

Mvx2Net

Generated by Doxygen 1.8.16

| | |
|---|-----------|
| 1 Mantis Vision: Mvx2 | 1 |
| 2 Mvx2API | 3 |
| 3 Release Notes | 7 |
| 4 Namespace Index | 17 |
| 4.1 Packages | 17 |
| 5 Hierarchical Index | 19 |
| 5.1 Class Hierarchy | 19 |
| 6 Class Index | 21 |
| 6.1 Class List | 21 |
| 7 Namespace Documentation | 25 |
| 7.1 Mvx2API Namespace Reference | 25 |
| 7.1.1 Enumeration Type Documentation | 27 |
| 7.1.1.1 MeshIndicesMode | 27 |
| 7.1.1.2 RunnerPlaybackMode | 28 |
| 7.1.1.3 RunnerPlaybackState | 28 |
| 7.2 Mvx2API.Experimental Namespace Reference | 28 |
| 8 Class Documentation | 31 |
| 8.1 Mvx2API.AsyncFrameAccessGraphNode Class Reference | 31 |
| 8.1.1 Detailed Description | 31 |
| 8.1.2 Constructor & Destructor Documentation | 31 |
| 8.1.2.1 AsyncFrameAccessGraphNode() | 31 |
| 8.1.3 Member Function Documentation | 32 |
| 8.1.3.1 SetFrameListener() | 32 |
| 8.2 Mvx2API.AutoCompressorGraphNode Class Reference | 32 |
| 8.2.1 Detailed Description | 32 |
| 8.2.2 Constructor & Destructor Documentation | 33 |
| 8.2.2.1 AutoCompressorGraphNode() | 33 |
| 8.3 Mvx2API.AutoDecompressorGraphNode Class Reference | 33 |
| 8.3.1 Detailed Description | 33 |
| 8.3.2 Constructor & Destructor Documentation | 33 |
| 8.3.2.1 AutoDecompressorGraphNode() | 33 |
| 8.4 Mvx2API.AutoSequentialGraphRunner Class Reference | 34 |
| 8.4.1 Detailed Description | 34 |
| 8.4.2 Constructor & Destructor Documentation | 34 |
| 8.4.2.1 AutoSequentialGraphRunner() | 34 |
| 8.4.3 Member Function Documentation | 35 |
| 8.4.3.1 GetPlaybackState() | 35 |
| 8.4.3.2 Pause() | 35 |

| | |
|---|----|
| 8.4.3.3 Play() | 35 |
| 8.4.3.4 Resume() | 36 |
| 8.4.3.5 SeekFrame() | 36 |
| 8.4.3.6 Stop() | 36 |
| 8.5 Mvx2API.BasicDataLayersGuids Class Reference | 37 |
| 8.5.1 Detailed Description | 38 |
| 8.6 Mvx2API.BlockFPSGraphNode Class Reference | 38 |
| 8.6.1 Detailed Description | 38 |
| 8.6.2 Constructor & Destructor Documentation | 38 |
| 8.6.2.1 BlockFPSGraphNode() | 38 |
| 8.6.3 Member Function Documentation | 39 |
| 8.6.3.1 SetFPS() | 39 |
| 8.7 Mvx2API.BlockGraphNode Class Reference | 39 |
| 8.7.1 Detailed Description | 40 |
| 8.7.2 Member Enumeration Documentation | 40 |
| 8.7.2.1 FullBehaviour | 40 |
| 8.7.3 Constructor & Destructor Documentation | 40 |
| 8.7.3.1 BlockGraphNode() | 40 |
| 8.7.4 Member Function Documentation | 41 |
| 8.7.4.1 GetDroppedFramesCount() | 41 |
| 8.7.4.2 SetFullBehaviour() | 41 |
| 8.8 Mvx2API.BlockManualGraphNode Class Reference | 41 |
| 8.8.1 Detailed Description | 42 |
| 8.8.2 Constructor & Destructor Documentation | 42 |
| 8.8.2.1 BlockManualGraphNode() | 42 |
| 8.8.3 Member Function Documentation | 42 |
| 8.8.3.1 PullNextProcessedFrame() | 42 |
| 8.9 Mvx2API.Col Struct Reference | 43 |
| 8.9.1 Detailed Description | 43 |
| 8.10 Mvx2API.DataProfile Class Reference | 43 |
| 8.10.1 Detailed Description | 44 |
| 8.10.2 Constructor & Destructor Documentation | 44 |
| 8.10.2.1 DataProfile() | 44 |
| 8.11 Mvx2API.DataProfileEnumerator Class Reference | 45 |
| 8.11.1 Detailed Description | 45 |
| 8.11.2 Constructor & Destructor Documentation | 45 |
| 8.11.2.1 DataProfileEnumerator() | 45 |
| 8.12 Mvx2API.DelegatedFrameListener Class Reference | 45 |
| 8.12.1 Detailed Description | 46 |
| 8.12.2 Constructor & Destructor Documentation | 46 |
| 8.12.2.1 DelegatedFrameListener() | 46 |
| 8.12.3 Member Function Documentation | 46 |

| | |
|--|----|
| 8.12.3.1 OnFrameProcessed() | 46 |
| 8.12.3.2 OnFrameProcessedDelegate() | 47 |
| 8.13 Mvx2API.DelegatedParameterValueChangeListener Class Reference | 47 |
| 8.13.1 Detailed Description | 48 |
| 8.13.2 Constructor & Destructor Documentation | 48 |
| 8.13.2.1 DelegatedParameterValueChangeListener() | 48 |
| 8.13.3 Member Function Documentation | 48 |
| 8.13.3.1 OnParameterValueChanged() | 48 |
| 8.13.3.2 OnParameterValueChangedDelegate() | 49 |
| 8.14 Mvx2API.FilterParameterNameEnumerator Class Reference | 49 |
| 8.14.1 Detailed Description | 49 |
| 8.14.2 Constructor & Destructor Documentation | 49 |
| 8.14.2.1 FilterParameterNameEnumerator() | 49 |
| 8.15 Mvx2API.Frame Class Reference | 50 |
| 8.15.1 Detailed Description | 51 |
| 8.15.2 Constructor & Destructor Documentation | 51 |
| 8.15.2.1 Frame() | 51 |
| 8.15.3 Member Function Documentation | 51 |
| 8.15.3.1 ActivateStreamWithIndex() | 51 |
| 8.15.3.2 CreateDataProfilesEnumerator() | 52 |
| 8.15.3.3 GetActiveStreamIndex() | 52 |
| 8.15.3.4 GetNumStreams() | 52 |
| 8.15.3.5 GetStreamAtomNr() | 52 |
| 8.15.3.6 GetStreamAtomTimestamp() | 53 |
| 8.15.3.7 GetStreamId() | 53 |
| 8.15.3.8 StreamContainsDataLayer() [1/2] | 53 |
| 8.15.3.9 StreamContainsDataLayer() [2/2] | 53 |
| 8.16 Mvx2API.FrameAccessGraphNode Class Reference | 55 |
| 8.16.1 Detailed Description | 55 |
| 8.16.2 Member Function Documentation | 55 |
| 8.16.2.1 GetRecentProcessedFrame() | 56 |
| 8.17 Mvx2API.FrameAudioExtractor Class Reference | 56 |
| 8.17.1 Detailed Description | 57 |
| 8.17.2 Member Function Documentation | 57 |
| 8.17.2.1 CopyPCMDData() [1/2] | 57 |
| 8.17.2.2 CopyPCMDData() [2/2] | 57 |
| 8.17.2.3 CopyPCMDDataRaw() [1/2] | 58 |
| 8.17.2.4 CopyPCMDDataRaw() [2/2] | 58 |
| 8.17.2.5 GetAudioSamplingInfo() [1/2] | 58 |
| 8.17.2.6 GetAudioSamplingInfo() [2/2] | 59 |
| 8.17.2.7 GetPCMDData() [1/2] | 59 |
| 8.17.2.8 GetPCMDData() [2/2] | 60 |

| | |
|---|----|
| 8.17.2.9 GetPCMDDataOffset() [1/2] | 60 |
| 8.17.2.10 GetPCMDDataOffset() [2/2] | 60 |
| 8.17.2.11 GetPCMDDataSize() [1/2] | 61 |
| 8.17.2.12 GetPCMDDataSize() [2/2] | 61 |
| 8.18 Mvx2API.FrameListener Class Reference | 62 |
| 8.18.1 Detailed Description | 62 |
| 8.18.2 Member Function Documentation | 62 |
| 8.18.2.1 OnFrameProcessed() | 62 |
| 8.19 Mvx2API.FrameMeshExtractor Class Reference | 63 |
| 8.19.1 Detailed Description | 63 |
| 8.19.2 Member Function Documentation | 63 |
| 8.19.2.1 GetMeshData() [1/2] | 63 |
| 8.19.2.2 GetMeshData() [2/2] | 64 |
| 8.20 Mvx2API.FrameMiscDataExtractor Class Reference | 64 |
| 8.20.1 Detailed Description | 65 |
| 8.20.2 Member Function Documentation | 65 |
| 8.20.2.1 GetByteArrayData() [1/2] | 65 |
| 8.20.2.2 GetByteArrayData() [2/2] | 65 |
| 8.20.2.3 GetColorCameraParams() [1/2] | 66 |
| 8.20.2.4 GetColorCameraParams() [2/2] | 66 |
| 8.20.2.5 GetIRCameraParams() [1/2] | 66 |
| 8.20.2.6 GetIRCameraParams() [2/2] | 67 |
| 8.20.2.7 GetSegmentID() [1/2] | 67 |
| 8.20.2.8 GetSegmentID() [2/2] | 68 |
| 8.20.2.9 GetTransform() [1/2] | 68 |
| 8.20.2.10 GetTransform() [2/2] | 68 |
| 8.21 Mvx2API.FrameTextureExtractor Class Reference | 69 |
| 8.21.1 Detailed Description | 70 |
| 8.21.2 Member Enumeration Documentation | 70 |
| 8.21.2.1 TextureType | 70 |
| 8.21.3 Member Function Documentation | 70 |
| 8.21.3.1 CopyTextureData() [1/2] | 70 |
| 8.21.3.2 CopyTextureData() [2/2] | 71 |
| 8.21.3.3 CopyTextureDataRow() [1/2] | 71 |
| 8.21.3.4 CopyTextureDataRow() [2/2] | 72 |
| 8.21.3.5 GetTextureData() [1/2] | 72 |
| 8.21.3.6 GetTextureData() [2/2] | 72 |
| 8.21.3.7 GetTextureDataSizeInBytes() [1/2] | 73 |
| 8.21.3.8 GetTextureDataSizeInBytes() [2/2] | 73 |
| 8.21.3.9 GetTextureResolution() [1/2] | 74 |
| 8.21.3.10 GetTextureResolution() [2/2] | 74 |
| 8.22 Mvx2API.Graph Class Reference | 75 |

| | |
|--|----|
| 8.22.1 Detailed Description | 75 |
| 8.22.2 Member Function Documentation | 75 |
| 8.22.2.1 Reinitialize() | 75 |
| 8.23 Mvx2API.GraphBuilder Class Reference | 76 |
| 8.23.1 Detailed Description | 76 |
| 8.23.2 Constructor & Destructor Documentation | 76 |
| 8.23.2.1 GraphBuilder() | 76 |
| 8.23.3 Member Function Documentation | 77 |
| 8.23.3.1 CompileGraphAndReset() | 77 |
| 8.23.3.2 ContainsDataProfile() | 77 |
| 8.23.3.3 CreateDataProfilesEnumerator() | 78 |
| 8.23.3.4 Refresh() | 78 |
| 8.24 Mvx2API.GraphNode Class Reference | 78 |
| 8.24.1 Detailed Description | 79 |
| 8.24.2 Constructor & Destructor Documentation | 79 |
| 8.24.2.1 GraphNode() | 79 |
| 8.24.3 Member Function Documentation | 79 |
| 8.24.3.1 NativeObjectToGraphNode() | 79 |
| 8.25 Mvx2API.GraphRunner Class Reference | 80 |
| 8.25.1 Detailed Description | 80 |
| 8.25.2 Constructor & Destructor Documentation | 80 |
| 8.25.2.1 GraphRunner() | 80 |
| 8.25.3 Member Function Documentation | 81 |
| 8.25.3.1 GetSourceInfo() | 81 |
| 8.26 Mvx2API.InjectFileDataGraphNode Class Reference | 81 |
| 8.26.1 Detailed Description | 81 |
| 8.26.2 Constructor & Destructor Documentation | 81 |
| 8.26.2.1 InjectFileDataGraphNode() | 81 |
| 8.26.3 Member Function Documentation | 82 |
| 8.26.3.1 SetFile() | 82 |
| 8.27 Mvx2API.InjectMemoryDataGraphNode Class Reference | 82 |
| 8.27.1 Detailed Description | 82 |
| 8.27.2 Constructor & Destructor Documentation | 83 |
| 8.27.2.1 InjectMemoryDataGraphNode() | 83 |
| 8.27.3 Member Function Documentation | 83 |
| 8.27.3.1 SetData() | 83 |
| 8.28 Mvx2API.InputEvent Class Reference | 83 |
| 8.28.1 Detailed Description | 84 |
| 8.28.2 Constructor & Destructor Documentation | 84 |
| 8.28.2.1 InputEvent() | 84 |
| 8.29 Mvx2API.KeyDownEvent Class Reference | 84 |
| 8.29.1 Detailed Description | 85 |

| | |
|--|----|
| 8.29.2 Constructor & Destructor Documentation | 85 |
| 8.29.2.1 KeyDownEvent() | 85 |
| 8.30 Mvx2API.KeyUpEvent Class Reference | 85 |
| 8.30.1 Detailed Description | 85 |
| 8.30.2 Constructor & Destructor Documentation | 86 |
| 8.30.2.1 KeyUpEvent() | 86 |
| 8.31 Mvx2API.ManualGraphBuilder Class Reference | 86 |
| 8.31.1 Detailed Description | 86 |
| 8.31.2 Member Function Documentation | 87 |
| 8.31.2.1 AppendGraphNode() | 87 |
| 8.31.2.2 operator+() | 88 |
| 8.32 Mvx2API.ManualLiveFrameSourceGraphNode Class Reference | 88 |
| 8.32.1 Detailed Description | 89 |
| 8.32.2 Constructor & Destructor Documentation | 89 |
| 8.32.2.1 ManualLiveFrameSourceGraphNode() | 89 |
| 8.32.3 Member Function Documentation | 89 |
| 8.32.3.1 ClearCache() | 90 |
| 8.32.3.2 ClearCacheAndReinitializeProperties() | 90 |
| 8.32.3.3 PropertiesAreInitialized() | 90 |
| 8.32.3.4 PushFrame() | 91 |
| 8.33 Mvx2API.ManualOfflineFrameSourceGraphNode Class Reference | 91 |
| 8.33.1 Detailed Description | 92 |
| 8.33.2 Constructor & Destructor Documentation | 92 |
| 8.33.2.1 ManualOfflineFrameSourceGraphNode() | 92 |
| 8.33.3 Member Function Documentation | 92 |
| 8.33.3.1 ClearCache() | 92 |
| 8.33.3.2 ClearCacheAndReinitializeProperties() | 93 |
| 8.33.3.3 PropertiesAreInitialized() | 93 |
| 8.33.3.4 PushFrame() | 94 |
| 8.34 Mvx2API.ManualSequentialGraphRunner Class Reference | 95 |
| 8.34.1 Detailed Description | 95 |
| 8.34.2 Constructor & Destructor Documentation | 95 |
| 8.34.2.1 ManualSequentialGraphRunner() | 95 |
| 8.34.3 Member Function Documentation | 96 |
| 8.34.3.1 ProcessNextFrame() | 96 |
| 8.34.3.2 RestartWithPlaybackMode() | 96 |
| 8.34.3.3 SeekFrame() | 96 |
| 8.35 Mvx2API.MeshData Class Reference | 97 |
| 8.35.1 Detailed Description | 98 |
| 8.35.2 Member Function Documentation | 98 |
| 8.35.2.1 CopyBoundingBox() | 98 |
| 8.35.2.2 CopyBoundingBoxRaw() | 99 |

| | |
|--|-----|
| 8.35.2.3 CopyColorsColRGBA() | 99 |
| 8.35.2.4 CopyColorsRGB() | 99 |
| 8.35.2.5 CopyColorsRGBARaw() | 101 |
| 8.35.2.6 CopyColorsRGBRaw() | 101 |
| 8.35.2.7 CopyIndices() | 102 |
| 8.35.2.8 CopyIndicesRaw() | 102 |
| 8.35.2.9 CopyNormals() | 102 |
| 8.35.2.10 CopyNormalsRaw() | 103 |
| 8.35.2.11 CopyNormalsVec3() | 103 |
| 8.35.2.12 CopyUVs() | 103 |
| 8.35.2.13 CopyUVsRaw() | 104 |
| 8.35.2.14 CopyUVsVec2() | 104 |
| 8.35.2.15 CopyVertices() | 104 |
| 8.35.2.16 CopyVerticesRaw() | 106 |
| 8.35.2.17 CopyVerticesVec3() | 106 |
| 8.35.2.18 GetColorsRGB() | 106 |
| 8.35.2.19 GetIndices() | 107 |
| 8.35.2.20 GetNormals() | 107 |
| 8.35.2.21 GetNumColors() | 107 |
| 8.35.2.22 GetNumIndices() | 107 |
| 8.35.2.23 GetNumNormals() | 108 |
| 8.35.2.24 GetNumUVs() | 108 |
| 8.35.2.25 GetNumVertices() | 108 |
| 8.35.2.26 GetUVs() | 108 |
| 8.35.2.27 GetVertices() | 109 |
| 8.36 Mvx2API.MeshSplitter Class Reference | 109 |
| 8.36.1 Detailed Description | 109 |
| 8.36.2 Constructor & Destructor Documentation | 109 |
| 8.36.2.1 MeshSplitter() | 109 |
| 8.36.3 Member Function Documentation | 110 |
| 8.36.3.1 GetSplitMeshData() | 110 |
| 8.36.3.2 GetSplitMeshesCount() | 110 |
| 8.36.3.3 SplitMesh() | 110 |
| 8.37 Mvx2API.MouseDoubleClickEvent Class Reference | 111 |
| 8.37.1 Detailed Description | 111 |
| 8.37.2 Constructor & Destructor Documentation | 111 |
| 8.37.2.1 MouseDoubleClickEvent() | 111 |
| 8.38 Mvx2API.MouseDownEvent Class Reference | 112 |
| 8.38.1 Detailed Description | 112 |
| 8.38.2 Constructor & Destructor Documentation | 112 |
| 8.38.2.1 MouseDownEvent() | 112 |
| 8.39 Mvx2API.MouseMoveEvent Class Reference | 113 |

| | |
|---|-----|
| 8.39.1 Detailed Description | 113 |
| 8.39.2 Constructor & Destructor Documentation | 113 |
| 8.39.2.1 MouseMoveEvent() | 113 |
| 8.40 Mvx2API.MouseUpEvent Class Reference | 114 |
| 8.40.1 Detailed Description | 114 |
| 8.40.2 Constructor & Destructor Documentation | 114 |
| 8.40.2.1 MouseUpEvent() | 114 |
| 8.41 Mvx2API.MouseWheelEvent Class Reference | 114 |
| 8.41.1 Detailed Description | 115 |
| 8.41.2 Constructor & Destructor Documentation | 115 |
| 8.41.2.1 MouseWheelEvent() | 115 |
| 8.42 Mvx2API.ParameterValueChangedListener Class Reference | 115 |
| 8.42.1 Detailed Description | 116 |
| 8.42.2 Member Function Documentation | 116 |
| 8.42.2.1 OnParameterValueChanged() | 116 |
| 8.43 Mvx2API.PluginsLoader Class Reference | 116 |
| 8.43.1 Detailed Description | 117 |
| 8.43.2 Member Function Documentation | 117 |
| 8.43.2.1 LoadPlugin() | 117 |
| 8.43.2.2 LoadPluginsInFolder() | 117 |
| 8.44 Mvx2API.RandomAccessGraphRunner Class Reference | 118 |
| 8.44.1 Detailed Description | 118 |
| 8.44.2 Constructor & Destructor Documentation | 118 |
| 8.44.2.1 RandomAccessGraphRunner() | 118 |
| 8.44.3 Member Function Documentation | 118 |
| 8.44.3.1 ProcessFrame() | 118 |
| 8.45 Mvx2API.Experimental.RendererGraphNode Class Reference | 119 |
| 8.45.1 Detailed Description | 119 |
| 8.45.2 Constructor & Destructor Documentation | 119 |
| 8.45.2.1 RendererGraphNode() | 119 |
| 8.45.3 Member Function Documentation | 120 |
| 8.45.3.1 DestroyRenderer() | 120 |
| 8.45.3.2 HandleInputEvent() | 120 |
| 8.45.3.3 Render() | 120 |
| 8.46 Mvx2API.SingleFilterGraphNode Class Reference | 121 |
| 8.46.1 Detailed Description | 122 |
| 8.46.2 Constructor & Destructor Documentation | 122 |
| 8.46.2.1 SingleFilterGraphNode() [1/2] | 122 |
| 8.46.2.2 SingleFilterGraphNode() [2/2] | 122 |
| 8.46.3 Member Function Documentation | 123 |
| 8.46.3.1 ContainsDataProfile() | 123 |
| 8.46.3.2 CreateDataProfilesEnumerator() | 123 |

| | |
|--|------------|
| 8.46.3.3 CreateParameterNamesEnumerator() | 124 |
| 8.46.3.4 RegisterParameterValueChangedListener() | 124 |
| 8.46.3.5 SetFilterParameterValue() | 125 |
| 8.46.3.6 TryGetFilterParameterValue() | 125 |
| 8.46.3.7 UnregisterParameterValueChangedListener() | 126 |
| 8.47 Mvx2API.SourceInfo Class Reference | 126 |
| 8.47.1 Detailed Description | 126 |
| 8.47.2 Member Function Documentation | 127 |
| 8.47.2.1 ContainsDataLayer() [1/2] | 127 |
| 8.47.2.2 ContainsDataLayer() [2/2] | 127 |
| 8.47.2.3 CreateDataProfilesEnumerator() | 127 |
| 8.47.2.4 GetFPS() | 128 |
| 8.47.2.5 GetNumFrames() | 128 |
| 8.48 Mvx2API.Utils Class Reference | 128 |
| 8.48.1 Detailed Description | 129 |
| 8.48.2 Member Function Documentation | 129 |
| 8.48.2.1 GetAppExeDirectory() | 129 |
| 8.48.2.2 GetAppExeFilePath() | 129 |
| 8.48.3 Property Documentation | 129 |
| 8.48.3.1 MVXLoggerInstance | 129 |
| 8.49 Mvx2API.Vec2 Struct Reference | 130 |
| 8.49.1 Detailed Description | 130 |
| 8.50 Mvx2API.Vec3 Struct Reference | 130 |
| 8.50.1 Detailed Description | 130 |
| Index | 131 |

