Mvx2Unity

Generated by Doxygen 1.8.16

Chapter 1

Mantis Vision: Mvx2Unity

A Unity3D plugin for processing Mvx2 volumetric content

Plugin Description

Mvx2Unity is a plugin for Unity3D game engine, which makes a subset of Mvx2 framework services, commonly known as Mvx2API, available for use in Unity3D-based applications.

The plugin contains multiple source code samples and their applications in sample Unity3D scenes. All the samples are ready to be used as-is in simpler applications, but can also serve as a source of a lot of inspiration for custom implementations based on Mvx2 services in more complex use-cases.

Table of Contents

Supported Platforms

Currently the plugin works on these platforms:

- Windows,
- MacOS,
- Android,
- iOS and
- LuminOS.

Chapter 2

FAQ

```
#1 Where to find documentation for Mvx2Unity, for its extensions and for included Mvx2 framework and
      plugins?
  - this FAQ text file:
    - `/Assets/Plugins/Mvx2/FAQ.txt`
  - comprehensive documentation for the plugin and all associated subproducts:
    - `/Assets/Plugins/Mvx2/doc/{product}.pdf`
    - `/Assets/Plugins/Mvx2/doc/{product}.zip` (compressed html format)
  - primary plugin documentation:
    - `/Assets/Plugins/Mvx2/doc/Mvx2Unity.pdf`
    - `/Assets/Plugins/Mvx2/doc/Mvx2Unity.zip` (compressed html format)
#2 Where to find sample scenes and associated assets (e.g.  sample stream definitions and data
      decompressors)?
  - `/Assets/Plugins/Mvx2/Scenes/{sample scene}.unity`
  - `/Assets/Plugins/Mvx2/Scenes/{sample stream definition}.asset`
#3 How to change a stream source file?
  1.  Determine a stream source definition asset by checking a `Data stream definition` field of a stream
      component in the scene,
    1a.  e.g.  in the `Mvx2BasicSample.scene`, check `MvxSimpleDataStream` component attached to
      `DataStream` object,
    1b.  its `Data stream definition` field references a stream definition asset located at
      `/Assets/Plugins/Mvx2/Scenes/FileDataStreamDefinition.asset`.
  2.  If the stream source definition is a file stream definition, it contains a `File path` field where the
      path to a source file can be changed,
    2a.  the field accepts not only absolute file paths, but also paths relative to standard
      `StreamingAssets` Unity folder - to specify these files drag & drop a StreamingAssets file inside the
      Unity editor.
  3.  For further information about stream source definitions check the comprehensive documentation (there
      are also other stream source definition types available than those for specifying files).
#4 How to attach decompressors to the stream?
  A support for decompressors is implemented in a generic way to all stream playback implementations (i.e.
      derivatives of `MvxDataStream`):
    - data decompressors are implemented as standalone asset files
      - e.g.  `/Assets/Plugins/Mvx2/Scenes/DataAutoDecompressor.asset`,
    - there is a `Data decompressors` field present in `MvxDataStream` components, which holds a list of
      decompressor assets to attach,
    - add or remove decompressor assets referenced by the field, as required.

#5 How is support for audio playback implemented?
  There are 2 independent implementations of audio playback support:
    - simple, which is based on asynchronous frames processing
      - `MvxAudioAsyncPlayer` derivative of `MvxAsyncDataProcessor`
      - frames are checked for presence of audio data and those are extracted and filled into a native audio
      player,
      - the audio playback may not be fluent, depending on the playback speed, its smoothness, and
      distribution of audio data to individual frames,
      - most sample scenes only contain this implementation.
    - advanced, in which the audio data themselves control the playback of the whole stream ('audio channel
      is the master'),
      - `MvxAudioPlayerStream` derivative of `MvxDataStream`
      - frames are pre-processed and pre-checked for audio data presence, and cached in such a way, to allow
      fluent audio playback,
      - whenever the audio playback reaches a part of audio data present in a certain frame, this frame is
      released for further processing to the stream,
      - `/Assets/Plugins/Mvx2/Scenes/Mvx2AudioPlayerStreamSample.unity` sample scene showcases this
      implementation.
```

