DEMOLITION

STUDIOS

Demolition Media Hap for Unity

version 0.9.6

Contents

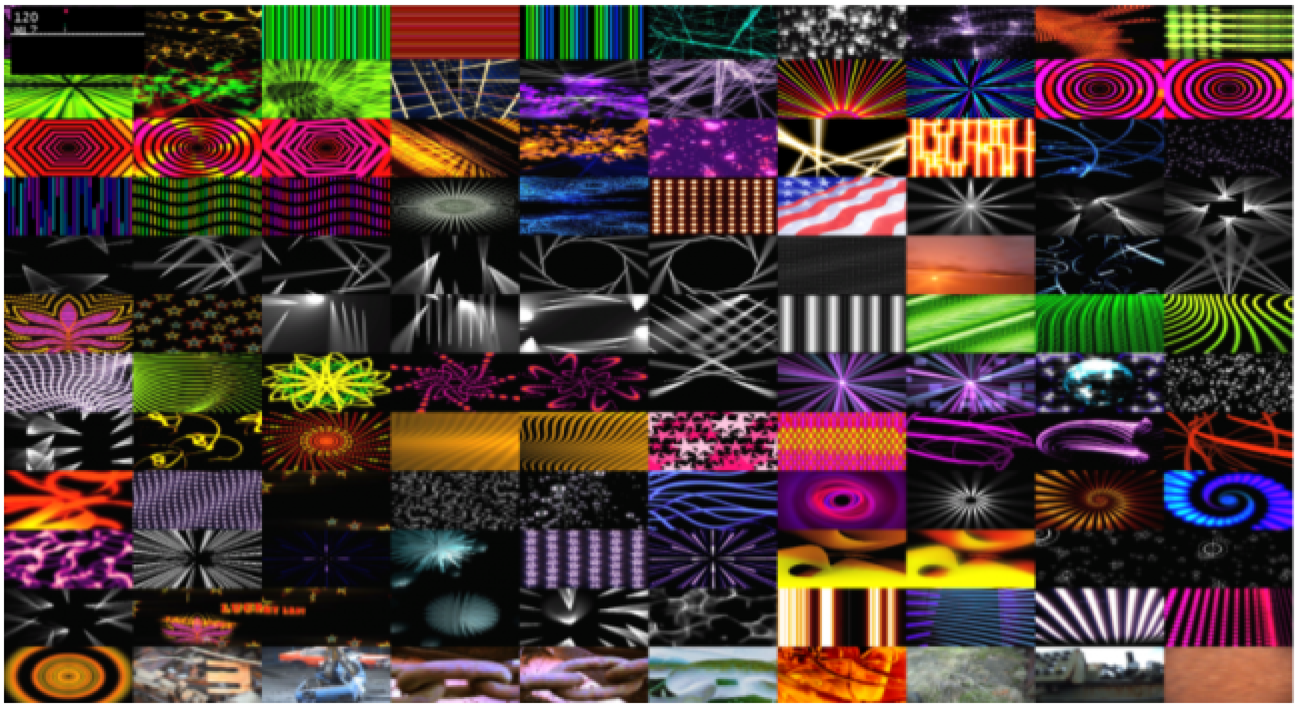
1. General information

1.1 The story

This plugin is the only one solution which managed to provide **lag-free high-framerate video playback** for The Complex mixed reality playscape. We have Unity running on **8 FullHD projectors with 6k video resolution**, which is **not nearly a limit**. We tried other solutions which present on the Asset Store, but they didn't work flawlessly. Even expensive ones. Finally, we made this plugin. It proved to be a rock-stable video playback solution and ass-saver many many times now.

1.2 Why and when to use HAP

The Hap video codec is specially designed for playing high-resolution videos on desktop platforms:
— You can play 8k and above with fairly low CPU usage, since it uses GPU decompression
— Seeking/scrubbing with Hap works very fast!
— Support for an optional alpha channel in the Hap Alpha codec
— Reduced data throughput to graphics hardware


Playing 120 SD videos at once with Hap codec (equals to a single 9600x3840 video)

The only downside of the Hap codec is that it produces quite large files. For situations where a computer can't handle playing back movies because the CPU usage is too high, using Hap may make it possible to reduce the overhead of playback. Additionally, as most movie formats do not have support for an alpha channel, the Hap Alpha can be a real saver.

Typical Hap codec use-cases include (but not limited to):
— Media installations
— Interactive playscapes
— VJ performances
— VR projects
— etc etc

Read more about Hap here and here.
You can find detailed instructions of how to prepare Hap-encoded videos in section 3.