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

- [Mvx2API](#)
- [Release Notes](#)

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 pluggins, 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 accessing 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 occurring 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::ErrorHandler` so it can store and provide last (human-readable) error when its `InstantiateMvxGraph()` or `InstantiateSimpleMvxGraph()` 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 `Mvx2Net.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 `MXV2_API` macro decoration from template functions:
 - `MXV::DataLayerFactory::CreateDataLayer` (2 overloads)
 - `MXV::FilterFactory::CreateFilter`
 - `MXV::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::ParameterNamesEnd](#) 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 `alsoDeinitialize` 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 occurring when `MVX::MutateTextureColor`, `MVX::TransformTextureConversion` and `MVX::MutateCompressor` classes are used or derived from
- **4.2.0_F2** | introduced new named purpose guides:
 - `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::TextureFormatConverter::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 `MVCCommon` 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 targetBoundingBox)` function to `Mvx2API::MeshData::CopyBoundingBoxRaw(IntPtr targetBoundingBox)`
- **5.0.0_MA2** | introduced `Mvx2API::FrameAudioExtractor::CopyPCMDDataRaw()` functions to Mvx2Net module as alternatives to `Mvx2API::FrameAudioExtractor::CopyPCMDData()` 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::ParameterValueChangeListener` 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::DataProfiles↵End()` 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::SingleFilterGraph↵Node::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 `MX::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

Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

| | |
|--|----|
| Mvx2API | 25 |
| Mvx2API.Experimental | 28 |

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

| | |
|---|-----|
| Mvx2API.BasicDataLayersGuids | 37 |
| Mvx2API.Col | 43 |
| Mvx2API.FrameAudioExtractor | 56 |
| Mvx2API.FrameMeshExtractor | 63 |
| Mvx2API.FrameMiscDataExtractor | 64 |
| Mvx2API.FrameTextureExtractor | 69 |
| IEnumerator | |
| Mvx2API.DataProfileEnumerator | 45 |
| Mvx2API.FilterParameterNameEnumerator | 49 |
| IEquatable | |
| Mvx2API.DataProfile | 43 |
| NativeObjectHolder | |
| Mvx2API.DataProfile | 43 |
| Mvx2API.Frame | 50 |
| Mvx2API.FrameListener | 62 |
| Mvx2API.DelegatedFrameListener | 45 |
| Mvx2API.Graph | 75 |
| Mvx2API.GraphBuilder | 76 |
| Mvx2API.ManualGraphBuilder | 86 |
| Mvx2API.GraphNode | 78 |
| Mvx2API.AsyncFrameAccessGraphNode | 31 |
| Mvx2API.AutoCompressorGraphNode | 32 |
| Mvx2API.AutoDecompressorGraphNode | 33 |
| Mvx2API.BlockGraphNode | 39 |
| Mvx2API.BlockFPSGraphNode | 38 |
| Mvx2API.BlockManualGraphNode | 41 |
| Mvx2API.Experimental.RendererGraphNode | 119 |
| Mvx2API.FrameAccessGraphNode | 55 |
| Mvx2API.InjectFileDataGraphNode | 81 |
| Mvx2API.InjectMemoryDataGraphNode | 82 |
| Mvx2API.ManualLiveFrameSourceGraphNode | 88 |
| Mvx2API.ManualOfflineFrameSourceGraphNode | 91 |
| Mvx2API.SingleFilterGraphNode | 121 |
| Mvx2API.GraphRunner | 80 |
| Mvx2API.AutoSequentialGraphRunner | 34 |

| | |
|--|-----|
| Mvx2API.ManualSequentialGraphRunner | 95 |
| Mvx2API.RandomAccessGraphRunner | 118 |
| Mvx2API.InputEvent | 83 |
| Mvx2API.KeyDownEvent | 84 |
| Mvx2API.KeyUpEvent | 85 |
| Mvx2API.MouseDoubleClickEvent | 111 |
| Mvx2API.MouseDownEvent | 112 |
| Mvx2API.MouseMoveEvent | 113 |
| Mvx2API.MouseUpEvent | 114 |
| Mvx2API.MouseWheelEvent | 114 |
| Mvx2API.MeshData | 97 |
| Mvx2API.MeshSplitter | 109 |
| Mvx2API.ParameterValueChangedListener | 115 |
| Mvx2API.DelegatedParameterValueChangedListener | 47 |
| Mvx2API.SourceInfo | 126 |
| Mvx2API.PluginsLoader | 116 |
| Mvx2API.Utils | 128 |
| Mvx2API.Vec2 | 130 |
| Mvx2API.Vec3 | 130 |

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

| | |
|--|----|
| Mvx2API.AsyncFrameAccessGraphNode | 31 |
| A graph node for asynchronous notifications about processed MVX frames | |
| Mvx2API.AutoCompressorGraphNode | 32 |
| A graph node for auto-compression of MVX data | |
| Mvx2API.AutoDecompressorGraphNode | 33 |
| A graph node for auto-decompression of MVX data | |
| Mvx2API.AutoSequentialGraphRunner | 34 |
| A sequential runner of data-processing graphs with automatic (synchronous/asynchronous) updates-invocation | |
| Mvx2API.BasicDataLayersGuids | 37 |
| A collection of GUID constants for unique identification of simple data layers | |
| Mvx2API.BlockFPSGraphNode | 38 |
| A blocking graph node with an automatized framerate-based frames-pulling capability | |
| Mvx2API.BlockGraphNode | 39 |
| A graph node with a buffering and execution-blocking capabilities | |
| Mvx2API.BlockManualGraphNode | 41 |
| A blocking graph node with a manual frames-pulling capability | |
| Mvx2API.Col | 43 |
| A structure containing color data | |
| Mvx2API.DataProfile | 43 |
| A profile of a single data item | |
| Mvx2API.DataProfileEnumerator | 45 |
| An iterator over profiles of data contained in a frame | |
| Mvx2API.DelegatedFrameListener | 45 |
| A listener for asynchronous reception of frames with an external delegate | |
| Mvx2API.DelegatedParameterValueChangeListener | 47 |
| A listener for changes of graph nodes' parameters with an external delegate | |
| Mvx2API.FilterParameterNameEnumerator | 49 |
| An iterator over names of filter parameters of a SingleFilterGraphNode | |
| Mvx2API.Frame | 50 |
| A frame of data | |
| Mvx2API.FrameAccessGraphNode | 55 |
| A graph node for direct access to processed MVX frames | |
| Mvx2API.FrameAudioExtractor | 56 |
| An extractor of audio data from frames | |

| | |
|--|-----|
| Mvx2API.FrameListener | |
| A listener for asynchronous reception of frames | 62 |
| Mvx2API.FrameMeshExtractor | |
| An extractor of mesh data from frames | 63 |
| Mvx2API.FrameMiscDataExtractor | |
| An extractor of miscellaneous data from frames | 64 |
| Mvx2API.FrameTextureExtractor | |
| An extractor of texture data from frames | 69 |
| Mvx2API.Graph | |
| A graph of data-processing nodes | 75 |
| Mvx2API.GraphBuilder | |
| A builder of data-processing graphs | 76 |
| Mvx2API.GraphNode | |
| A processing node | 78 |
| Mvx2API.GraphRunner | |
| A runner of data-processing graphs | 80 |
| Mvx2API.InjectFileDataGraphNode | |
| A graph node for injecting binary data from files to frames | 81 |
| Mvx2API.InjectMemoryDataGraphNode | |
| A graph node for injecting binary data from memory to frames | 82 |
| Mvx2API.InputEvent | |
| An input event structure | 83 |
| Mvx2API.KeyDownEvent | |
| A 'key down' event | 84 |
| Mvx2API.KeyUpEvent | |
| A 'key up' event | 85 |
| Mvx2API.ManualGraphBuilder | |
| A manual builder of data-processing graphs | 86 |
| Mvx2API.ManualLiveFrameSourceGraphNode | |
| A source graph node for manual production of MVX frames | 88 |
| Mvx2API.ManualOfflineFrameSourceGraphNode | |
| A source graph node for manual production of MVX frames | 91 |
| Mvx2API.ManualSequentialGraphRunner | |
| A sequential runner of data-processing graphs with manual updates-invocation | 95 |
| Mvx2API.MeshData | |
| A class containing data of a single mesh | 97 |
| Mvx2API.MeshSplitter | |
| 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 | 109 |
| Mvx2API.MouseDoubleClickEvent | |
| A 'mouse double-click' event | 111 |
| Mvx2API.MouseDownEvent | |
| A 'mouse down' event | 112 |
| Mvx2API.MouseMoveEvent | |
| A 'mouse move' event | 113 |
| Mvx2API.MouseUpEvent | |
| A 'mouse up' event | 114 |
| Mvx2API.MouseWheelEvent | |
| A 'mouse wheel' event | 114 |
| Mvx2API.ParameterValueChangedListener | |
| A listener for changes of graph nodes' parameters | 115 |
| Mvx2API.PluginsLoader | |
| A loader of MVX plugins | 116 |
| Mvx2API.RandomAccessGraphRunner | |
| A random-access runner of data-processing graphs | 118 |
| Mvx2API.Experimental.RendererGraphNode | |
| A graph node for rendering visual Mvx2 data | 119 |

| | |
|--|-----|
| Mvx2API.SingleFilterGraphNode | |
| A graph node with a single custom, explicitly specified, processing filter | 121 |
| Mvx2API.SourceInfo | |
| An information provider about an MVX source | 126 |
| Mvx2API.Utils | |
| An MVX utilities class | 128 |
| Mvx2API.Vec2 | |
| A structure containing 2D position data | 130 |
| Mvx2API.Vec3 | |
| A structure containing 3D position data | 130 |

Chapter 7

Namespace Documentation

7.1 Mvx2API Namespace Reference

Classes

- class [AsyncFrameAccessGraphNode](#)
A graph node for asynchronous notifications about processed MVX frames.
- class [AutoCompressorGraphNode](#)
A graph node for auto-compression of MVX data.
- class [AutoDecompressorGraphNode](#)
A graph node for auto-decompression of MVX data.
- class [AutoSequentialGraphRunner](#)
A sequential runner of data-processing graphs with automatic (synchronous/asynchronous) updates-invocation.
- class [BasicDataLayersGuids](#)
A collection of GUID constants for unique identification of simple data layers.
- class [BlockFPSGraphNode](#)
A blocking graph node with an automatized framerate-based frames-pulling capability.
- class [BlockGraphNode](#)
A graph node with a buffering and execution-blocking capabilities.
- class [BlockManualGraphNode](#)
A blocking graph node with a manual frames-pulling capability.
- struct [Col](#)
A structure containing color data.
- class [DataProfile](#)
A profile of a single data item.
- class [DataProfileEnumerator](#)
An iterator over profiles of data contained in a frame.
- class [DelegatedFrameListener](#)
A listener for asynchronous reception of frames with an external delegate.
- class [DelegatedParameterValueChangedListener](#)
A listener for changes of graph nodes' parameters with an external delegate.
- class [FilterParameterNameEnumerator](#)
An iterator over names of filter parameters of a [SingleFilterGraphNode](#).
- class [Frame](#)
A frame of data.
- class [FrameAccessGraphNode](#)

- A graph node for direct access to processed MVX frames.*

 - class [FrameAudioExtractor](#)

An extractor of audio data from frames.
 - class [FrameListener](#)

A listener for asynchronous reception of frames.
 - class [FrameMeshExtractor](#)

An extractor of mesh data from frames.
 - class [FrameMiscDataExtractor](#)

An extractor of miscellaneous data from frames.
 - class [FrameTextureExtractor](#)

An extractor of texture data from frames.
 - class [Graph](#)

A graph of data-processing nodes.
 - class [GraphBuilder](#)

A builder of data-processing graphs.
 - class [GraphNode](#)

A processing node.
 - class [GraphRunner](#)

A runner of data-processing graphs.
 - class [InjectFileDataGraphNode](#)

A graph node for injecting binary data from files to frames.
 - class [InjectMemoryDataGraphNode](#)

A graph node for injecting binary data from memory to frames.
 - class [InputEvent](#)

An input event structure.
 - class [KeyDownEvent](#)

A 'key down' event.
 - class [KeyUpEvent](#)

A 'key up' event.
 - class [ManualGraphBuilder](#)

A manual builder of data-processing graphs.
 - class [ManualLiveFrameSourceGraphNode](#)

A source graph node for manual production of MVX frames.
 - class [ManualOfflineFrameSourceGraphNode](#)

A source graph node for manual production of MVX frames.
 - class [ManualSequentialGraphRunner](#)

A sequential runner of data-processing graphs with manual updates-invocation.
 - class [MeshData](#)

A class containing data of a single mesh.
 - class [MeshSplitter](#)

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.
 - class [MouseDoubleClickEvent](#)

A 'mouse double-click' event.
 - class [MouseDownEvent](#)

A 'mouse down' event.
 - class [MouseMoveEvent](#)

A 'mouse move' event.
 - class [MouseUpEvent](#)

A 'mouse up' event.

- class [MouseWheelEvent](#)
A 'mouse wheel' event.
- class [ParameterValueChangeListener](#)
A listener for changes of graph nodes' parameters.
- class [PluginsLoader](#)
A loader of MVX plugins.
- class [RandomAccessGraphRunner](#)
A random-access runner of data-processing graphs.
- class [SingleFilterGraphNode](#)
A graph node with a single custom, explicitly specified, processing filter.
- class [SourceInfo](#)
An information provider about an MVX source.
- class [Utils](#)
An MVX utilities class.
- struct [Vec2](#)
A structure containing 2D position data.
- struct [Vec3](#)
A structure containing 3D position data.

Enumerations

- enum [MeshIndicesMode](#) { [MeshIndicesMode.MIM_PointList](#) = 0, [MeshIndicesMode.MIM_LineList](#) = 1, [MeshIndicesMode.MIM_TriangleList](#) = 2, [MeshIndicesMode.MIM_QuadList](#) = 3 }
Enumeration of indices modes.
- enum [RunnerPlaybackMode](#) { [RunnerPlaybackMode.RPM_FORWARD_ONCE](#) = 0, [RunnerPlaybackMode.RPM_FORWARD_LOOP](#) = 1, [RunnerPlaybackMode.RPM_BACKWARD_ONCE](#) = 2, [RunnerPlaybackMode.RPM_BACKWARD_LOOP](#) = 3, [RunnerPlaybackMode.RPM_PINGPONG](#) = 4, [RunnerPlaybackMode.RPM_PINGPONG_INVERSE](#) = 5, [RunnerPlaybackMode.RPM_REALTIME](#) = 255 }
An enumeration of supported MVX stream playback modes.
- enum [RunnerPlaybackState](#) { [RunnerPlaybackState.RPS_Stopped](#) = 0, [RunnerPlaybackState.RPS_Paused](#) = 1, [RunnerPlaybackState.RPS_Playing](#) = 2 }
An enumeration of runner playback states.

7.1.1 Enumeration Type Documentation

7.1.1.1 MeshIndicesMode

```
enum Mvx2API.MeshIndicesMode [strong]
```

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. |

7.1.1.2 RunnerPlaybackMode

```
enum Mvx2API.RunnerPlaybackMode [strong]
```

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. |

7.1.1.3 RunnerPlaybackState

```
enum Mvx2API.RunnerPlaybackState [strong]
```

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. |

7.2 Mvx2API.Experimental Namespace Reference**Classes**

- class [RendererGraphNode](#)

A graph node for rendering visual Mvx2 data.

Chapter 8

Class Documentation

8.1 Mvx2API.AsyncFrameAccessGraphNode Class Reference

A graph node for asynchronous notifications about processed MVX frames.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [AsyncFrameAccessGraphNode](#) ([FrameListener](#) frameListener)
A constructor.
- void [SetFrameListener](#) ([FrameListener](#) frameListener)
Sets an asynchronous frame listener to be used.

Additional Inherited Members

8.1.1 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.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 AsyncFrameAccessGraphNode()

```
Mvx2API.AsyncFrameAccessGraphNode.AsyncFrameAccessGraphNode (
    FrameListener frameListener )
```

A constructor.

Parameters

| | |
|----------------------|---------------------------------|
| <i>frameListener</i> | an asynchronous frames listener |
|----------------------|---------------------------------|

8.1.3 Member Function Documentation

8.1.3.1 SetFrameListener()

```
void Mvx2API.AsyncFrameAccessGraphNode.SetFrameListener (
    FrameListener frameListener )
```

Sets an asynchronous frame listener to be used.

Parameters

| | |
|----------------------|---------------------------------|
| <i>frameListener</i> | an asynchronous frames listener |
|----------------------|---------------------------------|

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

- `public/Mvx2API/frameaccess/AsyncFrameAccessGraphNode.cs`

8.2 Mvx2API.AutoCompressorGraphNode Class Reference

A graph node for auto-compression of MVX data.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [AutoCompressorGraphNode](#) (bool dropUncompressedInput=true)
A constructor.

Additional Inherited Members

8.2.1 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.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 AutoCompressorGraphNode()

```
Mvx2API.AutoCompressorGraphNode.AutoCompressorGraphNode (
    bool dropUncompressedInput = true )
```

A constructor.

Parameters

| | |
|------------------------------|---|
| <i>dropUncompressedInput</i> | 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.cs

8.3 Mvx2API.AutoDecompressorGraphNode Class Reference

A graph node for auto-decompression of MVX data.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [AutoDecompressorGraphNode](#) (bool dropCompressedInput=true)
A constructor.

Additional Inherited Members

8.3.1 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.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 AutoDecompressorGraphNode()

```
Mvx2API.AutoDecompressorGraphNode.AutoDecompressorGraphNode (
    bool dropCompressedInput = true )
```

A constructor.

Parameters

| | |
|----------------------------------|---|
| <code>dropCompressedInput</code> | 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.cs`

8.4 Mvx2API.AutoSequentialGraphRunner Class Reference

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

Inherits [Mvx2API.GraphRunner](#).

Public Member Functions

- [AutoSequentialGraphRunner](#) ([Graph](#) graph)
A constructor.
- bool [Play](#) ([RunnerPlaybackMode](#) playbackMode, bool blockUntilStopped=false)
Starts playback of the graph with a given playback mode.
- bool [Stop](#) ()
Invokes stopping of the graph playback.
- bool [Pause](#) ()
Pauses the graph playback.
- bool [Resume](#) ()
Resumes the graph playback.
- [RunnerPlaybackState](#) [GetPlaybackState](#) ()
Determines current playback state of the runner.
- void [SeekFrame](#) (UInt32 frameID)
Sets a frame with a given ID as the next to be processed.