Chapter 3

Main Scripts

The main emphasis of Mvx2Unity scripts is on accessing and processing/rendering of Mvx2 content from Mvx2 streams. The architecture of the scripts consists of four layers (see figure below):

- **data streams** – playback of Mvx2 content, reading and notifying about Mvx2 frames,
- **data stream definitions** – access to Mvx2 data streams (MVX2 file, network, custom),
- **data decompressors** – optional data decompressors (auto, H264, SYK, ...),
- **targets** - optional stream targets (writing to MVX2 file, transmitting over a network, custom),
- **data processors** – processing of Mvx2 frames streamed by data streams (rendering, audio playback, ...).

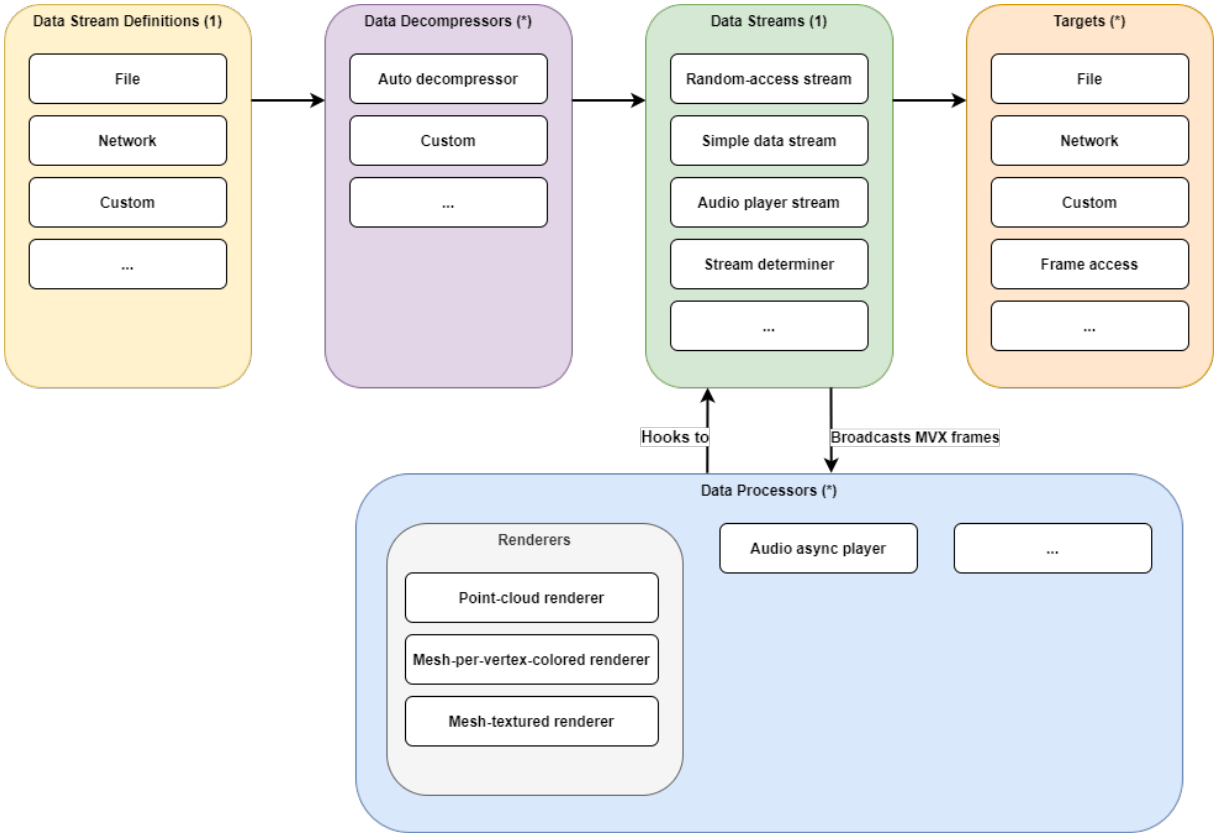

Figure 3.1 Figure: Scripts Architecture

This architecture makes it possible to have multiple independent data streams (referencing different Mvx2 sources) in a single Unity3D scene. Also, multiple (or none) data processors can be hooked to a single data stream, all listening for Mvx2 frames reported by it. Each processor can process frames in its own way.

