mvx2API::FrameTextureExtractor::GetTextureDataSizeInBytes (
            Frame * frame,
            TextureType textureType,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns size (in bytes) of a frame's texture data.

Parameters

frame	a frame
textureType	a type of the texture to extract
purposeGuid	a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

texture data size

8.25.2.4 GetTextureResolution()

```
MVX2_API bool Mvx2API::FrameTextureExtractor::GetTextureResolution (
            Frame * frame,
            TextureType textureType,
            uint16_t & width,
            uint16_t & height,
            MVCommon::Guid const & purposeGuid = MVCommon::Guid::Nil() )
```

Returns resolution of a frame's texture.

Parameters

frame	a frame
textureType	a type of the texture to extract
width	an outputted width of the texture
height	an outputted height of the texture
purposeGuid	a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid)

Returns

true if the texture resolution was successfully retrieved

8.26 public/Mvx2API/runners/RunnerPlaybackMode.h File Reference

Enumerations

- enum Mvx2API::RunnerPlaybackMode { Mvx2API::RPM_FORWARD_ONCE = 0, Mvx2API::RPM_FORWARD_LOOP = 1, Mvx2API::RPM_BACKWARD_ONCE = 2, Mvx2API::RPM_BACKWARD_LOOP = 3, Mvx2API::RPM_PINGPONG = 4, Mvx2API::RPM_PINGPONG_INVERSE = 5, Mvx2API::RPM_REALTIME = 255 }

  An enumeration of supported MVX stream playback modes.

8.26.1 Enumeration Type Documentation

8.26.1.1 RunnerPlaybackMode

`enum Mvx2API::RunnerPlaybackMode`

An enumeration of supported MVX stream playback modes.

Enumerator

RPM_FORWARD_ONCE	A stream is only played once in a forward direction.
RPM_FORWARD_LOOP	A stream is played in a loop in a forward direction.
RPM_BACKWARD_ONCE	A stream is only played once in a backward direction.
RPM_BACKWARD_LOOP	A stream is played in a loop in a backward direction.
RPM_PINGPONG	A stream is played in a loop in the alternating directions (ping-pong), starting with the forward direction.
RPM_PINGPONG_INVERSE	A stream is played in a loop in the alternating directions (ping-pong), starting with the backward direction.
RPM_REALTIME	A stream is played real-time as a 'live' data source produces frames.

8.27 public/Mvx2API/runners/RunnerPlaybackState.h File Reference

Enumerations

- enum Mvx2API::RunnerPlaybackState { Mvx2API::RPS_Stopped = 0, Mvx2API::RPS_Paused, Mvx2API::RPS_Playing }

  An enumeration of runner playback states.

8.27.1 Enumeration Type Documentation

8.27.1.1 RunnerPlaybackState

```
enum Mvx2API::RunnerPlaybackState
```

An enumeration of runner playback states.

Enumerator

RPS_Stopped	A runner is stopped.
RPS_Paused	A runner is running and is paused.
RPS_Playing	A runner is running and playing.

8.28 public/Mvx2API/utils/PluginsLoader.h File Reference

```
#include <Mvx2API/Mvx2API.h>
```

Functions

- MVX2_API void Mvx2API::PluginsLoader::LoadPluginsInFolder (MVCommon::String const &folder, bool checkCacheFile=false, bool storeCacheFile=false, bool checkSubfolders=true)

  Loads all MVX plugins from a specified folder.
- MVX2_API void Mvx2API::PluginsLoader::LoadPlugin (MVCommon::String const &pluginPath)

  Loads single MVX plugin specified by its path.

8.28.1 Function Documentation

8.28.1.1 LoadPlugin()

```
MVX2_API void Mvx2API::PluginsLoader::LoadPlugin (
            MVCommon::String const & pluginPath )
```

Loads single MVX plugin specified by its path.

Parameters

pluginPath	a path to the plugin

8.28.1.2 LoadPluginsInFolder()

```
MVX2_API void Mvx2API::PluginsLoader::LoadPluginsInFolder (
            MVCommon::String const & folder,
            bool checkCacheFile = false,
            bool storeCacheFile = false,
            bool checkSubfolders = true )
```

Loads all MVX plugins from a specified folder.

Parameters

folder	a folder containing MVX plugins
checkCacheFile	an indication whether to check existing cache file of plugins and their filters
storeCacheFile	an indication whether to store information about plugins and their filters into a cache file
checkSubfolders	if true, checks also subfolders of the folder

Index