Additional Inherited Members

8.4.1 Detailed Description

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

8.4.2 Constructor & Destructor Documentation

8.4.2.1 AutoSequentialGraphRunner()

```
Mvx2API.AutoSequentialGraphRunner.AutoSequentialGraphRunner (
    Graph graph )
```

A constructor.

Parameters

| | |
|--------------|----------------------------------|
| <i>graph</i> | a graph to create the runner for |
|--------------|----------------------------------|

8.4.3 Member Function Documentation

8.4.3.1 GetPlaybackState()

```
RunnerPlaybackState Mvx2API.AutoSequentialGraphRunner.GetPlaybackState ( )
```

Determines current playback state of the runner.

Returns

playback state

8.4.3.2 Pause()

```
bool Mvx2API.AutoSequentialGraphRunner.Pause ( )
```

Pauses the graph playback.

Returns

true if the graph playback successfully paused

8.4.3.3 Play()

```
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

| | |
|--------------------------|--|
| <i>playbackMode</i> | a playback mode |
| <i>blockUntilStopped</i> | an indication whether to block the call until the execution of the graph stops |

Returns

true if the graph playback successfully started

8.4.3.4 Resume()

```
bool Mvx2API.AutoSequentialGraphRunner.Resume ( )
```

Resumes the graph playback.

Returns

true if the graph playback successfully resumed

8.4.3.5 SeekFrame()

```
void Mvx2API.AutoSequentialGraphRunner.SeekFrame (
    UInt32 frameID )
```

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

Parameters

| | |
|----------------|---|
| <i>frameID</i> | an ID of the frame to be processed next |
|----------------|---|

8.4.3.6 Stop()

```
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.cs

8.5 Mvx2API.BasicDataLayersGuids Class Reference

A collection of GUID constants for unique identification of simple data layers.

Properties

- static MVCommon.Guid [AUDIO_DATA_LAYER](#) [get]
A getter of audio data layer GUID.
- static MVCommon.Guid [VERTEX_POSITIONS_DATA_LAYER](#) [get]
A getter of vertex positions data layer GUID.
- static MVCommon.Guid [VERTEX_COLORS_DATA_LAYER](#) [get]
A getter of vertex colors data layer GUID.
- static MVCommon.Guid [VERTEX_NORMALS_DATA_LAYER](#) [get]
A getter of vertex normals data layer GUID.
- static MVCommon.Guid [VERTEX_UVS_DATA_LAYER](#) [get]
A getter of vertex UVs data layer GUID.
- static MVCommon.Guid [VERTEX_INDICES_DATA_LAYER](#) [get]
A getter of vertex indices data layer GUID.
- static MVCommon.Guid [CAMERA_PARAMS_DATA_LAYER](#) [get]
A getter of camera params data layer GUID.
- static MVCommon.Guid [TRANSFORM_DATA_LAYER](#) [get]
A getter of transform data layer GUID.
- static MVCommon.Guid [SEGMENT_INFO_DATA_LAYER](#) [get]
A getter of segment info data layer GUID.
- static MVCommon.Guid [BYTEARRAY_DATA_LAYER](#) [get]
A getter of bytearray data layer GUID.
- static MVCommon.Guid [DEPTHMAP_TEXTURE_DATA_LAYER](#) [get]
A getter of depth map texture data layer GUID.
- static MVCommon.Guid [IR_TEXTURE_DATA_LAYER](#) [get]
A getter of IR texture data layer GUID.
- static MVCommon.Guid [RGB_TEXTURE_DATA_LAYER](#) [get]
A getter of RGB texture data layer GUID.
- static MVCommon.Guid [NVX_TEXTURE_DATA_LAYER](#) [get]
A getter of NVX texture data layer GUID.
- static MVCommon.Guid [NV12_TEXTURE_DATA_LAYER](#) [get]
A getter of NV12 texture data layer GUID.
- static MVCommon.Guid [NV21_TEXTURE_DATA_LAYER](#) [get]
A getter of NV21 texture data layer GUID.
- static MVCommon.Guid [DXT5YCOCG_TEXTURE_DATA_LAYER](#) [get]
A getter of DXT5YCOCG texture data layer GUID.
- static MVCommon.Guid [DXT1_TEXTURE_DATA_LAYER](#) [get]
A getter of DXT1 texture data layer GUID.
- static MVCommon.Guid [ETC2_TEXTURE_DATA_LAYER](#) [get]
A getter of ETC2 texture data layer GUID.
- static MVCommon.Guid [ASTC_TEXTURE_DATA_LAYER](#) [get]
A getter of ASTC texture data layer GUID.

8.5.1 Detailed Description

A collection of GUID constants for unique identification of simple data layers.

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

- `public/Mvx2API/data/BasicDataLayersGuids.cs`

8.6 Mvx2API.BlockFPSGraphNode Class Reference

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

Inherits [Mvx2API.BlockGraphNode](#).

Public Member Functions

- [BlockFPSGraphNode](#) (UInt32 bufferSize=3, float fps=[FPS_FROM_SOURCE](#), [FullBehaviour](#) fullBehaviour=[FullBehaviour.FB_DROP_FRAMES](#))
A constructor.
- void [SetFPS](#) (float fps)
Sets a new framerate to follow with frames-pulling.

Static Public Attributes

- const float [FPS_MAX](#) = 0.0f
A special framerate value indicating that the maximal possible framerate shall be used.
- const float [FPS_FROM_SOURCE](#) = -1.0f
A special framerate value indicating that the framerate of an open source shall be used.
- const float [FPS_FPS_HALF_FROM_SOURCE](#) = -2.0f
A special framerate value indicating that the half of the framerate of an open source shall be used.
- const float [FPS_DOUBLE_FROM_SOURCE](#) = -3.0f
A special framerate value indicating that the double of the framerate of an open source shall be used.

Additional Inherited Members

8.6.1 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.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 BlockFPSGraphNode()

```
Mvx2API.BlockFPSGraphNode.BlockFPSGraphNode (
    UInt32 bufferSize = 3,
    float fps = FPS\_FROM\_SOURCE,
    FullBehaviour fullBehaviour = FullBehaviour.FB\_DROP\_FRAMES )
```

A constructor.

Parameters

| | |
|----------------------|---|
| <i>bufferSize</i> | a size of internal frames buffer |
| <i>fps</i> | a framerate to follow with frames-pulling |
| <i>fullBehaviour</i> | an initial full-behaviour |

Exceptions

| | |
|-------------------------|--|
| <i>System.Exception</i> | raised in case the creation of the internal filter fails |
|-------------------------|--|

8.6.3 Member Function Documentation

8.6.3.1 SetFPS()

```
void Mvx2API.BlockFPSGraphNode.SetFPS (
    float fps )
```

Sets a new framerate to follow with frames-pulling.

Parameters

| | |
|------------|-----------------------|
| <i>fps</i> | a framerate to follow |
|------------|-----------------------|

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

- public/Mvx2API/graphnodes/BlockFPSGraphNode.cs

8.7 Mvx2API.BlockGraphNode Class Reference

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

Inherits [Mvx2API.GraphNode](#).

Inherited by [Mvx2API.BlockFPSGraphNode](#), and [Mvx2API.BlockManualGraphNode](#).

Public Types

- enum [FullBehaviour](#) { [FullBehaviour.FB_DROP_FRAMES](#), [FullBehaviour.FB_BLOCK_FRAMES](#) }
Enumeration of supported behaviours when the buffer of the node is full.

Public Member Functions

- void [SetFullBehaviour](#) ([FullBehaviour](#) fullBehaviour)
Sets a full-behaviour - action to perform when the internal buffer of frames becomes full.
- UInt64 [GetDroppedFramesCount](#) ()
Gets a value of internal counter of dropped frames.
- void [ResetDroppedFramesCounter](#) ()
Resets the internal counter of dropped frames to zero.

Protected Member Functions

- [BlockGraphNode](#) (IntPtr nativeObject)
A constructor.

Additional Inherited Members

8.7.1 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.

8.7.2 Member Enumeration Documentation

8.7.2.1 FullBehaviour

```
enum Mvx2API.BlockGraphNode.FullBehaviour [strong]
```

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. |

8.7.3 Constructor & Destructor Documentation

8.7.3.1 BlockGraphNode()

```
Mvx2API.BlockGraphNode.BlockGraphNode (
    IntPtr nativeObject ) [protected]
```

A constructor.

Parameters

| | |
|---------------------|----------------------------|
| <i>nativeObject</i> | a native graph node object |
|---------------------|----------------------------|

8.7.4 Member Function Documentation

8.7.4.1 GetDroppedFramesCount()

```
UInt64 Mvx2API.BlockGraphNode.GetDroppedFramesCount ( )
```

Gets a value of internal counter of dropped frames.

Returns

dropped frames count

8.7.4.2 SetFullBehaviour()

```
void Mvx2API.BlockGraphNode.SetFullBehaviour (
    FullBehaviour fullBehaviour )
```

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

Parameters

| | |
|----------------------|--------------------|
| <i>fullBehaviour</i> | a behaviour to set |
|----------------------|--------------------|

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

- public/Mvx2API/graphnodes/BlockGraphNode.cs

8.8 Mvx2API.BlockManualGraphNode Class Reference

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

Inherits [Mvx2API.BlockGraphNode](#).

Public Member Functions

- [BlockManualGraphNode](#) (UInt32 bufferSize=3, [FullBehaviour](#) fullBehaviour=[FullBehaviour.FB_DROP_FRAMES](#))
A constructor.
- void [PullNextProcessedFrame](#) ()
Releases the oldest of the buffered frames for further processing.

Additional Inherited Members

8.8.1 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.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 BlockManualGraphNode()

```
Mvx2API.BlockManualGraphNode.BlockManualGraphNode (
    UInt32 bufferSize = 3,
    FullBehaviour fullBehaviour = FullBehaviour.FB_DROP_FRAMES )
```

A constructor.

Parameters

| | |
|----------------------|----------------------------------|
| <i>bufferSize</i> | a size of internal frames buffer |
| <i>fullBehaviour</i> | an initial full-behaviour |

Exceptions

| | |
|-------------------------|--|
| <i>System.Exception</i> | raised in case the creation of the internal filter fails |
|-------------------------|--|

8.8.3 Member Function Documentation

8.8.3.1 PullNextProcessedFrame()

```
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.cs

8.9 Mvx2API.Col Struct Reference

A structure containing color data.

Public Attributes

- byte [r](#)
A red color component.
- byte [g](#)
A green color component.
- byte [b](#)
A blue color component.
- byte [a](#)
An alpha color component.

8.9.1 Detailed Description

A structure containing color data.

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

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

8.10 Mvx2API.DataProfile Class Reference

A profile of a single data item.

Inherits [NativeObjectHolder](#), and [IEquatable< DataProfile >](#).

Public Member Functions

- [DataProfile](#) (MVCommon.Guid [typeGuid](#), MVCommon.Guid [compressedTypeGuid](#), MVCommon.Guid [purposeGuid](#))
A constructor.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

Properties

- IntPtr [nativeDataProfileObject](#) [get]
A getter of the native [DataProfile](#) object.
- MVCommon.Guid [typeGuid](#) [get]
A getter of the data type guid.
- MVCommon.Guid [compressedTypeGuid](#) [get]
A getter of the compressed data type guid.
- MVCommon.Guid [purposeGuid](#) [get]
A getter of the purpose guid.

8.10.1 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).

8.10.2 Constructor & Destructor Documentation

8.10.2.1 DataProfile()

```
Mvx2API.DataProfile.DataProfile (
    MVCommon.Guid typeGuid,
    MVCommon.Guid compressedTypeGuid,
    MVCommon.Guid purposeGuid )
```

A constructor.

Parameters

| | |
|---------------------------|-----------------------------|
| <i>typeGuid</i> | a data type guid |
| <i>compressedTypeGuid</i> | a compressed data type guid |
| <i>purposeGuid</i> | a purpose guid |

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

- [public/Mvx2API/data/dataprofiles/DataProfile.cs](#)

8.11 Mvx2API.DataProfileEnumerator Class Reference

An iterator over profiles of data contained in a frame.

Inherits [IEnumerator< DataProfile >](#).

Public Member Functions

- [DataProfileEnumerator](#) (IntPtr beginIterator, IntPtr endIterator)
A constructor.

8.11.1 Detailed Description

An iterator over profiles of data contained in a frame.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 DataProfileEnumerator()

```
Mvx2API.DataProfileEnumerator.DataProfileEnumerator (
    IntPtr beginIterator,
    IntPtr endIterator )
```

A constructor.

Parameters

| | |
|----------------------|---|
| <i>beginIterator</i> | an iterator to the first data profile element |
| <i>endIterator</i> | an iterator to the last data profile element |

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

- [public/Mvx2API/data/dataprofiles/DataProfileEnumerator.cs](#)

8.12 Mvx2API.DelegatedFrameListener Class Reference

A listener for asynchronous reception of frames with an external delegate.

Inherits [Mvx2API.FrameListener](#).

Public Member Functions

- [DelegatedFrameListener](#) ([OnFrameProcessedDelegate](#) frameListenerDelegate)
A constructor.
- delegate void [OnFrameProcessedDelegate](#) ([Frame](#) frame)
A type of frame-being-processed delegates.

Protected Member Functions

- override void [OnFrameProcessed](#) ([Frame](#) frame)
A callback executed when a new frame is processed.

Additional Inherited Members

8.12.1 Detailed Description

A listener for asynchronous reception of frames with an external delegate.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 DelegatedFrameListener()

```
Mvx2API.DelegatedFrameListener.DelegatedFrameListener (
    OnFrameProcessedDelegate frameListenerDelegate )
```

A constructor.

Parameters

| | |
|------------------------------|--|
| <i>frameListenerDelegate</i> | an external delegate for reception of processed frames |
|------------------------------|--|

8.12.3 Member Function Documentation

8.12.3.1 OnFrameProcessed()

```
override void Mvx2API.DelegatedFrameListener.OnFrameProcessed (
    Frame frame ) [protected], [virtual]
```

A callback executed when a new frame is processed.

Parameters

| | |
|--------------|--|
| <i>frame</i> | a new frame (it is a responsibility of the client to dispose it) |
|--------------|--|

Implements [Mvx2API.FrameListener](#).

8.12.3.2 OnFrameProcessedDelegate()

```
delegate void Mvx2API.DelegatedFrameListener.OnFrameProcessedDelegate (
    Frame frame )
```

A type of frame-being-processed delegates.

Parameters

| | |
|--------------|--|
| <i>frame</i> | 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/DelegatedFrameListener.cs

8.13 Mvx2API.DelegatedParameterValueChangeListener Class Reference

A listener for changes of graph nodes' parameters with an external delegate.

Inherits [Mvx2API.ParameterValueChangeListener](#).

Public Member Functions

- [DelegatedParameterValueChangeListener](#) ([OnParameterValueChangedDelegate](#) parameterValue↔
ChangeListenerDelegate)
A constructor.
- delegate void [OnParameterValueChangedDelegate](#) ([GraphNode](#) graphNode, MVCommon.String parameter↔
Name, MVCommon.String parameterValueStr)
A type of parameter-value-changed delegates.

Protected Member Functions

- override void [OnParameterValueChanged](#) ([GraphNode](#) graphNode, MVCommon.String parameterName,
MVCommon.String parameterValueStr)
A callback executed when a parameter of a graph node changes its value.

Additional Inherited Members

8.13.1 Detailed Description

A listener for changes of graph nodes' parameters with an external delegate.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 DelegatedParameterValueChangedListener()

```
Mvx2API.DelegatedParameterValueChangedListener.DelegatedParameterValueChangedListener (
    OnParameterValueChangedDelegate parameterValueChangedListenerDelegate )
```

A constructor.

Parameters

| | |
|--|---|
| <i>parameterValueChangedListenerDelegate</i> | an external delegate for handling of changed parameter values |
|--|---|

8.13.3 Member Function Documentation

8.13.3.1 OnParameterValueChanged()

```
override void Mvx2API.DelegatedParameterValueChangedListener.OnParameterValueChanged (
    GraphNode graphNode,
    MVCommon.String parameterName,
    MVCommon.String parameterValueStr ) [protected], [virtual]
```

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

Parameters

| | |
|--------------------------|---|
| <i>graphNode</i> | a graph node containing the changed parameter |
| <i>parameterName</i> | name of the changed parameter |
| <i>parameterValueStr</i> | parameter's new value in a string form |

Implements [Mvx2API.ParameterValueChangedListener](#).

8.13.3.2 OnParameterValueChangedDelegate()

```
delegate void Mvx2API.DelegatedParameterValueChangedListener.OnParameterValueChangedDelegate (
    GraphNode graphNode,
    MVCommon.String parameterName,
    MVCommon.String parameterValueStr )
```

A type of parameter-value-changed delegates.

Parameters

| | |
|--------------------------|---|
| <i>graphNode</i> | a graph node containing the changed parameter |
| <i>parameterName</i> | name of the changed parameter |
| <i>parameterValueStr</i> | parameter's new value in a string form |

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

- public/Mvx2API/graphnodes/DelegatedParameterValueChangedListener.cs

8.14 Mvx2API.FilterParameterNameEnumerator Class Reference

An iterator over names of filter parameters of a [SingleFilterGraphNode](#).

Inherits `IEnumerator< MVCommon.String >`.

Public Member Functions

- [FilterParameterNameEnumerator](#) (IntPtr beginIterator, IntPtr endIterator)
A constructor.

8.14.1 Detailed Description

An iterator over names of filter parameters of a [SingleFilterGraphNode](#).

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.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 FilterParameterNameEnumerator()

```
Mvx2API.FilterParameterNameEnumerator.FilterParameterNameEnumerator (
    IntPtr beginIterator,
    IntPtr endIterator )
```

A constructor.

Parameters

| | |
|----------------------|--|
| <i>beginIterator</i> | an iterator to the first element of the filter parameter names collection |
| <i>endIterator</i> | an iterator behing the last element of the filter parameter names collection |

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

- public/Mvx2API/filters/FilterParameterNameEnumerator.cs

8.15 Mvx2API.Frame Class Reference

A frame of data.

Inherits NativeObjectHolder.

Public Member Functions

- [Frame](#) (IntPtr nativeObject)
A constructor.
- UInt32 [GetNumStreams](#) ()
Returns streams count of the frame.
- bool [ActivateStreamWithIndex](#) (UInt32 activeStreamIndex)
Sets a stream of the frame to be active.
- UInt32 [GetActiveStreamIndex](#) ()
Returns index of the currently active stream of the frame.
- UInt16 [GetStreamId](#) ()
Returns ID of the currently active stream of the frame.
- UInt32 [GetStreamAtomNr](#) ()
Returns the atom number in the currently active stream of the frame.
- UInt64 [GetStreamAtomTimestamp](#) ()
Returns the atom timestamp in the currently active stream of the frame.
- bool [StreamContainsDataLayer](#) (MVCommon.Guid dataLayerGuid, bool checkCompressedDataLayers↵ Too=true)
Checks whether the currently active stream contains a data layer with a given guid.
- bool [StreamContainsDataLayer](#) (MVCommon.Guid dataLayerGuid, MVCommon.Guid purposeGuid, bool checkCompressedDataLayersToo=true)
Checks whether the currently active stream contains a data layer with a given guid.
- [DataProfileEnumerator CreateDataProfilesEnumerator](#) ()
Creates an enumerator over data profile entries of the active stream.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

Properties

- IntPtr [nativeFrameObject](#) [get]
A getter of the native frame object.

8.15.1 Detailed Description

A frame of data.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 Frame()

```
Mvx2API.Frame.Frame (
    IntPtr nativeObject )
```

A constructor.

Parameters

| | |
|---------------------|---------------------------|
| <i>nativeObject</i> | a native MVX frame object |
|---------------------|---------------------------|

8.15.3 Member Function Documentation

8.15.3.1 ActivateStreamWithIndex()

```
bool Mvx2API.Frame.ActivateStreamWithIndex (
    UInt32 activeStreamIndex )
```

Sets a stream of the frame to be active.

Parameters

| | |
|--------------------------|------------------------------------|
| <i>activeStreamIndex</i> | an index of the stream to activate |
|--------------------------|------------------------------------|

Returns

true if the stream was successfully activated

8.15.3.2 CreateDataProfilesEnumerator()

```
DataProfileEnumerator Mvx2API.Frame.CreateDataProfilesEnumerator ( )
```

Creates an enumerator over data profile entries of the active stream.

Returns

a new enumerator

8.15.3.3 GetActiveStreamIndex()

```
UInt32 Mvx2API.Frame.GetActiveStreamIndex ( )
```

Returns index of the currently active stream of the frame.

Returns

currently active stream's index

8.15.3.4 GetNumStreams()

```
UInt32 Mvx2API.Frame.GetNumStreams ( )
```

Returns streams count of the frame.

Returns

streams count

8.15.3.5 GetStreamAtomNr()

```
UInt32 Mvx2API.Frame.GetStreamAtomNr ( )
```

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

Returns

atom number in the currently active stream

8.15.3.6 GetStreamAtomTimestamp()

```
UInt64 Mvx2API.Frame.GetStreamAtomTimestamp ( )
```

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

Returns

atom timestamp in the currently active stream

8.15.3.7 GetStreamId()

```
UInt16 Mvx2API.Frame.GetStreamId ( )
```

Returns ID of the currently active stream of the frame.

Returns

currently active stream's ID

8.15.3.8 StreamContainsDataLayer() [1/2]

```
bool Mvx2API.Frame.StreamContainsDataLayer (
    MVCommon.Guid dataLayerGuid,
    bool checkCompressedDataLayersToo = true )
```

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

Parameters

| | |
|-------------------------------------|--|
| <i>dataLayerGuid</i> | a guid of the data layer to check |
| <i>checkCompressedDataLayersToo</i> | 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

8.15.3.9 StreamContainsDataLayer() [2/2]

```
bool Mvx2API.Frame.StreamContainsDataLayer (
    MVCommon.Guid dataLayerGuid,
```

```
MVCommon.Guid purposeGuid,  
bool checkCompressedDataLayersToo = true )
```

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

Parameters

| | |
|-------------------------------------|--|
| <i>dataLayerGuid</i> | a guid of the data layer to check |
| <i>purposeGuid</i> | a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid) |
| <i>checkCompressedDataLayersToo</i> | 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.cs

8.16 Mvx2API.FrameAccessGraphNode Class Reference

A graph node for direct access to processed MVX frames.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [FrameAccessGraphNode](#) ()
A constructor.
- [Frame GetRecentProcessedFrame](#) ()
Returns the most recent frame processed by a containing graph.

Additional Inherited Members

8.16.1 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.

8.16.2 Member Function Documentation

8.16.2.1 GetRecentProcessedFrame()

```
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.cs

8.17 Mvx2API.FrameAudioExtractor Class Reference

An extractor of audio data from frames.