Below follows a short description of each of the main classes implemented in the plugin:

Data Streams

MvxDataRandomAccessStream

Has no automatized reading capabilities. Instead, frames reading has to be invoked manually and each such invocation requires an index of the frame to be read.

MvxSimpleDataStream

Is based on SimpleAPI's asynchronous reader. The reader is responsible for maintaining data stream's framerate information and asynchronously notifies about new Mvx2 frames at calculated time intervals. The stream inherits the same behaviour, meaning that this type of stream maintains automatized framerate-based playback of a data stream.

MvxAudioPlayerStream

Similarly as MvxSimpleDataStream maintains automatized reading (playback) of Mvx2 frames. This one however is based on sample-rate of audio contained in a stream. The main purpose of the stream is proper synchronization of audio and graphical playback with audio being the master, which means that graphical part of an Mvx2 frame is only rendered when audio playback of audio chunk of that frame started playback. Since this is a data stream, by 'rendering' one can naturally understand 'notifying about the frame'. This type of stream can only operate with data streams which contain audio data.

MvxDataStreamDeterminer

Similarly as MvxSimpleDataStream maintains automatized reading (playback) of Mvx2 frames. The stream however is just a wrapper for two other streams - MvxSimpleDataStream and MvxAudioPlayerStream. Its responsibility is to create a nested stream based on whether a given data source does contain audio data or not. In case it does, an audio stream is created and playback is driven by audio sample-rate. In the other case, a simple stream is created and playback is driven by framerate.

Data Stream Definitions

MvxFileDataStreamDefinition

Provides Mvx2 content by reading MVX2 files.

MvxNetworkDataStreamDefinition

Provides Mvx2 content by receiving it from network sockets.

MvxCustomDataStreamDefinition

Provides Mvx2 content by receiving it from arbitrary sources. A GUID of the source is required, as well as its parameters.

Data Decompressors

MvxDataAutoDecompressor

A decompressor for decompressing data types compressed using basic generic algorithms.

MvxCustomDataDecompressor

An arbitrary decompressor. A GUID of the decompressor is required, as well as its parameters.

Data Processors

MvxMeshPointCloudRenderer

A data processor, uses only vertex positions and vertex colors data from Mvx2 frames and renders them as a point-cloud.

MvxMeshPerVertexColoredRenderer

A data processor, uses vertex positions, vertex colors and vertex indices data from Mvx2 frames and renders them as colored triangle-based meshes.

MvxMeshTexturedRenderer

A data processor, uses vertex positions, vertex indices, vertex UVs and texture data from Mvx2 frames and renders them as textured triangle-based meshes.

MvxTextureExtractor

A data processor for extraction and simple rendering of textures present in Mvx2 frames, including depth-map.

MvxAudioAsyncPlayer

A data processors, uses audio data from Mvx2 frames to play them using Unity3D's audio-playback support.

Data Stream Targets

MvxFileTarget

Stores Mvx2 content to MVX2 files.

MvxNetworkTarget

Transmits Mvx2 content to network sockets.

MvxCustomTarget

Processes Mvx2 content using arbitrary targets. A GUID of the target is required, as well as its parameters.

MvxFrameAccessTarget

Notifies about Mvx2 frames using UnityEngine's events system.

Chapter 4

Mvx2API Overview

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, as well as simple access to basic frame data.

Mvx2BasicIO is an extension module of *Mvx2API* for working with MVX2-formatted files and network streams.

Quick overview of the *Mvx2API*'s and *Mvx2BasicIO*'s 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 graph nodes for compression and decompression of data,
- provides extractors of mesh data, texture data and audio data from frames,
- provides utility for fast extraction of simple data information about MVX2 files,
- provides mesh data structures and an utility for splitting of meshes into smaller submeshes,
- provides a way for adding more features via Mvx2 framework's plugins support (see below).

These services are provided as a collection of standalone .NET libraries, so they can be used by any .NET application. Since Unity3D is also based on .NET framework, they are integrated parts of the Mvx2Unity plugin. All services provided by the *Mvx2API* and *Mvx2BasicIO* are documented in standalone reference manuals located at *Assets/Plugins/Mvx2/doc/<here>* and are not further described in this document.

Mvx2 plugins

The Mvx2 framework is designed to be modular and so it has an integrated support for plugins. Plugins are standalone libraries (dll, so, ...) containing implementations of additional filters and data layer types, loaded by the framework on demand. The framework and the *Mvx2API* provide API for creating these filters and adding them to data-processing graphs.

In Mvx2Unity plugin, the Mxv2 plugin folders per platform are

- `Assets/Plugins/Mvx2/plugins/android/arm64-v8a`,
- `Assets/Plugins/Mvx2/plugins/android/armeabi-v7a`,
- `Assets/Plugins/Mvx2/plugins/ios`,
- `Assets/Plugins/Mvx2/plugins/luminos`,
- `Assets/Plugins/Mvx2/plugins/macos_unity` and
- `Assets/Plugins/Mvx2/plugins/windows`.

Note that import settings for additional plugins need to be adjusted appropriatelly.

Chapter 5

Project Setup

Following are necessary or recommended settings of projects that use Mvx2Unity:

Allow 'unsafe' Code

- **must be enabled**, otherwise Unity fails to load Mvx2Unity scripts
- found at *Project Settings -> Project/Player -> Other Settings*
- all platforms

Chapter 6

Release Notes

2.7.5

Initial version.

2.7.6

Documentation

- **2.7.6_D1** | added 'release notes' section

Mvx2

- **2.7.6_M1** | updated `MVGraph_SimpleAPI` dependency to version 1.0.0
- **2.7.6_M2** | updated `IOSCamera` plugin dependency to version 1.0.0

3.0.0

Mvx2Unity

- **3.0.0_MU1** | switched underlying framework from `MVGraphAPI` and `MVGraph_SimpleAPI` to `Mvx2API` and `Mvx2BasicIO` (with almost the same API)

Mvx2

- **3.0.0_M1** | updated `Mvx2/Mvx2API` dependency to version 4.0.0
- **3.0.0_M2** | updated `Mvx2BasicIO` dependency to version 3.0.0
- **3.0.0_M3** | updated `Mvx2File` plugin dependency to version 3.0.0
- **3.0.0_M4** | updated `SuperNetwork` plugin dependency to version 3.0.0
- **3.0.0_M5** | updated `IOSCamera` plugin dependency to version 3.0.0

Documentation

- **3.0.0_D1** | updated documentation after switch to `Mvx2API` and `Mvx2BasicIO`

3.0.1

Mvx2

- **3.0.1_M1** | updated `Mvx2/Mvx2API` dependency to version 4.1.0

Documentation

- **3.0.1_D1** | added a section about `Mvx2` plugins to the Mvx2API Overview documentation

Mvx2Unity

- **3.0.1_MU1** | extended mesh-textured renderer (and shader) with a support for extracting and rendering NV12 and NV21 textures
- **3.0.1_MU2** | fixed crashing caused by asynchronous audio player

3.0.2

Mvx2

- **3.0.2_M1** | updated `IOSCamera` plugin dependency to version 3.0.1

3.0.3

Mvx2

- **3.0.3_M1** | updated `Mvx2/Mvx2API` dependency to version 4.2.0
- **3.0.3_M2** | updated `Mvx2BasicIO` dependency to version 3.1.0
- **3.0.3_M3** | updated `SuperNetwork` plugin dependency to version 4.0.0