Static Public Member Functions

- static bool [GetAudioSamplingInfo](#) ([Frame](#) frame, out UInt32 numChannels, out UInt32 bitsPerSample, out UInt32 numSamplesPerSec)
Returns a frame's audio sampling information.
- static bool [GetAudioSamplingInfo](#) ([Frame](#) frame, out UInt32 numChannels, out UInt32 bitsPerSample, out UInt32 numSamplesPerSec, MVCommon.Guid purposeGuid)
Returns a frame's audio sampling information.
- static UInt32 [GetPCMDDataOffset](#) ([Frame](#) frame)
Returns a frame's audio pulse-code modulation (PCM) data offset.
- static UInt32 [GetPCMDDataOffset](#) ([Frame](#) frame, MVCommon.Guid purposeGuid)
Returns a frame's audio pulse-code modulation (PCM) data offset.
- static UInt32 [GetPCMDDataSize](#) ([Frame](#) frame)
Returns a frame's audio pulse-code modulation (PCM) data size (in bytes).
- static UInt32 [GetPCMDDataSize](#) ([Frame](#) frame, MVCommon.Guid purposeGuid)
Returns a frame's audio pulse-code modulation (PCM) data size (in bytes).
- static IntPtr [GetPCMDData](#) ([Frame](#) frame)
A getter of the raw pointer to audio pulse-code modulation (PCM) data.
- static IntPtr [GetPCMDData](#) ([Frame](#) frame, MVCommon.Guid purposeGuid)
A getter of the raw pointer to audio pulse-code modulation (PCM) data.
- static bool [CopyPCMDData](#) ([Frame](#) frame, byte[] targetData)
Copies a frame's audio pulse-code modulation (PCM) data.
- static bool [CopyPCMDData](#) ([Frame](#) frame, byte[] targetData, MVCommon.Guid purposeGuid)
Copies a frame's audio pulse-code modulation (PCM) data.
- static bool [CopyPCMDDataRaw](#) ([Frame](#) frame, IntPtr targetData)
Copies a frame's audio pulse-code modulation (PCM) data.
- static bool [CopyPCMDDataRaw](#) ([Frame](#) frame, IntPtr targetData, MVCommon.Guid purposeGuid)
Copies a frame's audio pulse-code modulation (PCM) data.

8.17.1 Detailed Description

An extractor of audio data from frames.

8.17.2 Member Function Documentation

8.17.2.1 CopyPCMDData() [1/2]

```
static bool Mvx2API.FrameAudioExtractor.CopyPCMDData (
    Frame frame,
    byte[] targetData ) [static]
```

Copies a frame's audio pulse-code modulation (PCM) data.

Parameters

| | |
|-------------------|---|
| <i>frame</i> | a frame |
| <i>targetData</i> | a target PCM data array (must be pre-allocated with (PCM data size) elements) |

Returns

true if the PCM data were successfully copied

8.17.2.2 CopyPCMDData() [2/2]

```
static bool Mvx2API.FrameAudioExtractor.CopyPCMDData (
    Frame frame,
    byte[] targetData,
    MVCommon.Guid purposeGuid ) [static]
```

Copies a frame's audio pulse-code modulation (PCM) data.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>targetData</i> | a target PCM data array (must be pre-allocated with (PCM data size) elements) |
| <i>purposeGuid</i> | 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

8.17.2.3 CopyPCMDDataRaw() [1/2]

```
static bool Mvx2API.FrameAudioExtractor.CopyPCMDDataRaw (
    Frame frame,
    IntPtr targetData ) [static]
```

Copies a frame's audio pulse-code modulation (PCM) data.

Parameters

| | |
|-------------------|--|
| <i>frame</i> | a frame |
| <i>targetData</i> | a target PCM data array (must be pre-allocated with (PCM data size) bytes) |

Returns

true if the PCM data were successfully copied

8.17.2.4 CopyPCMDDataRaw() [2/2]

```
static bool Mvx2API.FrameAudioExtractor.CopyPCMDDataRaw (
    Frame frame,
    IntPtr targetData,
    MVCommon.Guid purposeGuid ) [static]
```

Copies a frame's audio pulse-code modulation (PCM) data.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>targetData</i> | a target PCM data array (must be pre-allocated with (PCM data size) bytes) |
| <i>purposeGuid</i> | 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

8.17.2.5 GetAudioSamplingInfo() [1/2]

```
static bool Mvx2API.FrameAudioExtractor.GetAudioSamplingInfo (
    Frame frame,
    out UInt32 numChannels,
    out UInt32 bitsPerSample,
    out UInt32 numSamplesPerSec ) [static]
```

Returns a frame's audio sampling information.

Parameters

| | |
|-------------------------|--|
| <i>frame</i> | a frame |
| <i>numChannels</i> | an outputted count of audio channels |
| <i>bitsPerSample</i> | an outputted bits count per sample |
| <i>numSamplesPerSec</i> | an outputted count of samples per second |

Returns

true if the audio sampling information were successfully retrieved

8.17.2.6 GetAudioSamplingInfo() [2/2]

```
static bool Mvx2API.FrameAudioExtractor.GetAudioSamplingInfo (
    Frame frame,
    out UInt32 numChannels,
    out UInt32 bitsPerSample,
    out UInt32 numSamplesPerSec,
    MVCommon.Guid purposeGuid ) [static]
```

Returns a frame's audio sampling information.

Parameters

| | |
|-------------------------|--|
| <i>frame</i> | a frame |
| <i>numChannels</i> | an outputted count of audio channels |
| <i>bitsPerSample</i> | an outputted bits count per sample |
| <i>numSamplesPerSec</i> | an outputted count of samples per second |
| <i>purposeGuid</i> | 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

8.17.2.7 GetPCMDData() [1/2]

```
static IntPtr Mvx2API.FrameAudioExtractor.GetPCMDData (
    Frame frame ) [static]
```

A getter of the raw pointer to audio pulse-code modulation (PCM) data.

Parameters

| | |
|--------------|---------|
| <i>frame</i> | a frame |
|--------------|---------|

Returns

PCM data

8.17.2.8 GetPCMDData() [2/2]

```
static IntPtr Mvx2API.FrameAudioExtractor.GetPCMDData (
    Frame frame,
    MVCommon.Guid purposeGuid ) [static]
```

A getter of the raw pointer to audio pulse-code modulation (PCM) data.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>purposeGuid</i> | a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

PCM data

8.17.2.9 GetPCMDDataOffset() [1/2]

```
static UInt32 Mvx2API.FrameAudioExtractor.GetPCMDDataOffset (
    Frame frame ) [static]
```

Returns a frame's audio pulse-code modulation (PCM) data offset.

Parameters

| | |
|--------------|---------|
| <i>frame</i> | a frame |
|--------------|---------|

Returns

PCM data offset

8.17.2.10 GetPCMDDataOffset() [2/2]

```
static UInt32 Mvx2API.FrameAudioExtractor.GetPCMDDataOffset (
    Frame frame,
    MVCommon.Guid purposeGuid ) [static]
```

Returns a frame's audio pulse-code modulation (PCM) data offset.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>purposeGuid</i> | a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

PCM data offset

8.17.2.11 GetPCMDDataSize() [1/2]

```
static UInt32 Mvx2API.FrameAudioExtractor.GetPCMDDataSize (  
    Frame frame ) [static]
```

Returns a frame's audio pulse-code modulation (PCM) data size (in bytes).

Parameters

| | |
|--------------|---------|
| <i>frame</i> | a frame |
|--------------|---------|

Returns

PCM data size

8.17.2.12 GetPCMDDataSize() [2/2]

```
static UInt32 Mvx2API.FrameAudioExtractor.GetPCMDDataSize (  
    Frame frame,  
    MVCommon.Guid purposeGuid ) [static]
```

Returns a frame's audio pulse-code modulation (PCM) data size (in bytes).

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>purposeGuid</i> | a purpose guid of audio data to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

PCM data size

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

- public/Mvx2API/frameaccess/extractors/FrameAudioExtractor.cs

8.18 Mvx2API.FrameListener Class Reference

A listener for asynchronous reception of frames.

Inherits `NativeObjectHolder`.

Inherited by [Mvx2API.DelegatedFrameListener](#).

Public Member Functions

- [FrameListener](#) ()
A constructor.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.
- abstract void [OnFrameProcessed](#) ([Frame](#) frame)
A callback executed when a new frame is processed.

Properties

- `IntPtr` [nativeFrameListenerObject](#) [get]
A getter of the native frame listener object.

8.18.1 Detailed Description

A listener for asynchronous reception of frames.

8.18.2 Member Function Documentation

8.18.2.1 OnFrameProcessed()

```
abstract void Mvx2API.FrameListener.OnFrameProcessed (
    Frame frame ) [protected], [pure virtual]
```

A callback executed when a new frame is processed.

Parameters

| | |
|--------------|--|
| <i>frame</i> | a new frame (it is a responsibility of the client to dispose it) |
|--------------|--|

Implemented in [Mvx2API.DelegatedFrameListener](#).

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

- public/Mvx2API/frameaccess/FrameListener.cs

8.19 Mvx2API.FrameMeshExtractor Class Reference

An extractor of mesh data from frames.

Static Public Member Functions

- static [MeshData](#) [GetMeshData](#) ([Frame](#) frame)
Returns a frame's mesh data.
- static [MeshData](#) [GetMeshData](#) ([Frame](#) frame, MVCommon.Guid purposeGuid)
Returns a frame's mesh data.

8.19.1 Detailed Description

An extractor of mesh data from frames.

8.19.2 Member Function Documentation

8.19.2.1 GetMeshData() [1/2]

```
static MeshData Mvx2API.FrameMeshExtractor.GetMeshData (  
    Frame frame ) [static]
```

Returns a frame's mesh data.

Parameters

| | |
|--------------|---------|
| <i>frame</i> | a frame |
|--------------|---------|

Returns

frame's mesh

8.19.2.2 GetMeshData() [2/2]

```
static MeshData Mvx2API.FrameMeshExtractor.GetMeshData (
    Frame frame,
    MVCommon.Guid purposeGuid ) [static]
```

Returns a frame's mesh data.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>purposeGuid</i> | a purpose guid of mesh data to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

frame's mesh

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

- public/Mvx2API/frameaccess/extractors/FrameMeshExtractor.cs

8.20 Mvx2API.FrameMiscDataExtractor Class Reference

An extractor of miscellaneous data from frames.

Static Public Member Functions

- static bool [GetColorCameraParams](#) (Frame frame, MVCommon.CameraParams cameraParams)
Gets color camera parameters of a frame.
- static bool [GetColorCameraParams](#) (Frame frame, MVCommon.CameraParams cameraParams, MV↔Common.Guid purposeGuid)
Gets color camera parameters of a frame.
- static bool [GetIRCameraParams](#) (Frame frame, MVCommon.CameraParams cameraParams)
Gets IR camera parameters of a frame.
- static bool [GetIRCameraParams](#) (Frame frame, MVCommon.CameraParams cameraParams, MV↔Common.Guid purposeGuid)
Gets IR camera parameters of a frame.
- static bool [GetTransform](#) (Frame frame, MVCommon.Matrix4x4f transform)
Gets transformation matrix of a frame.
- static bool [GetTransform](#) (Frame frame, MVCommon.Matrix4x4f transform, MVCommon.Guid purposeGuid)
Gets transformation matrix of a frame.
- static bool [GetSegmentID](#) (Frame frame, out UInt16 segmentID)
Gets an ID of a segment a frame belongs to.
- static bool [GetSegmentID](#) (Frame frame, out UInt16 segmentID, MVCommon.Guid purposeGuid)
Gets an ID of a segment a frame belongs to.
- static bool [GetByteArrayData](#) (Frame frame, MVCommon.ByteArray byteArray)
Gets a bytearray data of a frame.
- static bool [GetByteArrayData](#) (Frame frame, MVCommon.ByteArray byteArray, MVCommon.Guid purpose↔Guid)
Gets a bytearray data of a frame.

8.20.1 Detailed Description

An extractor of miscellaneous data from frames.

8.20.2 Member Function Documentation

8.20.2.1 GetByteArrayData() [1/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetByteArrayData (
    Frame frame,
    MVCommon.ByteArray byteArray ) [static]
```

Gets a bytearray data of a frame.

Parameters

| | |
|------------------|---|
| <i>frame</i> | a frame |
| <i>byteArray</i> | a target to store the bytearray data in |

Returns

true if the frame contains bytearray data and it was successfully extracted

8.20.2.2 GetByteArrayData() [2/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetByteArrayData (
    Frame frame,
    MVCommon.ByteArray byteArray,
    MVCommon.Guid purposeGuid ) [static]
```

Gets a bytearray data of a frame.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>byteArray</i> | a target to store the bytearray data in |
| <i>purposeGuid</i> | 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

8.20.2.3 GetColorCameraParams() [1/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetColorCameraParams (
    Frame frame,
    MVCommon.CameraParams cameraParams ) [static]
```

Gets color camera parameters of a frame.

Parameters

| | |
|---------------------|--|
| <i>frame</i> | a frame |
| <i>cameraParams</i> | a target to store the camera parameters in |

Returns

true if the frame contains color camera parameters data and they were successfully extracted

8.20.2.4 GetColorCameraParams() [2/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetColorCameraParams (
    Frame frame,
    MVCommon.CameraParams cameraParams,
    MVCommon.Guid purposeGuid ) [static]
```

Gets color camera parameters of a frame.

Parameters

| | |
|---------------------|--|
| <i>frame</i> | a frame |
| <i>cameraParams</i> | a target to store the camera parameters in |
| <i>purposeGuid</i> | 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

8.20.2.5 GetIRCameraParams() [1/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetIRCameraParams (
    Frame frame,
    MVCommon.CameraParams cameraParams ) [static]
```

Gets IR camera parameters of a frame.

Parameters

| | |
|---------------------|--|
| <i>frame</i> | a frame |
| <i>cameraParams</i> | a target to store the camera parameters in |

Returns

true if the frame contains IR camera parameters data and they were successfully extracted

8.20.2.6 GetIRCameraParams() [2/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetIRCameraParams (  
    Frame frame,  
    MVCommon.CameraParams cameraParams,  
    MVCommon.Guid purposeGuid ) [static]
```

Gets IR camera parameters of a frame.

Parameters

| | |
|---------------------|--|
| <i>frame</i> | a frame |
| <i>cameraParams</i> | a target to store the camera parameters in |
| <i>purposeGuid</i> | 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

8.20.2.7 GetSegmentID() [1/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetSegmentID (  
    Frame frame,  
    out UInt16 segmentID ) [static]
```

Gets an ID of a segment a frame belongs to.

Parameters

| | |
|------------------|-------------------------------------|
| <i>frame</i> | a frame |
| <i>segmentID</i> | a target to store the segment ID in |

Returns

true if the frame contains segment information data and it was successfully extracted

8.20.2.8 GetSegmentID() [2/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetSegmentID (
    Frame frame,
    out UInt16 segmentID,
    MVCommon.Guid purposeGuid ) [static]
```

Gets an ID of a segment a frame belongs to.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>segmentID</i> | a target to store the segment ID in |
| <i>purposeGuid</i> | 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

8.20.2.9 GetTransform() [1/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetTransform (
    Frame frame,
    MVCommon.Matrix4x4f transform ) [static]
```

Gets transformation matrix of a frame.

Parameters

| | |
|------------------|--|
| <i>frame</i> | a frame |
| <i>transform</i> | a target to store the transformation matrix in |

Returns

true if the frame contains transformation data and it was successfully extracted

8.20.2.10 GetTransform() [2/2]

```
static bool Mvx2API.FrameMiscDataExtractor.GetTransform (
    Frame frame,
    MVCommon.Matrix4x4f transform,
    MVCommon.Guid purposeGuid ) [static]
```

Gets transformation matrix of a frame.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>transform</i> | a target to store the transformation matrix in |
| <i>purposeGuid</i> | 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

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

- public/Mvx2API/frameaccess/extractors/FrameMiscDataExtractor.cs

8.21 Mvx2API.FrameTextureExtractor Class Reference

An extractor of texture data from frames.

Public Types

- enum [TextureType](#) {
[TextureType.TT_DEPTH](#) = 0, [TextureType.TT_IR](#) = 1, [TextureType.TT_RGB](#) = 2, [TextureType.TT_NVX](#) = 3,
[TextureType.TT_DXT5YCOCG](#) = 4, [TextureType.TT_DXT1](#) = 5, [TextureType.TT_ETC2](#) = 6, [TextureType.TT_ASTC](#)
= 7,
[TextureType.TT_NV12](#) = 8, [TextureType.TT_NV21](#) = 9 }
An enumeration of texture types.

Static Public Member Functions

- static bool [GetTextureResolution](#) ([Frame](#) frame, [TextureType](#) textureType, out UInt16 width, out UInt16 height)
Returns resolution of a frame's texture.
- static bool [GetTextureResolution](#) ([Frame](#) frame, [TextureType](#) textureType, out UInt16 width, out UInt16 height, MVCommon.Guid purposeGuid)
Returns resolution of a frame's texture.
- static UInt32 [GetTextureDataSizeInBytes](#) ([Frame](#) frame, [TextureType](#) textureType)
Returns size (in bytes) of a frame's texture data.
- static UInt32 [GetTextureDataSizeInBytes](#) ([Frame](#) frame, [TextureType](#) textureType, MVCommon.Guid purposeGuid)
Returns size (in bytes) of a frame's texture data.
- static IntPtr [GetTextureData](#) ([Frame](#) frame, [TextureType](#) textureType)
Returns raw pointer to the texture data owned by a frame.
- static IntPtr [GetTextureData](#) ([Frame](#) frame, [TextureType](#) textureType, MVCommon.Guid purposeGuid)
Returns raw pointer to the texture data owned by a frame.
- static bool [CopyTextureData](#) ([Frame](#) frame, [TextureType](#) textureType, byte[] targetData)
Copies a frame's texture data.
- static bool [CopyTextureData](#) ([Frame](#) frame, [TextureType](#) textureType, byte[] targetData, MVCommon.Guid purposeGuid)
Copies a frame's texture data.
- static bool [CopyTextureDataRow](#) ([Frame](#) frame, [TextureType](#) textureType, IntPtr targetData)
Copies a frame's texture data.
- static bool [CopyTextureDataRow](#) ([Frame](#) frame, [TextureType](#) textureType, IntPtr targetData, MVCommon.Guid purposeGuid)
Copies a frame's texture data.

8.21.1 Detailed Description

An extractor of texture data from frames.

8.21.2 Member Enumeration Documentation

8.21.2.1 TextureType

```
enum Mvx2API.FrameTextureExtractor.TextureType [strong]
```

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. |

8.21.3 Member Function Documentation

8.21.3.1 CopyTextureData() [1/2]

```
static bool Mvx2API.FrameTextureExtractor.CopyTextureData (
    Frame frame,
    TextureType textureType,
    byte[] targetData ) [static]
```

Copies a frame's texture data.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>targetData</i> | an outputted texture data array (must be pre-allocated with (texture data size) elements) |

Returns

true if the texture data were successfully copied

8.21.3.2 CopyTextureData() [2/2]

```
static bool Mvx2API.FrameTextureExtractor.CopyTextureData (
    Frame frame,
    TextureType textureType,
    byte[] targetData,
    MVCommon.Guid purposeGuid ) [static]
```

Copies a frame's texture data.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>targetData</i> | an outputted texture data array (must be pre-allocated with (texture data size) elements) |
| <i>purposeGuid</i> | a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

true if the texture data were successfully copied

8.21.3.3 CopyTextureDataRow() [1/2]

```
static bool Mvx2API.FrameTextureExtractor.CopyTextureDataRow (
    Frame frame,
    TextureType textureType,
    IntPtr targetData ) [static]
```

Copies a frame's texture data.

Parameters

| | |
|--------------------|--|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>targetData</i> | an outputted texture data array (must be pre-allocated with (texture data size) bytes) |

Returns

true if the texture data were successfully copied

8.21.3.4 CopyTextureDataRow() [2/2]

```
static bool Mvx2API.FrameTextureExtractor.CopyTextureDataRow (
    Frame frame,
    TextureType textureType,
    IntPtr targetData,
    MVCommon.Guid purposeGuid ) [static]
```

Copies a frame's texture data.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>targetData</i> | an outputted texture data array (must be pre-allocated with (texture data size) bytes) |
| <i>purposeGuid</i> | a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

true if the texture data were successfully copied

8.21.3.5 GetTextureData() [1/2]

```
static IntPtr Mvx2API.FrameTextureExtractor.GetTextureData (
    Frame frame,
    TextureType textureType ) [static]
```

Returns raw pointer to the texture data owned by a frame.

Parameters

| | |
|--------------------|----------------------------------|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |

Returns

texture data

8.21.3.6 GetTextureData() [2/2]

```
static IntPtr Mvx2API.FrameTextureExtractor.GetTextureData (
    Frame frame,
    TextureType textureType,
    MVCommon.Guid purposeGuid ) [static]
```

Returns raw pointer to the texture data owned by a frame.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>purposeGuid</i> | a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

texture data

8.21.3.7 GetTextureDataSizeInBytes() [1/2]

```
static UInt32 Mvx2API.FrameTextureExtractor.GetTextureDataSizeInBytes (
    Frame frame,
    TextureType textureType ) [static]
```

Returns size (in bytes) of a frame's texture data.

Parameters

| | |
|--------------------|----------------------------------|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |

Returns

texture data size

8.21.3.8 GetTextureDataSizeInBytes() [2/2]

```
static UInt32 Mvx2API.FrameTextureExtractor.GetTextureDataSizeInBytes (
    Frame frame,
    TextureType textureType,
    MVCommon.Guid purposeGuid ) [static]
```

Returns size (in bytes) of a frame's texture data.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>purposeGuid</i> | a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

texture data size

8.21.3.9 GetTextureResolution() [1/2]

```
static bool Mvx2API.FrameTextureExtractor.GetTextureResolution (
    Frame frame,
    TextureType textureType,
    out UInt16 width,
    out UInt16 height ) [static]
```

Returns resolution of a frame's texture.

Parameters

| | |
|--------------------|------------------------------------|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>width</i> | an outputted width of the texture |
| <i>height</i> | an outputted height of the texture |

Returns

true if the texture resolution was successfully retrieved

8.21.3.10 GetTextureResolution() [2/2]

```
static bool Mvx2API.FrameTextureExtractor.GetTextureResolution (
    Frame frame,
    TextureType textureType,
    out UInt16 width,
    out UInt16 height,
    MVCommon.Guid purposeGuid ) [static]
```

Returns resolution of a frame's texture.

Parameters

| | |
|--------------------|---|
| <i>frame</i> | a frame |
| <i>textureType</i> | a type of the texture to extract |
| <i>width</i> | an outputted width of the texture |
| <i>height</i> | an outputted height of the texture |
| <i>purposeGuid</i> | a purpose guid of texture to extract (Guid::Nil() is interpreted as 'any' purpose guid) |

Returns

true if the texture resolution was successfully retrieved

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

- public/Mvx2API/frameaccess/extractors/FrameTextureExtractor.cs

8.22 Mvx2API.Graph Class Reference

A graph of data-processing nodes.

Inherits NativeObjectHolder.

Public Member Functions

- bool [Reinitialize](#) ()
Reinitializes the graph.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

8.22.1 Detailed Description

A graph of data-processing nodes.

8.22.2 Member Function Documentation

8.22.2.1 Reinitialize()

```
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.cs

8.23 Mvx2API.GraphBuilder Class Reference

A builder of data-processing graphs.

Inherits NativeObjectHolder.

Inherited by [Mvx2API.ManualGraphBuilder](#).

Public Member Functions

- [Graph CompileGraphAndReset](#) ()
Compiles a graph being built and resets the builder for another graph to be built.
- void [Reset](#) ()
Resets the builder by removing all already appended graph nodes.
- bool [Refresh](#) ()
Refreshes the builder.
- bool [ContainsDataProfile](#) (MVCommon.Guid dataLayerGuid, MVCommon.Guid purposeGuid, bool checkCompressedDataLayersToo=true)
Checks whether the graph being built in its current state contains a data profile with a given guid.
- [DataProfileEnumerator CreateDataProfilesEnumerator](#) ()
Creates an enumerator over data profile entries of the graph being built in its current state.

Protected Member Functions

- [GraphBuilder](#) (IntPtr nativeObject)
A constructor.
- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

8.23.1 Detailed Description

A builder of data-processing graphs.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 GraphBuilder()

```
Mvx2API.GraphBuilder.GraphBuilder (
    IntPtr nativeObject ) [protected]
```

A constructor.

Parameters

| | |
|---------------------|-------------------------------|
| <i>nativeObject</i> | a native graph builder object |
|---------------------|-------------------------------|

8.23.3 Member Function Documentation

8.23.3.1 CompileGraphAndReset()

```
Graph Mvx2API.GraphBuilder.CompileGraphAndReset ( )
```

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 null if the graph reinitialization fails

8.23.3.2 ContainsDataProfile()

```
bool Mvx2API.GraphBuilder.ContainsDataProfile (
    MVCommon.Guid dataLayerGuid,
    MVCommon.Guid purposeGuid,
    bool checkCompressedDataLayersToo = true )
```

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

Parameters

| | |
|-------------------------------------|--|
| <i>dataLayerGuid</i> | a guid of the data layer to check |
| <i>purposeGuid</i> | a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid) |
| <i>checkCompressedDataLayersToo</i> | 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

8.23.3.3 CreateDataProfilesEnumerator()

`DataProfileEnumerator` `Mvx2API.GraphBuilder.CreateDataProfilesEnumerator ()`

Creates an enumerator over data profile entries of the graph being built in its current state.

Returns

a new enumerator

8.23.3.4 Refresh()

`bool` `Mvx2API.GraphBuilder.Refresh ()`

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

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

- `public/Mvx2API/core/GraphBuilder.cs`

8.24 Mvx2API.GraphNode Class Reference

A processing node.

Inherits `NativeObjectHolder`.

Inherited by `Mvx2API.AsyncFrameAccessGraphNode`, `Mvx2API.AutoCompressorGraphNode`, `Mvx2API.AutoDecompressorGraphNode`, `Mvx2API.BlockGraphNode`, `Mvx2API.Experimental.RendererGraphNode`, `Mvx2API.FrameAccessGraphNode`, `Mvx2API.InjectFileDataGraphNode`, `Mvx2API.InjectMemoryDataGraphNode`, `Mvx2API.ManualLiveFrameSourceGraphNode`, `Mvx2API.ManualOfflineFrameSourceGraphNode`, and `Mvx2API.SingleFilterGraphNode`.

Static Public Member Functions

- static `GraphNode` `NativeObjectToGraphNode` (`IntPtr` `nativeGraphNodeObject`)
Tries to get managed `GraphNode` instance which wraps a native graph node object.

Protected Member Functions

- `GraphNode` (`IntPtr` `nativeObject`)
A constructor.
- override void `DestroyNativeObject` ()
Destroys the native object in a customized way.

8.24.1 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.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 GraphNode()

```
Mvx2API.GraphNode.GraphNode (
    IntPtr nativeObject ) [protected]
```

A constructor.

Parameters

| | |
|---------------------|--------------------------|
| <i>nativeObject</i> | native graph node object |
|---------------------|--------------------------|

8.24.3 Member Function Documentation

8.24.3.1 NativeObjectToGraphNode()

```
static GraphNode Mvx2API.GraphNode.NativeObjectToGraphNode (
    IntPtr nativeGraphNodeObject ) [static]
```

Tries to get managed [GraphNode](#) instance which wraps a native graph node object.

Parameters

| | |
|------------------------------|---|
| <i>nativeGraphNodeObject</i> | a native graph node object to find the managed GraphNode instance for |
|------------------------------|---|

Returns

a [GraphNode](#) instance or null if there is no [GraphNode](#) instance for the given native object

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

- `public/Mvx2API/core/GraphNode.cs`

8.25 Mvx2API.GraphRunner Class Reference

A runner of data-processing graphs.

Inherits [NativeObjectHolder](#).

Inherited by [Mvx2API.AutoSequentialGraphRunner](#), [Mvx2API.ManualSequentialGraphRunner](#), and [Mvx2API.RandomAccessGraphR](#)

Public Member Functions

- [SourceInfo GetSourceInfo](#) ()
Retrieves source information about the currently open MVX source.

Protected Member Functions

- [GraphRunner](#) (IntPtr nativeObject)
A constructor.
- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

8.25.1 Detailed Description

A runner of data-processing graphs.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 GraphRunner()

```
Mvx2API.GraphRunner.GraphRunner (
    IntPtr nativeObject ) [protected]
```

A constructor.

Parameters

| | |
|---------------------|------------------------------|
| <i>nativeObject</i> | a native graph runner object |
|---------------------|------------------------------|

8.25.3 Member Function Documentation

8.25.3.1 GetSourceInfo()

`SourceInfo` `Mvx2API.GraphRunner.GetSourceInfo ()`

Retrieves source information about the currently open MVX source.

Returns

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

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

- `public/Mvx2API/core/GraphRunner.cs`

8.26 Mvx2API.InjectFileDataGraphNode Class Reference

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

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [InjectFileDataGraphNode](#) (MVCommon.Guid dataPurposeGuid)
A constructor.
- void [SetFile](#) (MVCommon.String filePath)
Sets a new file to inject the binary content of to frames.

Additional Inherited Members

8.26.1 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.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 InjectFileDataGraphNode()

`Mvx2API.InjectFileDataGraphNode.InjectFileDataGraphNode (`
`MVCommon.Guid dataPurposeGuid)`

A constructor.

Parameters

| | |
|------------------------|-----------------------------------|
| <i>dataPurposeGuid</i> | purpose guid of the injected data |
|------------------------|-----------------------------------|

8.26.3 Member Function Documentation

8.26.3.1 SetFile()

```
void Mvx2API.InjectFileDataGraphNode.SetFile (
    MVCommon.String filePath )
```

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

Parameters

| | |
|-----------------|--------------------|
| <i>filePath</i> | a path of the file |
|-----------------|--------------------|

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

- public/Mvx2API/graphnodes/InjectFileDataGraphNode.cs

8.27 Mvx2API.InjectMemoryDataGraphNode Class Reference

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

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [InjectMemoryDataGraphNode](#) (MVCommon.Guid dataPurposeGuid)
A constructor.
- void [SetData](#) (MVCommon.ByteArray data)
Sets a new data to inject to frames.

Additional Inherited Members

8.27.1 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.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 InjectMemoryDataGraphNode()

```
Mvx2API.InjectMemoryDataGraphNode.InjectMemoryDataGraphNode (
    MVCommon.Guid dataPurposeGuid )
```

A constructor.

Parameters

| | |
|------------------------|-----------------------------------|
| <i>dataPurposeGuid</i> | purpose guid of the injected data |
|------------------------|-----------------------------------|

Exceptions

| | |
|-------------------------|--|
| <i>System.Exception</i> | raised in case the creation of the internal filter fails |
|-------------------------|--|

8.27.3 Member Function Documentation

8.27.3.1 SetData()

```
void Mvx2API.InjectMemoryDataGraphNode.SetData (
    MVCommon.ByteArray data )
```

Sets a new data to inject to frames.

Parameters

| | |
|-------------|--------|
| <i>data</i> | a data |
|-------------|--------|

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

- public/Mvx2API/graphnodes/InjectMemoryDataGraphNode.cs

8.28 Mvx2API.InputEvent Class Reference

An input event structure.

Inherits NativeObjectHolder.

Inherited by [Mvx2API.KeyDownEvent](#), [Mvx2API.KeyUpEvent](#), [Mvx2API.MouseDoubleClickEvent](#), [Mvx2API.MouseDownEvent](#), [Mvx2API.MouseMoveEvent](#), [Mvx2API.MouseUpEvent](#), and [Mvx2API.MouseWheelEvent](#).

Protected Member Functions

- [InputEvent](#) (IntPtr nativeObject)
A constructor.
- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

Properties

- IntPtr [nativeEventObject](#) [get]
A getter of the native event object.

8.28.1 Detailed Description

An input event structure.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 InputEvent()

```
Mvx2API.InputEvent.InputEvent (
    IntPtr nativeObject ) [protected]
```

A constructor.

Parameters

| | |
|---------------------|-----------------------|
| <i>nativeObject</i> | a native event object |
|---------------------|-----------------------|

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

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

8.29 Mvx2API.KeyDownEvent Class Reference

A 'key down' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [KeyDownEvent](#) (Int32 key)
A constructor.

Additional Inherited Members

8.29.1 Detailed Description

A 'key down' event.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 KeyDownEvent()

```
Mvx2API.KeyDownEvent.KeyDownEvent (
    Int32 key )
```

A constructor.

Parameters

| | |
|------------|-----------------------------|
| <i>key</i> | a value of key pressed down |
|------------|-----------------------------|

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

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

8.30 Mvx2API.KeyUpEvent Class Reference

A 'key up' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [KeyUpEvent](#) (Int32 key)
A constructor.

Additional Inherited Members

8.30.1 Detailed Description

A 'key up' event.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 KeyUpEvent()

```
Mvx2API.KeyUpEvent.KeyUpEvent (
    Int32 key )
```

A constructor.

Parameters

| | |
|------------|-------------------------|
| <i>key</i> | a value of key released |
|------------|-------------------------|

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

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

8.31 Mvx2API.ManualGraphBuilder Class Reference

A manual builder of data-processing graphs.

Inherits [Mvx2API.GraphBuilder](#).

Public Member Functions

- [ManualGraphBuilder](#) ()
A constructor.
- [ManualGraphBuilder AppendGraphNode](#) ([GraphNode](#) graphNode)
Appends a graph node to the graph being built.

Static Public Member Functions

- static [ManualGraphBuilder operator+](#) ([ManualGraphBuilder](#) graphBuilder, [GraphNode](#) graphNode)
Appends a graph node to a graph builder.

Additional Inherited Members

8.31.1 Detailed Description

A manual builder of data-processing graphs.

8.31.2 Member Function Documentation

8.31.2.1 AppendGraphNode()

```
ManualGraphBuilder Mvx2API.ManualGraphBuilder.AppendGraphNode (
    GraphNode graphNode )
```

Appends a graph node to the graph being built.

Parameters

| | |
|------------------|------------------------|
| <i>graphNode</i> | a graph node to append |
|------------------|------------------------|

Returns

this graph builder

Exceptions

| | |
|---|---|
| <i>System.InvalidOperationException</i> | raised when the graph builder fails to append the graph node to the graph |
|---|---|

8.31.2.2 operator+()

```
static ManualGraphBuilder Mvx2API.ManualGraphBuilder.operator+ (
    ManualGraphBuilder graphBuilder,
    GraphNode graphNode ) [static]
```

Appends a graph node to a graph builder.

Parameters

| | |
|---------------------|---------------------------------------|
| <i>graphBuilder</i> | a graph builder to append the node to |
| <i>graphNode</i> | a graph node to append |

Returns

the graph builder

Exceptions

| | |
|---|---|
| <i>System.InvalidOperationException</i> | raised when the graph builder fails to append the graph node to the graph |
|---|---|

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

- public/Mvx2API/core/ManualGraphBuilder.cs

8.32 Mvx2API.ManualLiveFrameSourceGraphNode Class Reference

A source graph node for manual production of MVX frames.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [ManualLiveFrameSourceGraphNode](#) ()
A constructor.
- bool [ClearCacheAndReinitializeProperties](#) ([Frame](#) frame, float declaredFPS, bool reassignSequential↵ FrameNumbers=true)
Clears the queue of frames and reinitializes the internal filter's properties based on the first stream of a provided frame.
- bool [PropertiesAreInitialized](#) ()
Checks whether the internal filter's properties have been initialized already.
- void [ClearCache](#) (bool revertReassignedFrameNumbers=true)
Clears the queue of frames.
- bool [PushFrame](#) ([Frame](#) frame)
Pushes another frame to the queue.

Additional Inherited Members

8.32.1 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.

8.32.2 Constructor & Destructor Documentation

8.32.2.1 ManualLiveFrameSourceGraphNode()

```
Mvx2API.ManualLiveFrameSourceGraphNode.ManualLiveFrameSourceGraphNode ( )
```

A constructor.

Exceptions

| | |
|-------------------------|--|
| <i>System.Exception</i> | raised in case the creation of the internal filter fails |
|-------------------------|--|

8.32.3 Member Function Documentation

8.32.3.1 ClearCache()

```
void Mvx2API.ManualLiveFrameSourceGraphNode.ClearCache (
    bool revertReassignedFrameNumbers = true )
```

Clears the queue of frames.