3.0.4

Mvx2

- **3.0.4_M1** | updated `MVCommon` dependency to version 3.0.0
- **3.0.4_M2** | updated `Mvx2/Mvx2API` dependency to version 5.0.0
- **3.0.4_M3** | updated `Mvx2BasicIO` dependency to version 4.0.0
- **3.0.4_M4** | updated `SuperNetwork` plugin dependency to version 4.1.0
- **3.0.4_M5** | updated `MVX2File` plugin dependency to version 3.1.0
- **3.0.4_M6** | updated `IOSCamera` plugin dependency to version 3.1.0

Mvx2Unity

- **3.0.4_MU1** | replaced potentially unsafe data-extraction procedures (e.g. when garbage collector relocates data fields in memory during the extraction) by safe alternatives in following functions:
  - MVXUnity::MvxAudioPlayerStream::ExtractAudioFromFrame()
  - MVXUnity::MvxAudioAsyncPlayer::ProcessAudioData()
  - MVXUnity::MvxUtils::GetFrameBoundingBox()
  - MVXUnity::MvxTextureExtractor::LoadTexture()

Known bugs

- **3.0.4_KB1** | MVXUnity::MvxAudioAsyncPlayer causes random application crashes

3.0.5

Mvx2

- **3.0.5_M1** | updated `MVCommon` dependency to version 4.0.0
- **3.0.5_M2** | updated `Mvx2/Mvx2API` dependency to version 6.0.0
- **3.0.5_M3** | updated `Mvx2BasicIO` dependency to version 5.0.0
- **3.0.5_M4** | updated `SuperNetwork` plugin dependency to version 4.2.0
- **3.0.5_M5** | updated `MVX2File` plugin dependency to version 3.3.0
- **3.0.5_M6** | updated `IOSCamera` plugin dependency to version 3.2.0

Documentation

- **3.0.5_D1** | introduced PDF documentation as an alternative to the HTML one:
  - `Assets/Plugins/Mvx2/doc/Mvx2Unity.pdf`
- **3.0.5_D2** | introduced PDF documentations of the Mvx2Unity dependencies as alternatives to the HTML ones:
  - `Assets/Plugins/Mvx2/doc/Mvx2BasicIO{Net}.pdf`
  - `Assets/Plugins/Mvx2/doc/Mvx2{Net}.pdf`
  - `Assets/Plugins/Mvx2/doc/MVCommon{Net}.pdf`
  - `Assets/Plugins/Mvx2/doc/MVX2File.pdf`
  - `Assets/Plugins/Mvx2/doc/SuperNetwork.pdf`
  - `Assets/Plugins/Mvx2/doc/MVZMQNetwork.pdf`
  - `Assets/Plugins/Mvx2/doc/IOSCamera.pdf`

3.0.6

Mvx2Unity

- **3.0.6_MU1** | upgraded to Unity 2019.4.19f1
- **3.0.6_MU2** | fixed multiple scripts to only detect presence of uncompressed texture data types in frames before trying to extract them:
  - MVXUnity::MvxTextureExtractor
  - MVXUnity::MvxMeshTexturedRenderer
  - MVXUnity::MvxDepthmapPointCloudRenderer
- **3.0.6_MU3** | refactored MVXUnity::MvxDataStream:
  - removed `mvxSourceInfo` getter (any data and data profile queries based on source info are unreliable, because processing pipelines may completely change frames' data)
  - added `framesCount` for getting count of frames of the open stream
  - `onStreamOpen` event no longer references a source info of the open stream
  - MVXUnity::MvxDataStream derivatives refactored accordingly:
    - MVXUnity::MvxDataReaderStream
    - MVXUnity::MvxDataStreamDeterminer
- **3.0.6_MU4** | updated all frame processors to access data and make support decisions based only on uncompressed data presence in individual frames to process, instead of taking compressed data into consideration as well, since those may never become uncompressed (depending on the specific pipeline that preceeds frame access):
  - MVXUnity::MvxTextureExtractor
  - MVXUnity::MvxAudioAsyncPlayer
  - MVXUnity::MvxMeshPerVertexColoredRenderer
  - MVXUnity::MvxMeshPointCloudRenderer
  - MVXUnity::MvxMeshTexturedRenderer
- **3.0.6_MU5** | added support for preventing frame processors from processing a frame in case another referenced frame processor is able to process the frame
  - all frame processors have processorsPreventingThisFromProcessing collection visible in Inspector
  - use case example: render frame as a point-cloud only in case mesh-textured renderer lacks necessary data in the frame
- **3.0.6_MU6** | added support for explicit setup of data decompressors (via Inspector), appended to streams right behind their data source
  - there are two basic implementations of decompressors:
    - MVXUnity::MvxDataAutoDecompressor
    - MVXUnity::MvxCustomDataDecompressor
  - data streams no longer auto-add Mvx2API::AutoDecompressorGraphNode to their pipelines
  - affects all data stream types:
    - MVXUnity::MvxAudioPlayerStream
    - MVXUnity::MvxSimpleDataStream
    - MVXUnity::MvxDataRandomAccessStream
    - MVXUnity::MvxDataStreamDeterminer
    - MVXUnity::MvxDataStreamThumbnail
  - updated data streams of all scenes to append an auto decompressor to their data pipelines:

    * Assets/Plugins/Mvx2/Scenes/iOSCameraStreamSample.unity
    * Assets/Plugins/Mvx2/Scenes/Mvx2AudioPlayerStreamSample.unity
    * Assets/Plugins/Mvx2/Scenes/Mvx2BasicSample.unity
    * Assets/Plugins/Mvx2/Scenes/Mvx2RandomAccessStreamSample.unity
    * Assets/Plugins/Mvx2/Scenes/Mvx2SimpleStreamSample.unity
    * Assets/Plugins/Mvx2/Scenes/Mvx2StreamDeterminerSample.unity
    * Assets/Plugins/Mvx2/Scenes/Mvx2TextureExtractorSample.unity

Extensions

- **3.0.6_E1** | introduced standalone documentation files (including release notes) per each extension of the core Mvx2Unity package
  - new documentation files reside in Assets/Plugins/Mvx2/doc folder
  - changes of the extensions will no longer be documented in the core package release notes

Documentation

- **3.0.6_D1** | added Project Setup section
- **3.0.6_D2** | updated Main Scripts documentation section

3.1.0

Mvx2Unity

- **3.1.0_MU1** | extended MVXUnity::MvxTextureExtractor with a support for extracting NVX, NV12 and NV21 textures
- **3.1.0_MU2** | introduced MVXUnity::MvxDataStreamSourceRuntime base class
  - instance is created by stream definitions and is held by data streams during the playback
  - data streams stop their playback when source runtime indicates end of the stream via MVXUnity::Mvx↩DataStreamSourceRuntime::StreamEnded() function
- **3.1.0_MU3** | refactored MVXUnity::MvxDataStreamDefinition:
  - function GetSourceGraphNode() replaced with AppendSource(Mvx2API.ManualGraphBuilder graph↩Builder)
  - derivatives can append multiple graph nodes to a graph being built
  - derivatives can return a custom source runtime instance (see **3.1.0_MU2**)
- **3.1.0_MU4** | turned MVXUnity::MvxFPSCounter base class into an abstract class
- **3.1.0_MU5** | introduced MVXUnity::MvxLoggerSink::logLevel field so the log level can easily be switched from Unity inspector
- **3.1.0_MU6** | refactored MVXUnity::MvxFileAccessor to access `StreamingAssets` files in a default way on all platforms except on Android
  - previously an exception was thrown on platforms other than Android, Windows, MacOS, iOS and LuminOS
- **3.1.0_MU7** | refactored custom preparation of `StreamingAssets` files by MVXUnity::MvxFileAccessor on Android platform
  - obsolete `UnityEngine.WWW` utilized for accessing the files replaced by `UnityEngine.`↩`Networking.UnityWebRequest`
- **3.1.0_MU8** | refactored MVXUnity::MvxAudioAsyncPlayer to extract audio data from frames immediatelly from the main Unity thread instead of utilizing a standalone thread
  - fixes occasional crashes when MVXUnity::MvxAudioAsyncPlayer is part of the scene