Parameters

| | |
|-------------------------------------|--|
| <i>revertReassignedFrameNumbers</i> | 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 |
|-------------------------------------|--|

8.32.3.2 ClearCacheAndReinitializeProperties()

```
bool Mvx2API.ManualLiveFrameSourceGraphNode.ClearCacheAndReinitializeProperties (
    Frame frame,
    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

| | |
|---------------------------------------|---|
| <i>frame</i> | a frame to reinitialize the internal filter's properties with |
| <i>declaredFPS</i> | declared rate of frames production |
| <i>reassignSequentialFrameNumbers</i> | 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

8.32.3.3 PropertiesAreInitialized()

```
bool Mvx2API.ManualLiveFrameSourceGraphNode.PropertiesAreInitialized ( )
```

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

Returns

true if the properties have been already initialized

8.32.3.4 PushFrame()

```
bool Mvx2API.ManualLiveFrameSourceGraphNode.PushFrame (
    Frame frame )
```

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

| | |
|--------------|-----------------|
| <i>frame</i> | 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.cs

8.33 Mvx2API.ManualOfflineFrameSourceGraphNode Class Reference

A source graph node for manual production of MVX frames.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [ManualOfflineFrameSourceGraphNode](#) ()
A constructor.
- bool [ClearCacheAndReinitializeProperties](#) ([Frame](#) frame, float declaredFPS, bool reassignSequential↔FrameNumbers=true)
Clears the collection of frames and reinitializes the internal filter's properties based on the first stream of a provided frame.
- bool [PropertiesAreInitialized](#) ()
Checks whether the internal filter's properties have been initialized already.
- bool [ClearCache](#) ()
Clears the collection of frames.
- bool [PushFrame](#) ([Frame](#) frame)
Pushes another frame to the collection.

Additional Inherited Members

8.33.1 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.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 ManualOfflineFrameSourceGraphNode()

```
Mvx2API.ManualOfflineFrameSourceGraphNode.ManualOfflineFrameSourceGraphNode ( )
```

A constructor.

Exceptions

| | |
|-------------------------|--|
| <i>System.Exception</i> | raised in case the creation of the internal filter fails |
|-------------------------|--|

8.33.3 Member Function Documentation

8.33.3.1 ClearCache()

```
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

8.33.3.2 ClearCacheAndReinitializeProperties()

```
bool Mvx2API.ManualOfflineFrameSourceGraphNode.ClearCacheAndReinitializeProperties (
    Frame frame,
    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

| | |
|---------------------------------------|---|
| <i>frame</i> | a frame to reinitialize the internal filter's properties with |
| <i>declaredFPS</i> | declared rate of frames production |
| <i>reassignSequentialFrameNumbers</i> | 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

8.33.3.3 PropertiesAreInitialized()

```
bool Mvx2API.ManualOfflineFrameSourceGraphNode.PropertiesAreInitialized ( )
```

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

Returns

true if the properties have been already initialized

8.33.3.4 PushFrame()

```
bool Mvx2API.ManualOfflineFrameSourceGraphNode.PushFrame (  
    Frame frame )
```

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

| | |
|--------------|-----------------|
| <i>frame</i> | 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.cs

8.34 Mvx2API.ManualSequentialGraphRunner Class Reference

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

Inherits [Mvx2API.GraphRunner](#).

Public Member Functions

- [ManualSequentialGraphRunner](#) ([Graph](#) graph)
A constructor.
- bool [RestartWithPlaybackMode](#) ([RunnerPlaybackMode](#) playbackMode)
Restarts the runner with a new playback mode.
- bool [ProcessNextFrame](#) ()
Processes a subsequent frame (depending on the current playback mode).
- void [SeekFrame](#) (UInt32 frameID)
Sets a frame with a given ID as the next to be processed.

Additional Inherited Members

8.34.1 Detailed Description

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

8.34.2 Constructor & Destructor Documentation

8.34.2.1 ManualSequentialGraphRunner()

```
Mvx2API.ManualSequentialGraphRunner.ManualSequentialGraphRunner (
    Graph graph )
```

A constructor.

Parameters

| | |
|--------------|----------------------------------|
| <i>graph</i> | a graph to create the runner for |
|--------------|----------------------------------|

8.34.3 Member Function Documentation

8.34.3.1 ProcessNextFrame()

```
bool Mvx2API.ManualSequentialGraphRunner.ProcessNextFrame ( )
```

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

Returns

true if no error occurred during the processing

8.34.3.2 RestartWithPlaybackMode()

```
bool Mvx2API.ManualSequentialGraphRunner.RestartWithPlaybackMode (
    RunnerPlaybackMode playbackMode )
```

Restarts the runner with a new playback mode.

Parameters

| | |
|---------------------|---------------------------------|
| <i>playbackMode</i> | a playback mode to restart with |
|---------------------|---------------------------------|

Returns

true if the playback mode was successfully changed

8.34.3.3 SeekFrame()

```
void Mvx2API.ManualSequentialGraphRunner.SeekFrame (
    UInt32 frameID )
```

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

Parameters

| | |
|----------------|---|
| <i>frameID</i> | an ID of the frame to be processed next |
|----------------|---|

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

- public/Mvx2API/runners/ManualSequentialGraphRunner.cs

8.35 Mvx2API.MeshData Class Reference

A class containing data of a single mesh.

Inherits NativeObjectHolder.

Public Member Functions

- UInt32 [GetNumVertices](#) ()
A getter of the vertices count.
- IntPtr [GetVertices](#) ()
A getter of the raw pointer to vertices collection.
- bool [CopyVertices](#) (float[] targetVertices)
Copies vertices collection to the target array.
- bool [CopyVerticesRaw](#) (IntPtr targetVertices)
Copies vertices collection to the target array.
- bool [CopyVerticesVec3](#) (Vec3[] targetVertices)
Copies vertices collection to the target array.
- UInt32 [GetNumNormals](#) ()
A getter of the normals count.
- IntPtr [GetNormals](#) ()
A getter of the raw pointer to normals collection.
- bool [CopyNormals](#) (float[] targetNormals)
Copies vertex normals collection to the target array.
- bool [CopyNormalsRaw](#) (IntPtr targetNormals)
Copies vertex normals collection to the target array.
- bool [CopyNormalsVec3](#) (Vec3[] targetNormals)
Copies vertex normals collection to the target array.
- UInt32 [GetNumColors](#) ()
A getter of the colors count.
- IntPtr [GetColorsRGB](#) ()
A getter of the raw pointer to RGB colors collection.
- bool [CopyColorsRGB](#) (byte[] targetColors)
Copies vertex RGB colors collection to the target array.
- bool [CopyColorsRGBRaw](#) (IntPtr targetColors)
Copies vertex RGB colors collection to the target array.
- bool [CopyColorsColRGBA](#) (Col[] targetColors)
Copies vertex RGBA colors collection to the target array.
- bool [CopyColorsRGBARaw](#) (IntPtr targetColors)
Copies vertex RGBA colors collection to the target array.

- UInt32 [GetNumUVs](#) ()
A getter of the UVs count.
- IntPtr [GetUVs](#) ()
A getter of the raw pointer to UVs collection.
- bool [CopyUVs](#) (float[] targetUVs)
Copies vertex UVs collection to the target array.
- bool [CopyUVsRaw](#) (IntPtr targetUVs)
Copies vertex UVs collection to the target array.
- bool [CopyUVsVec2](#) ([Vec2](#)[] targetUVs)
Copies vertex UVs collection to the target array.
- UInt32 [GetNumIndices](#) ()
A getter of the indices count.
- IntPtr [GetIndices](#) ()
A getter of the raw pointer to indices collection.
- bool [CopyIndices](#) (UInt32[] targetIndices)
Copies vertex indices collection to the target array.
- bool [CopyIndicesRaw](#) (IntPtr targetIndices)
Copies vertex indices collection to the target array.
- bool [CopyBoundingBox](#) (float[] targetBoundingBox)
Copies bounding box data to the target array.
- bool [CopyBoundingBoxRaw](#) (IntPtr targetBoundingBox)
Copies bounding box data to the target array.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

Properties

- IntPtr [nativeMeshDataObject](#) [get]
A getter of the native mesh data object.

8.35.1 Detailed Description

A class containing data of a single mesh.

8.35.2 Member Function Documentation

8.35.2.1 CopyBoundingBox()

```
bool Mvx2API.MeshData.CopyBoundingBox (
    float[] targetBoundingBox )
```

Copies bounding box data to the target array.

Parameters

| | |
|--------------------------|--|
| <i>targetBoundingBox</i> | an outputted bounding box data array (must be pre-allocated with 6 elements) |
|--------------------------|--|

Returns

true if the bounding box was successfully copied

8.35.2.2 CopyBoundingBoxRaw()

```
bool Mvx2API.MeshData.CopyBoundingBoxRaw (
    IntPtr targetBoundingBox )
```

Copies bounding box data to the target array.

Parameters

| | |
|--------------------------|---|
| <i>targetBoundingBox</i> | an outputted bounding box data array (must be pre-allocated with 6 * sizeof(float) bytes) |
|--------------------------|---|

Returns

true if the bounding box was successfully copied

8.35.2.3 CopyColorsColRGBA()

```
bool Mvx2API.MeshData.CopyColorsColRGBA (
    Col[] targetColors )
```

Copies vertex RGBA colors collection to the target array.

Parameters

| | |
|---------------------|--|
| <i>targetColors</i> | an outputted vertex RGBA colors array (must be pre-allocated with (colors count) elements) |
|---------------------|--|

Returns

true if the vertex RGBA colors were successfully copied

8.35.2.4 CopyColorsRGB()

```
bool Mvx2API.MeshData.CopyColorsRGB (
    byte[] targetColors )
```

Copies vertex RGB colors collection to the target array.

Parameters

| | |
|---------------------|---|
| <i>targetColors</i> | 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

8.35.2.5 CopyColorsRGBARaw()

```
bool Mvx2API.MeshData.CopyColorsRGBARaw (  
    IntPtr targetColors )
```

Copies vertex RGBA colors collection to the target array.

Parameters

| | |
|---------------------|--|
| <i>targetColors</i> | an outputted vertex RGBA colors array (must be pre-allocated with (4 * colors count * sizeof(byte)) bytes) |
|---------------------|--|

Returns

true if the vertex RGBA colors were successfully copied

8.35.2.6 CopyColorsRGBRaw()

```
bool Mvx2API.MeshData.CopyColorsRGBRaw (  
    IntPtr targetColors )
```

Copies vertex RGB colors collection to the target array.

Parameters

| | |
|---------------------|---|
| <i>targetColors</i> | an outputted vertex RGB colors array (must be pre-allocated with (3 * colors count * sizeof(byte)) bytes) |
|---------------------|---|

Returns

true if the vertex RGB colors were successfully copied

8.35.2.7 CopyIndices()

```
bool Mvx2API.MeshData.CopyIndices (
    UInt32[] targetIndices )
```

Copies vertex indices collection to the target array.

Parameters

| | |
|----------------------|---|
| <i>targetIndices</i> | an outputted vertex indices array (must be pre-allocated with (indices count) elements) |
|----------------------|---|

Returns

true if the vertex indices were successfully copied

8.35.2.8 CopyIndicesRaw()

```
bool Mvx2API.MeshData.CopyIndicesRaw (
    IntPtr targetIndices )
```

Copies vertex indices collection to the target array.

Parameters

| | |
|----------------------|---|
| <i>targetIndices</i> | an outputted vertex indices array (must be pre-allocated with (indices count * sizeof(UInt32)) bytes) |
|----------------------|---|

Returns

true if the vertex indices were successfully copied

8.35.2.9 CopyNormals()

```
bool Mvx2API.MeshData.CopyNormals (
    float[] targetNormals )
```

Copies vertex normals collection to the target array.

Parameters

| | |
|----------------------|---|
| <i>targetNormals</i> | an outputted vertex normals array (must be pre-allocated with (3 * normals count) elements) |
|----------------------|---|

Returns

true if the vertex normals were successfully copied

8.35.2.10 CopyNormalsRaw()

```
bool Mvx2API.MeshData.CopyNormalsRaw (
    IntPtr targetNormals )
```

Copies vertex normals collection to the target array.

Parameters

| | |
|----------------------|--|
| <i>targetNormals</i> | an outputted vertex normals array (must be pre-allocated with (3 * normals count * sizeof(float)) bytes) |
|----------------------|--|

Returns

true if the vertex normals were successfully copied

8.35.2.11 CopyNormalsVec3()

```
bool Mvx2API.MeshData.CopyNormalsVec3 (
    Vec3[] targetNormals )
```

Copies vertex normals collection to the target array.

Parameters

| | |
|----------------------|---|
| <i>targetNormals</i> | an outputted vertex normals array (must be pre-allocated with (normals count) elements) |
|----------------------|---|

Returns

true if the vertex normals were successfully copied

8.35.2.12 CopyUVs()

```
bool Mvx2API.MeshData.CopyUVs (
    float[] targetUVs )
```

Copies vertex UVs collection to the target array.

Parameters

| | |
|------------------|---|
| <i>targetUVs</i> | an outputted vertex UVs array (must be pre-allocated with (2 * UVs count) elements) |
|------------------|---|

Returns

true if the vertex UVs were successfully copied

8.35.2.13 CopyUVsRaw()

```
bool Mvx2API.MeshData.CopyUVsRaw (
    IntPtr targetUVs )
```

Copies vertex UVs collection to the target array.

Parameters

| | |
|------------------|--|
| <i>targetUVs</i> | an outputted vertex UVs array (must be pre-allocated with (2 * UVs count * sizeof(float)) bytes) |
|------------------|--|

Returns

true if the vertex UVs were successfully copied

8.35.2.14 CopyUVsVec2()

```
bool Mvx2API.MeshData.CopyUVsVec2 (
    Vec2[] targetUVs )
```

Copies vertex UVs collection to the target array.

Parameters

| | |
|------------------|---|
| <i>targetUVs</i> | an outputted vertex UVs array (must be pre-allocated with (UVs count) elements) |
|------------------|---|

Returns

true if the vertex UVs were successfully copied

8.35.2.15 CopyVertices()

```
bool Mvx2API.MeshData.CopyVertices (
    float[] targetVertices )
```

Copies vertices collection to the target array.

Parameters

| | |
|-----------------------|--|
| <i>targetVertices</i> | an outputted vertex positions array (must be pre-allocated with (3 * vertices count) elements) |
|-----------------------|--|

Returns

true if the vertex positions were successfully copied

8.35.2.16 CopyVerticesRaw()

```
bool Mvx2API.MeshData.CopyVerticesRaw (
    IntPtr targetVertices )
```

Copies vertices collection to the target array.

Parameters

| | |
|-----------------------|---|
| <i>targetVertices</i> | an outputted vertex positions array (must be pre-allocated with (3 * vertices count * sizeof(float)) bytes) |
|-----------------------|---|

Returns

true if the vertex positions were successfully copied

8.35.2.17 CopyVerticesVec3()

```
bool Mvx2API.MeshData.CopyVerticesVec3 (
    Vec3[] targetVertices )
```

Copies vertices collection to the target array.

Parameters

| | |
|-----------------------|--|
| <i>targetVertices</i> | an outputted vertex positions array (must be pre-allocated with (vertices count) elements) |
|-----------------------|--|

Returns

true if the vertex positions were successfully copied

8.35.2.18 GetColorsRGB()

```
IntPtr Mvx2API.MeshData.GetColorsRGB ( )
```

A getter of the raw pointer to RGB colors collection.

Returns

mesh RGB colors

8.35.2.19 GetIndices()

```
IntPtr Mvx2API.MeshData.GetIndices ( )
```

A getter of the raw pointer to indices collection.

Returns

mesh indices

8.35.2.20 GetNormals()

```
IntPtr Mvx2API.MeshData.GetNormals ( )
```

A getter of the raw pointer to normals collection.

Returns

mesh normals

8.35.2.21 GetNumColors()

```
UInt32 Mvx2API.MeshData.GetNumColors ( )
```

A getter of the colors count.

Returns

count of mesh colors

8.35.2.22 GetNumIndices()

```
UInt32 Mvx2API.MeshData.GetNumIndices ( )
```

A getter of the indices count.

Returns

count of mesh indices

8.35.2.23 GetNumNormals()

```
UInt32 Mvx2API.MeshData.GetNumNormals ( )
```

A getter of the normals count.

Returns

count of mesh normals

8.35.2.24 GetNumUVs()

```
UInt32 Mvx2API.MeshData.GetNumUVs ( )
```

A getter of the UVs count.

Returns

count of mesh UVs

8.35.2.25 GetNumVertices()

```
UInt32 Mvx2API.MeshData.GetNumVertices ( )
```

A getter of the vertices count.

Returns

count of mesh vertices

8.35.2.26 GetUVs()

```
IntPtr Mvx2API.MeshData.GetUVs ( )
```

A getter of the raw pointer to UVs collection.

Returns

mesh UVs

8.35.2.27 GetVertices()

```
IntPtr Mvx2API.MeshData.GetVertices ( )
```

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.cs

8.36 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.

Inherits NativeObjectHolder.

Public Member Functions

- [MeshSplitter](#) (UInt32 maxVerticesCount)
A constructor.
- void [ClearResults](#) ()
Clears results of the previous mesh splitting.
- void [SplitMesh](#) ([MeshData](#) 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.
- UInt32 [GetSplitMeshesCount](#) ()
A getter of split meshes count.
- [MeshData](#) [GetSplitMeshData](#) (UInt32 meshIndex)
Returns a split submesh with a given index.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

8.36.1 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.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 MeshSplitter()

```
Mvx2API.MeshSplitter.MeshSplitter (
    UInt32 maxVerticesCount )
```

A constructor.

Parameters

| | |
|-------------------------|---|
| <i>maxVerticesCount</i> | a maximal count of vertices contained in the resulting split meshes |
|-------------------------|---|

8.36.3 Member Function Documentation

8.36.3.1 GetSplitMeshData()

```
MeshData Mvx2API.MeshSplitter.GetSplitMeshData (
    UInt32 meshIndex )
```

Returns a split submesh with a given index.

Parameters

| | |
|------------------|-----------------------------------|
| <i>meshIndex</i> | 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

8.36.3.2 GetSplitMeshesCount()

```
UInt32 Mvx2API.MeshSplitter.GetSplitMeshesCount ( )
```

A getter of split meshes count.

Returns

count of meshes

8.36.3.3 SplitMesh()

```
void Mvx2API.MeshSplitter.SplitMesh (
    MeshData 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

| | |
|-----------------------|---|
| <i>mesh</i> | a mesh to split |
| <i>indicesMode</i> | an interpretation of indices collection (will be preserved in split meshes) |
| <i>includeNormals</i> | indication whether normals of the mesh shall be included in the splitting process and thus in the resulting submeshes |
| <i>includeColors</i> | indication whether colors of the mesh shall be included in the splitting process and thus in the resulting submeshes |
| <i>includeUVs</i> | 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.cs`

8.37 Mvx2API.MouseDoubleClickEvent Class Reference

A 'mouse double-click' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [MouseDoubleClickEvent](#) (Int32 button, Int32 x, Int32 y)
A constructor.

Additional Inherited Members

8.37.1 Detailed Description

A 'mouse double-click' event.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 MouseDoubleClickEvent()

```
Mvx2API.MouseDoubleClickEvent.MouseDoubleClickEvent (
    Int32 button,
    Int32 x,
    Int32 y )
```

A constructor.

Parameters

| | |
|---------------|---|
| <i>button</i> | a mouse button double-clicked |
| <i>x</i> | an x-coordinate of mouse during the event |
| <i>y</i> | an y-coordinate of mouse during the event |

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

- `public/Mvx2API/data/events/MouseDoubleClickEvent.cs`

8.38 Mvx2API.MouseDownEvent Class Reference

A 'mouse down' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [MouseDownEvent](#) (Int32 button, Int32 x, Int32 y)
A constructor.

Additional Inherited Members

8.38.1 Detailed Description

A 'mouse down' event.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 MouseDownEvent()

```
Mvx2API.MouseDownEvent.MouseDownEvent (
    Int32 button,
    Int32 x,
    Int32 y )
```

A constructor.

Parameters

| | |
|---------------|---|
| <i>button</i> | a mouse button pressed down |
| <i>x</i> | an x-coordinate of mouse during the event |
| <i>y</i> | an y-coordinate of mouse during the event |

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

- `public/Mvx2API/data/events/MouseDownEvent.cs`

8.39 Mvx2API.MouseMoveEvent Class Reference

A 'mouse move' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [MouseMoveEvent](#) (Int32 x, Int32 y)
A constructor.

Additional Inherited Members

8.39.1 Detailed Description

A 'mouse move' event.

8.39.2 Constructor & Destructor Documentation

8.39.2.1 MouseMoveEvent()

```
Mvx2API.MouseMoveEvent.MouseMoveEvent (
    Int32 x,
    Int32 y )
```

A constructor.

Parameters

| | |
|----------|---|
| <i>x</i> | an x-coordinate of mouse during the event |
| <i>y</i> | an y-coordinate of mouse during the event |

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

- `public/Mvx2API/data/events/MouseMoveEvent.cs`

8.40 Mvx2API.MouseUpEvent Class Reference

A 'mouse up' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [MouseUpEvent](#) (Int32 button, Int32 x, Int32 y)
A constructor.

Additional Inherited Members

8.40.1 Detailed Description

A 'mouse up' event.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 MouseUpEvent()

```
Mvx2API.MouseUpEvent.MouseUpEvent (
    Int32 button,
    Int32 x,
    Int32 y )
```

A constructor.

Parameters

| | |
|---------------|---|
| <i>button</i> | a mouse button released |
| <i>x</i> | an x-coordinate of mouse during the event |
| <i>y</i> | an y-coordinate of mouse during the event |

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

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

8.41 Mvx2API.MouseWheelEvent Class Reference

A 'mouse wheel' event.

Inherits [Mvx2API.InputEvent](#).

Public Member Functions

- [MouseWheelEvent](#) (float delta, Int32 x, Int32 y)
A constructor.

Additional Inherited Members

8.41.1 Detailed Description

A 'mouse wheel' event.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 MouseWheelEvent()

```
Mvx2API.MouseWheelEvent.MouseWheelEvent (
    float delta,
    Int32 x,
    Int32 y )
```

A constructor.

Parameters

| | |
|--------------|---|
| <i>delta</i> | a delta value representing mouse wheel movement |
| <i>x</i> | an x-coordinate of mouse during the event |
| <i>y</i> | an y-coordinate of mouse during the event |

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

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

8.42 Mvx2API.ParameterValueChangedListener Class Reference

A listener for changes of graph nodes' parameters.

Inherits [NativeObjectHolder](#).

Inherited by [Mvx2API.DelegatedParameterValueChangedListener](#).

Public Member Functions

- [ParameterValueChangedListener](#) ()
A constructor.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.
- abstract void [OnParameterValueChanged](#) ([GraphNode](#) graphNode, MVCommon.String parameterName, MVCommon.String parameterValueStr)
A callback executed when a parameter of a graph node changes its value.

Properties

- IntPtr [nativeParameterValueChangeListenerObject](#) [get]
A getter of the native parameter value changed listener object.

8.42.1 Detailed Description

A listener for changes of graph nodes' parameters.

8.42.2 Member Function Documentation

8.42.2.1 OnParameterValueChanged()

```
abstract void Mvx2API.ParameterValueChangedListener.OnParameterValueChanged (
    GraphNode graphNode,
    MVCommon.String parameterName,
    MVCommon.String parameterValueStr ) [protected], [pure virtual]
```

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

Parameters

| | |
|--------------------------|---|
| <i>graphNode</i> | a graph node containing the changed parameter |
| <i>parameterName</i> | name of the changed parameter |
| <i>parameterValueStr</i> | parameter's new value in a string form |

Implemented in [Mvx2API.DelegatedParameterValueChangeListener](#).

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

- public/Mvx2API/graphnodes/ParameterValueChangeListener.cs

8.43 Mvx2API.PluginsLoader Class Reference

A loader of MVX plugins.

Static Public Member Functions

- static void [LoadPluginsInFolder](#) (MVCommon.String folder, bool checkCacheFile=false, bool storeCacheFile=false, bool checkSubfolders=true)
Loads all MVX plugins from a specified folder.
- static void [LoadPlugin](#) (MVCommon.String pluginPath)
Loads single MVX plugin specified by its path.

8.43.1 Detailed Description

A loader of MVX plugins.

8.43.2 Member Function Documentation

8.43.2.1 LoadPlugin()

```
static void Mvx2API.PluginsLoader.LoadPlugin (
    MVCommon.String pluginPath ) [static]
```

Loads single MVX plugin specified by its path.

Parameters

| | |
|-------------------|----------------------|
| <i>pluginPath</i> | a path to the plugin |
|-------------------|----------------------|

8.43.2.2 LoadPluginsInFolder()

```
static void Mvx2API.PluginsLoader.LoadPluginsInFolder (
    MVCommon.String folder,
    bool checkCacheFile = false,
    bool storeCacheFile = false,
    bool checkSubfolders = true ) [static]
```

Loads all MVX plugins from a specified folder.

Parameters

| | |
|------------------------|--|
| <i>folder</i> | a folder containing MVX plugin |
| <i>checkCacheFile</i> | an indication whether to check existing cache file of plugins and their filters |
| <i>storeCacheFile</i> | an indication whether to store information about plugins and their filters into a cache file |
| <i>checkSubfolders</i> | if true, checks also subfolders of the folder |

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

- `public/Mvx2API/Utils/PluginsLoader.cs`

8.44 Mvx2API.RandomAccessGraphRunner Class Reference

A random-access runner of data-processing graphs.

Inherits [Mvx2API.GraphRunner](#).

Public Member Functions

- [RandomAccessGraphRunner](#) ([Graph](#) graph)
A constructor.
- `bool` [ProcessFrame](#) (`UInt32` frameID)
Processes a frame with the given ID.

Additional Inherited Members

8.44.1 Detailed Description

A random-access runner of data-processing graphs.

8.44.2 Constructor & Destructor Documentation

8.44.2.1 RandomAccessGraphRunner()

```
Mvx2API.RandomAccessGraphRunner.RandomAccessGraphRunner (
    Graph graph )
```

A constructor.

Parameters

| | |
|--------------|----------------------------------|
| <i>graph</i> | a graph to create the runner for |
|--------------|----------------------------------|

8.44.3 Member Function Documentation

8.44.3.1 ProcessFrame()

```
bool Mvx2API.RandomAccessGraphRunner.ProcessFrame (
    UInt32 frameID )
```

Processes a frame with the given ID.

Parameters

| | |
|----------------|-------------------------------|
| <i>frameID</i> | an ID of the frame to process |
|----------------|-------------------------------|

Returns

true if no error occurred during the processing

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

- public/Mvx2API/runners/RandomAccessGraphRunner.cs

8.45 Mvx2API.Experimental.RendererGraphNode Class Reference

A graph node for rendering visual Mvx2 data.

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [RendererGraphNode](#) (MVCommon.Guid rendererGuid)
A constructor.
- void [Render](#) (Int32 width, Int32 height, bool reinit, Int32 fbo=0)
Invokes rendering of cached data using internal rendering facility.
- void [DestroyRenderer](#) ()
Destroys internal rendering facility (e.g. resources).
- void [HandleInputEvent](#) (InputEvent evt)
Gives internal rendering facility an opportunity to handle input events and customize its behaviour.

Additional Inherited Members

8.45.1 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.

8.45.2 Constructor & Destructor Documentation

8.45.2.1 RendererGraphNode()

```
Mvx2API.Experimental.RendererGraphNode.RendererGraphNode (
    MVCommon.Guid rendererGuid )
```

A constructor.

Parameters

| | |
|---------------------|--|
| <i>rendererGuid</i> | a Guid of renderer filter to instantiate |
|---------------------|--|

Exceptions

| | |
|---------------------------------|--|
| <i>System.ArgumentException</i> | raised when a filter with the given Guid is not registered or it is not a renderer |
|---------------------------------|--|

8.45.3 Member Function Documentation**8.45.3.1 DestroyRenderer()**

```
void Mvx2API.Experimental.RendererGraphNode.DestroyRenderer ( )
```

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

Exceptions

| | |
|---|--|
| <i>System.InvalidOperationException</i> | raised when internal filter does not exist yet or it is not a renderer |
|---|--|

8.45.3.2 HandleInputEvent()

```
void Mvx2API.Experimental.RendererGraphNode.HandleInputEvent (
    InputEvent evt )
```

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

Parameters

| | |
|------------|----------------|
| <i>evt</i> | an input event |
|------------|----------------|

Exceptions

| | |
|---|--|
| <i>System.InvalidOperationException</i> | raised when internal filter does not exist yet or it is not a renderer |
|---|--|

8.45.3.3 Render()

```
void Mvx2API.Experimental.RendererGraphNode.Render (
    Int32 width,
```

```

    Int32 height,
    bool reinit,
    Int32 fbo = 0 )

```

Invokes rendering of cached data using internal rendering facility.

Parameters

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

Exceptions

| | |
|---|--|
| <i>System.InvalidOperationException</i> | 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.cs

8.46 Mvx2API.SingleFilterGraphNode Class Reference

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

Inherits [Mvx2API.GraphNode](#).

Public Member Functions

- [SingleFilterGraphNode](#) (MVCommon.Guid filterGuid, bool singleFilterInstance=false)
A constructor.
- [SingleFilterGraphNode](#) (MVCommon.Guid filterGuid, bool singleFilterInstance, MVCommon.String filter↵Name)
A constructor.
- bool [SetFilterParameterValue](#) (MVCommon.String paramName, MVCommon.String value)
Sets a value of the filter's parameter.
- bool [TryGetFilterParameterValue](#) (MVCommon.String paramName, out MVCommon.String value)
Returns a value of the filter's parameter.
- bool [RegisterParameterValueChangedListener](#) (MVCommon.String paramName, [ParameterValueChangedListener](#) parameterValueChangedListener)
Registers a listener for a parameter value changed event.
- void [UnregisterParameterValueChangedListener](#) (MVCommon.String paramName, [ParameterValueChangedListener](#) parameterValueChangedListener)
Unregisters a listener for a parameter value changed event.
- void [UnregisterAllParameterValueChangedListeners](#) ()
Unregisters all registered listeners for any parameter value changed events.
- [FilterParameterNameEnumerator](#) [CreateParameterNamesEnumerator](#) ()
Creates an enumerator over names of the internal filter's parameters.
- bool [ContainsDataProfile](#) (MVCommon.Guid dataLayerGuid, MVCommon.Guid purposeGuid, bool check↵CompressedDataLayersToo=true)
Checks whether the internal filter (assuming it exists already) contains a data profile with a given guid on its output.
- [DataProfileEnumerator](#) [CreateDataProfilesEnumerator](#) ()
Creates an enumerator over data profile entries of the internal filter (assuming it exists already).

Additional Inherited Members

8.46.1 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.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 SingleFilterGraphNode() [1/2]

```
Mvx2API.SingleFilterGraphNode.SingleFilterGraphNode (
    MVCommon.Guid filterGuid,
    bool singleFilterInstance = false )
```

A constructor.

Parameters

| | |
|-----------------------------|---|
| <i>filterGuid</i> | a GUID of filter |
| <i>singleFilterInstance</i> | 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 |

8.46.2.2 SingleFilterGraphNode() [2/2]

```
Mvx2API.SingleFilterGraphNode.SingleFilterGraphNode (
    MVCommon.Guid filterGuid,
    bool singleFilterInstance,
    MVCommon.String filterName )
```

A constructor.

Parameters

| | |
|-----------------------------|---|
| <i>filterGuid</i> | a GUID of filter |
| <i>singleFilterInstance</i> | 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 |
| <i>filterName</i> | a custom name of the filter |

8.46.3 Member Function Documentation

8.46.3.1 ContainsDataProfile()

```
bool Mvx2API.SingleFilterGraphNode.ContainsDataProfile (
    MVCommon.Guid dataLayerGuid,
    MVCommon.Guid 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 preceding 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

| | |
|-------------------------------------|--|
| <i>dataLayerGuid</i> | a guid of the data layer to check |
| <i>purposeGuid</i> | a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid) |
| <i>checkCompressedDataLayersToo</i> | 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

| | |
|---|---|
| <i>System.InvalidOperationException</i> | raised in case the internal filter does not exist yet |
|---|---|

8.46.3.2 CreateDataProfilesEnumerator()

```
DataProfileEnumerator Mvx2API.SingleFilterGraphNode.CreateDataProfilesEnumerator ( )
```

Creates an enumerator over data profile entries 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 preceding 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.

Returns

a new enumerator

Exceptions

| | |
|---|---|
| <i>System.InvalidOperationException</i> | raised in case the internal filter does not exist yet |
|---|---|

8.46.3.3 CreateParameterNamesEnumerator()

```
FilterParameterNameEnumerator Mvx2API.SingleFilterGraphNode.CreateParameterNamesEnumerator ( )
```

Creates an enumerator over names of the internal filter's parameters.

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.

Returns

a new enumerator

Exceptions

| | |
|---|---|
| <i>System.InvalidOperationException</i> | raised in case the internal filter does not exist yet |
|---|---|

8.46.3.4 RegisterParameterValueChangedListener()

```
bool Mvx2API.SingleFilterGraphNode.RegisterParameterValueChangedListener (
    MVCommon.String paramName,
    ParameterValueChangedListener parameterValueChangedListener )
```

Registers a listener for a parameter value changed event.

Parameters

| | |
|--------------------------------------|---|
| <i>paramName</i> | a name of the parameter to listen to changes of |
| <i>parameterValueChangedListener</i> | a listener for the value change event |

Returns

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

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | raised in case the internal filter is supposed to exist already but does not |
|---------------------------|--|

8.46.3.5 SetFilterParameterValue()

```
bool Mvx2API.SingleFilterGraphNode.SetFilterParameterValue (
    MVCommon.String paramName,
    MVCommon.String value )
```

Sets a value of the filter's parameter.

Parameters

| | |
|------------------|---|
| <i>paramName</i> | a name of the parameter to set |
| <i>value</i> | a string representation of the value to set |

Returns

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

Exceptions

| | |
|---|--|
| <i>System.InvalidOperationException</i> | 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.

8.46.3.6 TryGetFilterParameterValue()

```
bool Mvx2API.SingleFilterGraphNode.TryGetFilterParameterValue (
    MVCommon.String paramName,
    out MVCommon.String value )
```

Returns a value of the filter's parameter.

Parameters

| | |
|------------------|---|
| <i>paramName</i> | a name of the parameter to get |
| <i>value</i> | a resulting parameter value after the call in case true is returned |

Returns

true if the parameter exists and its value was retrieved

Exceptions

| | |
|---|---|
| <i>System.InvalidOperationException</i> | raised in case the internal filter does not exist yet |
|---|---|

8.46.3.7 UnregisterParameterValueChangedListener()

```
void Mvx2API.SingleFilterGraphNode.UnregisterParameterValueChangedListener (
    MVCommon.String paramName,
    ParameterValueChangedListener parameterValueChangedListener )
```

Unregisters a listener for a parameter value changed event.

Parameters

| | |
|--------------------------------------|---|
| <i>paramName</i> | a name of the parameter to stop listening to changes of |
| <i>parameterValueChangedListener</i> | a listener to unregister |

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

- public/Mvx2API/graphnodes/SingleFilterGraphNode.cs

8.47 Mvx2API.SourceInfo Class Reference

An information provider about an MVX source.

Inherits NativeObjectHolder.

Public Member Functions

- UInt32 [GetNumFrames](#) ()
Returns number of frames in the source.
- float [GetFPS](#) ()
Returns source's framerate.
- bool [ContainsDataLayer](#) (MVCommon.Guid dataLayerGuid, bool checkCompressedDataLayersToo=true)
Checks whether the source contains a data layer with a given guid.
- bool [ContainsDataLayer](#) (MVCommon.Guid dataLayerGuid, MVCommon.Guid purposeGuid, bool checkCompressedDataLayersToo=true)
Checks whether the source contains a data layer with a given guid.
- [DataProfileEnumerator CreateDataProfilesEnumerator](#) ()
Creates an enumerator over data profile entries of the source.

Protected Member Functions

- override void [DestroyNativeObject](#) ()
Destroys the native object in a customized way.

8.47.1 Detailed Description

An information provider about an MVX source.

8.47.2 Member Function Documentation

8.47.2.1 ContainsDataLayer() [1/2]

```
bool Mvx2API.SourceInfo.ContainsDataLayer (
    MVCommon.Guid dataLayerGuid,
    bool checkCompressedDataLayersToo = true )
```

Checks whether the source contains a data layer with a given guid.

Parameters

| | |
|-------------------------------------|--|
| <i>dataLayerGuid</i> | a guid of the data layer to check |
| <i>checkCompressedDataLayersToo</i> | 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

8.47.2.2 ContainsDataLayer() [2/2]

```
bool Mvx2API.SourceInfo.ContainsDataLayer (
    MVCommon.Guid dataLayerGuid,
    MVCommon.Guid purposeGuid,
    bool checkCompressedDataLayersToo = true )
```

Checks whether the source contains a data layer with a given guid.

Parameters

| | |
|-------------------------------------|--|
| <i>dataLayerGuid</i> | a guid of the data layer to check |
| <i>purposeGuid</i> | a purpose guid of the data layer to check (Guid::Nil() is interpreted as 'any' purpose guid) |
| <i>checkCompressedDataLayersToo</i> | 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

8.47.2.3 CreateDataProfilesEnumerator()

```
DataProfileEnumerator Mvx2API.SourceInfo.CreateDataProfilesEnumerator ( )
```

Creates an enumerator over data profile entries of the source.

Returns

a new enumerator

8.47.2.4 GetFPS()

```
float Mvx2API.SourceInfo.GetFPS ( )
```

Returns source's framerate.

Returns

framerate

8.47.2.5 GetNumFrames()

```
UInt32 Mvx2API.SourceInfo.GetNumFrames ( )
```

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.cs

8.48 Mvx2API.Utils Class Reference

An MVX utilities class.

Static Public Member Functions

- static MVCommon.String [GetAppExeFilePath](#) ()
Returns path of the application's executable file.
- static MVCommon.String [GetAppExeDirectory](#) ()
Returns directory of the application's executable file.

Properties

- static MVCommon.Logger? [MVXLoggerInstance](#) [get, set]
A property for accessing MVX logger instance.
- static MVCommon.GuidAliasDatabase [MVXGuidAliasDatabase](#) [get]
A getter of the database containing MVX2 framework's internal guids and their aliases.

8.48.1 Detailed Description

An MVX utilities class.

8.48.2 Member Function Documentation

8.48.2.1 GetAppExeDirectory()

```
static MVCommon.String Mvx2API.Utils.GetAppExeDirectory ( ) [static]
```

Returns directory of the application's executable file.

Returns

executable directory

8.48.2.2 GetAppExeFilePath()

```
static MVCommon.String Mvx2API.Utils.GetAppExeFilePath ( ) [static]
```

Returns path of the application's executable file.

Returns

executable file path

8.48.3 Property Documentation

8.48.3.1 MVXLoggerInstance