Documentation

- **3.1.0_D1** | added FAQ section
- **3.1.0_D2** | added standalone FAQ.txt file:
  - Assets/Plugins/Mvx2/FAQ.txt

4.0.0

Mvx2Unity

- **4.0.0_MU1** | consolidated all plugins into one Unity package for simpler importing
- **4.0.0_MU2** | added assembly definitions to separate Mvx2Unity from the rest of the project and allow unsafe code without changing project settings
- **4.0.0_MU3** | added option for MvxDataStream to asynchronously initialize and dispose graph to prevent stutters while using a stream definition that takes longer to initialize (such as FileStream)
  - added toggle UseAsyncInit
  - refactored OpenReader and DisposeReader to use Unity Coroutine which internally utilizes Task if enabled
- **4.0.0_MU4** | added new API to MvxDataStream:
  - autoPlay - option to enable automatic play/restart on script start or stream definition change
  - isPaused - indicates whether the playback is paused
  - playbackSpeed - a modifier for playback speed ranges from 0x to 3x
  - onPlaybackRestarted - event which occurs when the playback has looped/restarted
  - onPlaybackEnd - event which occurs when the playback has ended on last frame
  - lastReceivedFrameNr - last received frame number
  - sourceFPS - original framerate of the clip
- **4.0.0_MU5** | added new API to MvxAudioPlayerStream and MvxDataStreamDeterminer:
  - mute
  - volume
- **4.0.0_MU6** | refactored followStreamFPS to playAtMaxFPS to be easier to understand and more representative of its function alongside of playbackSpeed
- **4.0.0_MU7** | changed all Stream Definitions to be MonoBehaviour to match common design patterns in Unity. This now requires the definition to be added to a GameObject as a component instead of ScriptableObject .asset in project files.
- **4.0.0_MU8** | added ByteStreamDefinition that reads mvx data from TextAsset or byte[] (Unity 2021.2+ TextAsset supports reading directly from NativeArray without allocating new byte array and copying data)
- **4.0.0_MU9** | added prefabs for DataStream players and StreamDefinitions
- **4.0.0_MU10** | added automatic loading of decompressors from Resources
- **4.0.0_MU11** | added new Sample Scenes using the prefabs
- **4.0.0_MU12** | refactored custom inspector properties for stream definitions to use generic functions and getter/setter delegates to simplify adding parameters
- **4.0.0_MU13** | added editor script for downloading sample MVX file whenever a sample scene is opened (and it is not downloaded yet)
- **4.0.0_MU14** | fixed shaders for linear color space
- **4.0.0_MU15** | added shadowcaster pass to mesh shaders

4.0.1

Mvx2Unity

- **4.0.1_MU1** | added ForceDisposeStream to address hang on application exit
- **4.0.1_MU2** | added SerializeField to audio source of MVX Audio Streams to allow references to custom audio sources
- **4.0.1_MU3** | updated RYSK decompressor GUID to use new sync variant on all platforms
- **4.0.1_MU4** | removed CreateAssetMenu for all Stream Definitions causing warnings in editor

4.0.2

Mvx2Unity

- **4.0.2_MU1** | added platform support for Apple Silicon and transitioned from bundles to universal dylib.
- **4.0.2_MU2** | added platform support for Android x86_64
- **4.0.2_MU3** | fixed pointcloud shader on Metal

4.0.3

Mvx2Unity

- **4.0.3_MU1** | fixed wrong colorspace conversion for textures using ASTC, DXT and ETC compression
- **4.0.3_MU2** | added RestartStream and RestartPlayback to MvxDataStream
- **4.0.3_MU3** | corrected windows plugins dll import settings