```
MVCommon.Logger? Mvx2API.Utils.MVXLoggerInstance [static], [get], [set]
```

A property for accessing MVX logger instance.

There is no logger instance set by default - it is a responsibility of the application to install one.

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

- public/Mvx2API/Utils/Utils.cs

8.49 Mvx2API_VEC2 Struct Reference

A structure containing 2D position data.

Public Attributes

- float [x](#)
A x-coordinate.
- float [y](#)
A y-coordinate.

8.49.1 Detailed Description

A structure containing 2D position data.

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

- `public/Mvx2API/data/mesh/MeshDataTypes.cs`

8.50 Mvx2API_VEC3 Struct Reference

A structure containing 3D position data.

Public Attributes

- float [x](#)
A x-coordinate.
- float [y](#)
A y-coordinate.
- float [z](#)
A z-coordinate.

8.50.1 Detailed Description

A structure containing 3D position data.

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

- `public/Mvx2API/data/mesh/MeshDataTypes.cs`

Index

- ActivateStreamWithIndex
 - Mvx2API.Frame, [51](#)
- AppendGraphNode
 - Mvx2API.ManualGraphBuilder, [87](#)
- AsyncFrameAccessGraphNode
 - Mvx2API.AsyncFrameAccessGraphNode, [31](#)
- AutoCompressorGraphNode
 - Mvx2API.AutoCompressorGraphNode, [33](#)
- AutoDecompressorGraphNode
 - Mvx2API.AutoDecompressorGraphNode, [33](#)
- AutoSequentialGraphRunner
 - Mvx2API.AutoSequentialGraphRunner, [34](#)
- BlockFPSGraphNode
 - Mvx2API.BlockFPSGraphNode, [38](#)
- BlockGraphNode
 - Mvx2API.BlockGraphNode, [40](#)
- BlockManualGraphNode
 - Mvx2API.BlockManualGraphNode, [42](#)
- ClearCache
 - Mvx2API.ManualLiveFrameSourceGraphNode, [89](#)
 - Mvx2API.ManualOfflineFrameSourceGraphNode, [92](#)
- ClearCacheAndReinitializeProperties
 - Mvx2API.ManualLiveFrameSourceGraphNode, [90](#)
 - Mvx2API.ManualOfflineFrameSourceGraphNode, [92](#)
- CompileGraphAndReset
 - Mvx2API.GraphBuilder, [77](#)
- ContainsDataLayer
 - Mvx2API.SourceInfo, [127](#)
- ContainsDataProfile
 - Mvx2API.GraphBuilder, [77](#)
 - Mvx2API.SingleFilterGraphNode, [123](#)
- CopyBoundingBox
 - Mvx2API.MeshData, [98](#)
- CopyBoundingBoxRaw
 - Mvx2API.MeshData, [99](#)
- CopyColorsColRGBA
 - Mvx2API.MeshData, [99](#)
- CopyColorsRGB
 - Mvx2API.MeshData, [99](#)
- CopyColorsRGBARaw
 - Mvx2API.MeshData, [101](#)
- CopyColorsRGBRaw
 - Mvx2API.MeshData, [101](#)
- CopyIndices
 - Mvx2API.MeshData, [101](#)
- CopyIndicesRaw
 - Mvx2API.MeshData, [102](#)
- CopyNormals
 - Mvx2API.MeshData, [102](#)
- CopyNormalsRaw
 - Mvx2API.MeshData, [103](#)
- CopyNormalsVec3
 - Mvx2API.MeshData, [103](#)
- CopyPCMDData
 - Mvx2API.FrameAudioExtractor, [57](#)
- CopyPCMDDataRaw
 - Mvx2API.FrameAudioExtractor, [57](#), [58](#)
- CopyTextureData
 - Mvx2API.FrameTextureExtractor, [70](#), [71](#)
- CopyTextureDataRaw
 - Mvx2API.FrameTextureExtractor, [71](#)
- CopyUVs
 - Mvx2API.MeshData, [103](#)
- CopyUVsRaw
 - Mvx2API.MeshData, [104](#)
- CopyUVsVec2
 - Mvx2API.MeshData, [104](#)
- CopyVertices
 - Mvx2API.MeshData, [104](#)
- CopyVerticesRaw
 - Mvx2API.MeshData, [106](#)
- CopyVerticesVec3
 - Mvx2API.MeshData, [106](#)
- CreateDataProfilesEnumerator
 - Mvx2API.Frame, [51](#)
 - Mvx2API.GraphBuilder, [77](#)
 - Mvx2API.SingleFilterGraphNode, [123](#)
 - Mvx2API.SourceInfo, [127](#)
- CreateParameterNamesEnumerator
 - Mvx2API.SingleFilterGraphNode, [124](#)
- DataProfile
 - Mvx2API.DataProfile, [44](#)
- DataProfileEnumerator
 - Mvx2API.DataProfileEnumerator, [45](#)
- DelegatedFrameListener
 - Mvx2API.DelegatedFrameListener, [46](#)
- DelegatedParameterValueChangeListener
 - Mvx2API.DelegatedParameterValueChangeListener, [48](#)
- DestroyRenderer
 - Mvx2API.Experimental.RendererGraphNode, [120](#)
- FB_BLOCK_FRAMES
 - Mvx2API.BlockGraphNode, [40](#)
- FB_DROP_FRAMES

- Mvx2API.BlockGraphNode, 40
- FilterParameterNameEnumerator
 - Mvx2API.FilterParameterNameEnumerator, 49
- Frame
 - Mvx2API.Frame, 51
- FullBehaviour
 - Mvx2API.BlockGraphNode, 40
- GetActiveStreamIndex
 - Mvx2API.Frame, 52
- GetAppExeDirectory
 - Mvx2API.Utills, 129
- GetAppExeFilePath
 - Mvx2API.Utills, 129
- GetAudioSamplingInfo
 - Mvx2API.FrameAudioExtractor, 58, 59
- GetByteArrayData
 - Mvx2API.FrameMiscDataExtractor, 65
- GetColorCameraParams
 - Mvx2API.FrameMiscDataExtractor, 65, 66
- GetColorsRGB
 - Mvx2API.MeshData, 106
- GetDroppedFramesCount
 - Mvx2API.BlockGraphNode, 41
- GetFPS
 - Mvx2API.SourceInfo, 128
- GetIndices
 - Mvx2API.MeshData, 107
- GetIrcameraParams
 - Mvx2API.FrameMiscDataExtractor, 66, 67
- GetMeshData
 - Mvx2API.FrameMeshExtractor, 63
- GetNormals
 - Mvx2API.MeshData, 107
- GetNumColors
 - Mvx2API.MeshData, 107
- GetNumFrames
 - Mvx2API.SourceInfo, 128
- GetNumIndices
 - Mvx2API.MeshData, 107
- GetNumNormals
 - Mvx2API.MeshData, 107
- GetNumStreams
 - Mvx2API.Frame, 52
- GetNumUVs
 - Mvx2API.MeshData, 108
- GetNumVertices
 - Mvx2API.MeshData, 108
- GetPCMDData
 - Mvx2API.FrameAudioExtractor, 59, 60
- GetPCMDDataOffset
 - Mvx2API.FrameAudioExtractor, 60
- GetPCMDDataSize
 - Mvx2API.FrameAudioExtractor, 61
- GetPlaybackState
 - Mvx2API.AutoSequentialGraphRunner, 35
- GetRecentProcessedFrame
 - Mvx2API.FrameAccessGraphNode, 55
- GetSegmentID
 - Mvx2API.FrameMiscDataExtractor, 67, 68
- GetSourceInfo
 - Mvx2API.GraphRunner, 81
- GetSplitMeshData
 - Mvx2API.MeshSplitter, 110
- GetSplitMeshesCount
 - Mvx2API.MeshSplitter, 110
- GetStreamAtomNr
 - Mvx2API.Frame, 52
- GetStreamAtomTimestamp
 - Mvx2API.Frame, 52
- GetStreamId
 - Mvx2API.Frame, 53
- GetTextureData
 - Mvx2API.FrameTextureExtractor, 72
- GetTextureDataSizeInBytes
 - Mvx2API.FrameTextureExtractor, 73
- GetTextureResolution
 - Mvx2API.FrameTextureExtractor, 74
- GetTransform
 - Mvx2API.FrameMiscDataExtractor, 68
- GetUVs
 - Mvx2API.MeshData, 108
- GetVertices
 - Mvx2API.MeshData, 108
- GraphBuilder
 - Mvx2API.GraphBuilder, 76
- GraphNode
 - Mvx2API.GraphNode, 79
- GraphRunner
 - Mvx2API.GraphRunner, 80
- HandleInputEvent
 - Mvx2API.Experimental.RendererGraphNode, 120
- InjectFileDataGraphNode
 - Mvx2API.InjectFileDataGraphNode, 81
- InjectMemoryDataGraphNode
 - Mvx2API.InjectMemoryDataGraphNode, 83
- InputEvent
 - Mvx2API.InputEvent, 84
- KeyDownEvent
 - Mvx2API.KeyDownEvent, 85
- KeyUpEvent
 - Mvx2API.KeyUpEvent, 86
- LoadPlugin
 - Mvx2API.PluginsLoader, 117
- LoadPluginsInFolder
 - Mvx2API.PluginsLoader, 117
- ManualLiveFrameSourceGraphNode
 - Mvx2API.ManualLiveFrameSourceGraphNode, 89
- ManualOfflineFrameSourceGraphNode
 - Mvx2API.ManualOfflineFrameSourceGraphNode, 92
- ManualSequentialGraphRunner
 - Mvx2API.ManualSequentialGraphRunner, 95

- MeshIndicesMode
 - Mvx2API, [27](#)
- MeshSplitter
 - Mvx2API.MeshSplitter, [109](#)
- MIM_LineList
 - Mvx2API, [28](#)
- MIM_PointList
 - Mvx2API, [28](#)
- MIM_QuadList
 - Mvx2API, [28](#)
- MIM_TriangleList
 - Mvx2API, [28](#)
- MouseDoubleClickEvent
 - Mvx2API.MouseDoubleClickEvent, [111](#)
- MouseDownEvent
 - Mvx2API.MouseDownEvent, [112](#)
- MouseMoveEvent
 - Mvx2API.MouseMoveEvent, [113](#)
- MouseUpEvent
 - Mvx2API.MouseUpEvent, [114](#)
- MouseWheelEvent
 - Mvx2API.MouseWheelEvent, [115](#)
- Mvx2API, [25](#)
 - MeshIndicesMode, [27](#)
 - MIM_LineList, [28](#)
 - MIM_PointList, [28](#)
 - MIM_QuadList, [28](#)
 - MIM_TriangleList, [28](#)
 - RPM_BACKWARD_LOOP, [28](#)
 - RPM_BACKWARD_ONCE, [28](#)
 - RPM_FORWARD_LOOP, [28](#)
 - RPM_FORWARD_ONCE, [28](#)
 - RPM_PINGPONG, [28](#)
 - RPM_PINGPONG_INVERSE, [28](#)
 - RPM_REALTIME, [28](#)
 - RPS_Paused, [28](#)
 - RPS_Playing, [28](#)
 - RPS_Stopped, [28](#)
 - RunnerPlaybackMode, [28](#)
 - RunnerPlaybackState, [28](#)
- Mvx2API.AsyncFrameAccessGraphNode, [31](#)
 - AsyncFrameAccessGraphNode, [31](#)
 - SetFrameListener, [32](#)
- Mvx2API.AutoCompressorGraphNode, [32](#)
 - AutoCompressorGraphNode, [33](#)
- Mvx2API.AutoDecompressorGraphNode, [33](#)
 - AutoDecompressorGraphNode, [33](#)
- Mvx2API.AutoSequentialGraphRunner, [34](#)
 - AutoSequentialGraphRunner, [34](#)
 - GetPlaybackState, [35](#)
 - Pause, [35](#)
 - Play, [35](#)
 - Resume, [36](#)
 - SeekFrame, [36](#)
 - Stop, [36](#)
- Mvx2API.BasicDataLayersGuids, [37](#)
- Mvx2API.BlockFPSGraphNode, [38](#)
 - BlockFPSGraphNode, [38](#)
- SetFPS, [39](#)
- Mvx2API.BlockGraphNode, [39](#)
 - BlockGraphNode, [40](#)
 - FB_BLOCK_FRAMES, [40](#)
 - FB_DROP_FRAMES, [40](#)
 - FullBehaviour, [40](#)
 - GetDroppedFramesCount, [41](#)
 - SetFullBehaviour, [41](#)
- Mvx2API.BlockManualGraphNode, [41](#)
 - BlockManualGraphNode, [42](#)
 - PullNextProcessedFrame, [42](#)
- Mvx2API.Col, [43](#)
- Mvx2API.DataProfile, [43](#)
 - DataProfile, [44](#)
- Mvx2API.DataProfileEnumerator, [45](#)
 - DataProfileEnumerator, [45](#)
- Mvx2API.DelegatedFrameListener, [45](#)
 - DelegatedFrameListener, [46](#)
 - OnFrameProcessed, [46](#)
 - OnFrameProcessedDelegate, [47](#)
- Mvx2API.DelegatedParameterValueChangeListener, [47](#)
 - DelegatedParameterValueChangeListener, [48](#)
 - OnParameterValueChanged, [48](#)
 - OnParameterValueChangedDelegate, [48](#)
- Mvx2API.Experimental, [28](#)
- Mvx2API.Experimental.RendererGraphNode, [119](#)
 - DestroyRenderer, [120](#)
 - HandleInputEvent, [120](#)
 - Render, [120](#)
 - RendererGraphNode, [119](#)
- Mvx2API.FilterParameterNameEnumerator, [49](#)
 - FilterParameterNameEnumerator, [49](#)
- Mvx2API.Frame, [50](#)
 - ActivateStreamWithIndex, [51](#)
 - CreateDataProfilesEnumerator, [51](#)
 - Frame, [51](#)
 - GetActiveStreamIndex, [52](#)
 - GetNumStreams, [52](#)
 - GetStreamAtomNr, [52](#)
 - GetStreamAtomTimestamp, [52](#)
 - GetStreamId, [53](#)
 - StreamContainsDataLayer, [53](#)
- Mvx2API.FrameAccessGraphNode, [55](#)
 - GetRecentProcessedFrame, [55](#)
- Mvx2API.FrameAudioExtractor, [56](#)
 - CopyPCMDData, [57](#)
 - CopyPCMDDataRaw, [57, 58](#)
 - GetAudioSamplingInfo, [58, 59](#)
 - GetPCMDData, [59, 60](#)
 - GetPCMDDataOffset, [60](#)
 - GetPCMDDataSize, [61](#)
- Mvx2API.FrameListener, [62](#)
 - OnFrameProcessed, [62](#)
- Mvx2API.FrameMeshExtractor, [63](#)
 - GetMeshData, [63](#)
- Mvx2API.FrameMiscDataExtractor, [64](#)
 - GetByteArrayData, [65](#)

- GetColorCameraParams, [65](#), [66](#)
- GetIRCameraParams, [66](#), [67](#)
- GetSegmentID, [67](#), [68](#)
- GetTransform, [68](#)
- Mvx2API.FrameTextureExtractor, [69](#)
 - CopyTextureData, [70](#), [71](#)
 - CopyTextureDataRow, [71](#)
 - GetTextureData, [72](#)
 - GetTextureDataSizeInBytes, [73](#)
 - GetTextureResolution, [74](#)
 - TextureType, [70](#)
 - TT_ASTC, [70](#)
 - TT_DEPTH, [70](#)
 - TT_DXT1, [70](#)
 - TT_DXT5YCOCG, [70](#)
 - TT_ETC2, [70](#)
 - TT_IR, [70](#)
 - TT_NV12, [70](#)
 - TT_NV21, [70](#)
 - TT_NVX, [70](#)
 - TT_RGB, [70](#)
- Mvx2API.Graph, [75](#)
 - Reinitialize, [75](#)
- Mvx2API.GraphBuilder, [76](#)
 - CompileGraphAndReset, [77](#)
 - ContainsDataProfile, [77](#)
 - CreateDataProfilesEnumerator, [77](#)
 - GraphBuilder, [76](#)
 - Refresh, [78](#)
- Mvx2API.GraphNode, [78](#)
 - GraphNode, [79](#)
 - NativeObjectToGraphNode, [79](#)
- Mvx2API.GraphRunner, [80](#)
 - GetSourceInfo, [81](#)
 - GraphRunner, [80](#)
- Mvx2API.InjectFileDataGraphNode, [81](#)
 - InjectFileDataGraphNode, [81](#)
 - SetFile, [82](#)
- Mvx2API.InjectMemoryDataGraphNode, [82](#)
 - InjectMemoryDataGraphNode, [83](#)
 - SetData, [83](#)
- Mvx2API.InputEvent, [83](#)
 - InputEvent, [84](#)
- Mvx2API.KeyDownEvent, [84](#)
 - KeyDownEvent, [85](#)
- Mvx2API.KeyUpEvent, [85](#)
 - KeyUpEvent, [86](#)
- Mvx2API.ManualGraphBuilder, [86](#)
 - AppendGraphNode, [87](#)
 - operator+, [88](#)
- Mvx2API.ManualLiveFrameSourceGraphNode, [88](#)
 - ClearCache, [89](#)
 - ClearCacheAndReinitializeProperties, [90](#)
 - ManualLiveFrameSourceGraphNode, [89](#)
 - PropertiesAreInitialized, [90](#)
 - PushFrame, [91](#)
- Mvx2API.ManualOfflineFrameSourceGraphNode, [91](#)
 - ClearCache, [92](#)
 - ClearCacheAndReinitializeProperties, [92](#)
 - ManualOfflineFrameSourceGraphNode, [92](#)
 - PropertiesAreInitialized, [93](#)
 - PushFrame, [93](#)
- Mvx2API.ManualSequentialGraphRunner, [95](#)
 - ManualSequentialGraphRunner, [95](#)
 - ProcessNextFrame, [96](#)
 - RestartWithPlaybackMode, [96](#)
 - SeekFrame, [96](#)
- Mvx2API.MeshData, [97](#)
 - CopyBoundingBox, [98](#)
 - CopyBoundingBoxRaw, [99](#)
 - CopyColorsColRGBA, [99](#)
 - CopyColorsRGB, [99](#)
 - CopyColorsRGBARaw, [101](#)
 - CopyColorsRGBRaw, [101](#)
 - CopyIndices, [101](#)
 - CopyIndicesRaw, [102](#)
 - CopyNormals, [102](#)
 - CopyNormalsRaw, [103](#)
 - CopyNormalsVec3, [103](#)
 - CopyUVs, [103](#)
 - CopyUVsRaw, [104](#)
 - CopyUVsVec2, [104](#)
 - CopyVertices, [104](#)
 - CopyVerticesRaw, [106](#)
 - CopyVerticesVec3, [106](#)
 - GetColorsRGB, [106](#)
 - GetIndices, [107](#)
 - GetNormals, [107](#)
 - GetNumColors, [107](#)
 - GetNumIndices, [107](#)
 - GetNumNormals, [107](#)
 - GetNumUVs, [108](#)
 - GetNumVertices, [108](#)
 - GetUVs, [108](#)
 - GetVertices, [108](#)
- Mvx2API.MeshSplitter, [109](#)
 - GetSplitMeshData, [110](#)
 - GetSplitMeshesCount, [110](#)
 - MeshSplitter, [109](#)
 - SplitMesh, [110](#)
- Mvx2API.MouseDoubleClickEvent, [111](#)
 - MouseDoubleClickEvent, [111](#)
- Mvx2API.MouseDownEvent, [112](#)
 - MouseDownEvent, [112](#)
- Mvx2API.MouseMoveEvent, [113](#)
 - MouseMoveEvent, [113](#)
- Mvx2API.MouseUpEvent, [114](#)
 - MouseUpEvent, [114](#)
- Mvx2API.MouseWheelEvent, [114](#)
 - MouseWheelEvent, [115](#)
- Mvx2API.ParameterValueChangedListener, [115](#)
 - OnParameterValueChanged, [116](#)
- Mvx2API.PluginsLoader, [116](#)
 - LoadPlugin, [117](#)
 - LoadPluginsInFolder, [117](#)
- Mvx2API.RandomAccessGraphRunner, [118](#)

- ProcessFrame, 118
- RandomAccessGraphRunner, 118
- Mvx2API.SingleFilterGraphNode, 121
 - ContainsDataProfile, 123
 - CreateDataProfilesEnumerator, 123
 - CreateParameterNamesEnumerator, 124
 - RegisterParameterValueChangedListener, 124
 - SetFilterParameterValue, 125
 - SingleFilterGraphNode, 122
 - TryGetFilterParameterValue, 125
 - UnregisterParameterValueChangedListener, 126
- Mvx2API.SourceInfo, 126
 - ContainsDataLayer, 127
 - CreateDataProfilesEnumerator, 127
 - GetFPS, 128
 - GetNumFrames, 128
- Mvx2API.Utils, 128
 - GetAppExeDirectory, 129
 - GetAppExeFilePath, 129
 - MVXLoggerInstance, 129
- Mvx2API.Vec2, 130
- Mvx2API.Vec3, 130
- MVXLoggerInstance
 - Mvx2API.Utils, 129
- NativeObjectToGraphNode
 - Mvx2API.GraphNode, 79
- OnFrameProcessed
 - Mvx2API.DelegatedFrameListener, 46
 - Mvx2API.FrameListener, 62
- OnFrameProcessedDelegate
 - Mvx2API.DelegatedFrameListener, 47
- OnParameterValueChanged
 - Mvx2API.DelegatedParameterValueChangedListener, 48
 - Mvx2API.ParameterValueChangedListener, 116
- OnParameterValueChangedDelegate
 - Mvx2API.DelegatedParameterValueChangedListener, 48
- operator+
 - Mvx2API.ManualGraphBuilder, 88
- Pause
 - Mvx2API.AutoSequentialGraphRunner, 35
- Play
 - Mvx2API.AutoSequentialGraphRunner, 35
- ProcessFrame
 - Mvx2API.RandomAccessGraphRunner, 118
- ProcessNextFrame
 - Mvx2API.ManualSequentialGraphRunner, 96
- PropertiesAreInitialized
 - Mvx2API.ManualLiveFrameSourceGraphNode, 90
 - Mvx2API.ManualOfflineFrameSourceGraphNode, 93
- PullNextProcessedFrame
 - Mvx2API.BlockManualGraphNode, 42
- PushFrame
 - Mvx2API.ManualLiveFrameSourceGraphNode, 91
 - Mvx2API.ManualOfflineFrameSourceGraphNode, 93
- RandomAccessGraphRunner
 - Mvx2API.RandomAccessGraphRunner, 118
- Refresh
 - Mvx2API.GraphBuilder, 78
- RegisterParameterValueChangedListener
 - Mvx2API.SingleFilterGraphNode, 124
- Reinitialize
 - Mvx2API.Graph, 75
- Render
 - Mvx2API.Experimental.RendererGraphNode, 120
- RendererGraphNode
 - Mvx2API.Experimental.RendererGraphNode, 119
- RestartWithPlaybackMode
 - Mvx2API.ManualSequentialGraphRunner, 96
- Resume
 - Mvx2API.AutoSequentialGraphRunner, 36
- RPM_BACKWARD_LOOP
 - Mvx2API, 28
- RPM_BACKWARD_ONCE
 - Mvx2API, 28
- RPM_FORWARD_LOOP
 - Mvx2API, 28
- RPM_FORWARD_ONCE
 - Mvx2API, 28
- RPM_PINGPONG
 - Mvx2API, 28
- RPM_PINGPONG_INVERSE
 - Mvx2API, 28
- RPM_REALTIME
 - Mvx2API, 28
- RPS_Paused
 - Mvx2API, 28
- RPS_Playing
 - Mvx2API, 28
- RPS_Stopped
 - Mvx2API, 28
- RunnerPlaybackMode
 - Mvx2API, 28
- RunnerPlaybackState
 - Mvx2API, 28
- SeekFrame
 - Mvx2API.AutoSequentialGraphRunner, 36
 - Mvx2API.ManualSequentialGraphRunner, 96
- SetData
 - Mvx2API.InjectMemoryDataGraphNode, 83
- SetFile
 - Mvx2API.InjectFileDataGraphNode, 82
- SetFilterParameterValue
 - Mvx2API.SingleFilterGraphNode, 125
- SetFPS
 - Mvx2API.BlockFPSGraphNode, 39
- SetFrameListener
 - Mvx2API.AsyncFrameAccessGraphNode, 32
- SetFullBehaviour
 - Mvx2API.BlockGraphNode, 41

- SingleFilterGraphNode
 - Mvx2API.SingleFilterGraphNode, [122](#)
- SplitMesh
 - Mvx2API.MeshSplitter, [110](#)
- Stop
 - Mvx2API.AutoSequentialGraphRunner, [36](#)
- StreamContainsDataLayer
 - Mvx2API.Frame, [53](#)
- TextureType
 - Mvx2API.FrameTextureExtractor, [70](#)
- TryGetFilterParameterValue
 - Mvx2API.SingleFilterGraphNode, [125](#)
- TT_ASTC
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_DEPTH
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_DXT1
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_DXT5YCOCG
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_ETC2
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_IR
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_NV12
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_NV21
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_NVX
 - Mvx2API.FrameTextureExtractor, [70](#)
- TT_RGB
 - Mvx2API.FrameTextureExtractor, [70](#)
- UnregisterParameterValueChangeListener
 - Mvx2API.SingleFilterGraphNode, [126](#)