4.0.4

Mvx2Unity

- **4.0.4_MU1** | added first time import dialog that prompts user to open sample scene
- **4.0.4_MU2** | moved sample scenes to Assets/Mvx Sample Scenes and renamed them for better visibility
- **4.0.4_MU2** | removed AudioAsyncPlayer from simple sample scene to discourage its use with clips containing audio
- **4.0.4_MU4** | added canvas texts to sample scenes describing the mvx components in the scene
- **4.0.4_MU5** | added custom lit shader for textured mvx only with ambient and diffuse light component. Supports multiple light sources (all types), shadow casting and shadow receiving
- **4.0.4_MU6** | moved texture conversions in mvx shaders into separate file SampleMvxTexture.cginc
- **4.0.4_MU7** | added shader parameter _UseNormals set by MvxMeshTexturedRenderer if mvx clip contains normals. Needed for correct behaviour of the custom lit shader

4.1.0

Mvx2Unity

- **4.1.0_MU1** | Android libraries built with NDK25 and static libc++ to be compatible with Unity 2023, or other libraries that might have used libc++_shared
- **4.1.0_MU2** | reworked how MvxAudioPlayerStream queues mesh frames to be presented to be compatible with Unity 2023. The mesh is no longer shown exactly when audio for that frame was being read. Instead it is queued up and presented at correct time depending on separate timer. Audio time is used to correct desync.
- **4.1.0_MU3** | fixed colorspace fix not needed for RGB texture
- **4.1.0_MU4** | fixed some scenes having invalid lighting settings assigned which caused crashes on newer Unity versions
- **4.1.0_MU5** | removed android target sdk version from manifest xml
- **4.1.0_MU6** | added workaround for audioSource settings in Unity 2023 not applying to generated audio
- **4.1.0_MU7** | refresh AssetDatabase after sample mvx was downloaded to not trigger download dialog again

Chapter 7

Sample Scenes

There are multiple sample scenes showing typical setup of the plugin's scripts and various usages of Mvx2API:

Mvx2BasicSample

Demonstrates a basic file data stream with textured renderer and audio player.

Mvx2AudioPlayerStreamSample

Demonstrates a behaviour of *MvxAudioPlayerStream*, where rendering is synchronized with audio playback (with audio being the master). There are three switchable independent renderers in the scene.

Mvx2SimpleStreamSample

Demonstrates a behaviour of *MvxSimpleDataStream*, where rendering is based on MVX2 file's framerate. There are three switchable independent renderers in the scene and also an asynchronous audio player (playing audio data of Mvx2 frames reported by the stream, based on frame-rate).

Mvx2StreamDeterminerSample

Demonstrates a behaviour of *MvxDataStreamDeterminer*, which decides which actual data stream (framerate-based or sample-rate-based) to use depending on whether an MVX2 file contains audio data or not.

Mvx2RandomAccessStreamSample

Demonstrates a behaviour of *MvxDataRandomAccessStream*, where a UI slider can be used to adjust which Mvx2 frame to render. This scene furthermore shows how a custom source definition (*MvxCustomData↩StreamDefinition*) can be used for specifying arbitrary sources with a known GUID and parameters protocol.

Mvx2TextureExtractorSample

Demonstrates a behaviour of *MvxSimpleDataStream* and two *MvxTextureExtractor* instances, which extract and render depth-map and RGB data from processed frames.

Chapter 8

Utilities

Editor-mode 3D thumbnail

- There is a special data stream implemented (*MvxDataStreamThumbnail*) operating with data streams in editor-mode of Unity3D. It allows previewing Mvx2 content (single, but any frame) without the necessity to run a scene. It can be used as a placeholder for proper placement of player-mode renderers in a scene. When play-mode is activated, all Unity3D objects created by the thumbnail script are destroyed.
- The simplest way to add a thumbnail to scene is via menu *GameObject/Mvx2/Thumbnails/<thumbnail with desired rendering type>*. A stream definition has to be specified on the game object afterwards.

Mvx2 logger sink

- It may be difficult to debug Mvx2 framework's behaviour when used in Unity3D applications. There is a script named MvxLoggerSink, which makes it possible to see logs generated by Mvx2 framework directly in the Unity3D's console. These logs are often descriptive enough for successful debugging. The logger sink should not be active in a deployed application to preserve memory and computational resources.
- As a component it has be added to any game object in a scene.