1.3 Plugin features

— **Hardware accelerated HAP** video playback **without any external codecs** needed
— Low CPU/memory usage. Frames are **decompressed on the GPU**
— Play **10k @ 50 fps** (new - see section 3.3) videos, play multiple videos at once, with extremely fast frame-precise seeking
— **Transparent videos** with Hap Alpha and Hap Q Alpha codecs.
— Works with majority of graphics APIs on desktop platforms: **DX9, DX11, OpenGL, Metal**
— **Chunked Hap** support for even faster decoding
— Both **32-bit** and **64-bit** builds supported! Even on OS X!
— Audio output through the Unity **Native Audio API** plugin and **AudioSource**
— Suits for both programmers and artists: **C#** API, **IMGUI/uGUI** wrappers provided
— Example scenes with the typical usage scenarios

1.4 Requirements

— Unity 2017, Unity 2018 or Unity 2019 (might still work in 5.x)
— Windows 7/8/10 (32-bit and 64-bit)
— OS X 10.9 and above (32-bit and 64-bit), 10.11 and above for Metal rendering.
— Linux plugin build is still missing, although it’s possible to do.

1.5 Demo version

There is a demo version available, so you could test everything before you buy. After you are done with testing the demo version and satisfied with it, once the full version is bought on the asset store, it should seamlessly drop-in replace the demo version. They differ only in the native plugin code, but the rest (C# scripts, resources, etc) are the same in both versions.

The demo version has a glitchy watermark effect which can appear periodically in the video, and usually look like horizontal bars. You can find an example of how exactly does it looks in the appendix of this document.

1.6 (new) Native plugin upgrade instructions on Windows

These are the steps which we recommend to take in order to to replace the native plugin .dll files on Windows with newer ones:

1. Make sure your project isn't opened inside the Unity Editor
2. In Explorer go to `<your project root folder>/Assets/DemolitionMedia`
3. Remove the `Plugins` folder completely (you've read it right, just delete it!)
4. Open your project in the Unity Editor
5. Import the latest Demolition Media Hap package (which is on the Asset Store)

At this point it should be updated and work! To make the audio work, restart the Unity Editor

1.7 Support

We offer support using e-mail address demolition.studios.rocks@gmail.com. Really, if you have any problems just write me and I will try to answer ASAP. If you've bought the plugin on the asset store (thank you!) don't hesitate to include your invoice number in the mail body. You can ask anything about the demo version too, of course.

If you think that something isn't working, you can also post an issue on github.

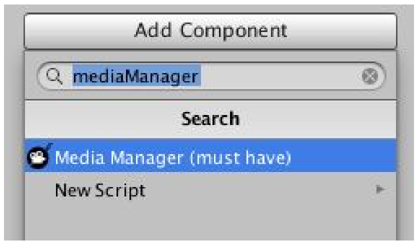
2. Usage

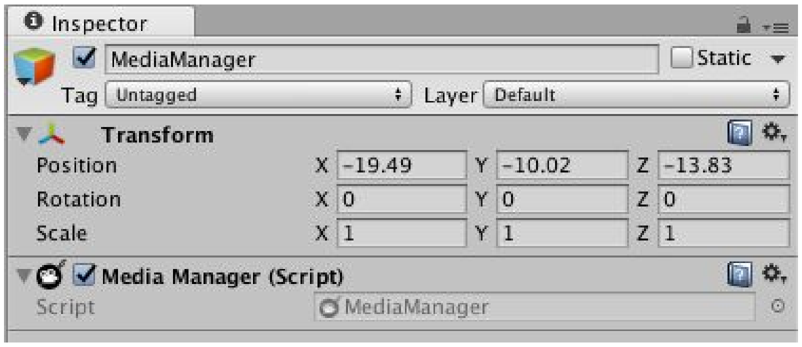
Watch this video for the step by step usage instructions!

2.1 Basic IMGUI usage

Follow these instructions to get a basic scene with IMGUI video player working:

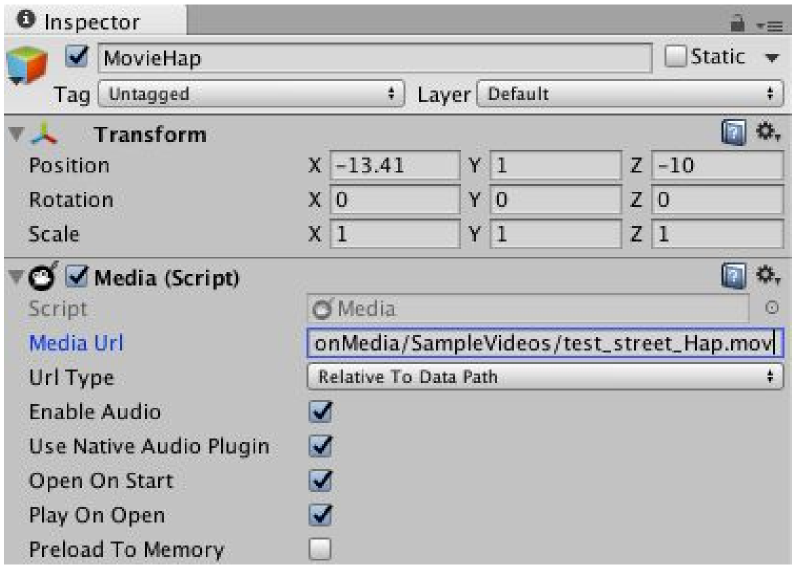
1. Create a new scene
2. Add an empty object to the scene, attach `MediaManager` script to it.





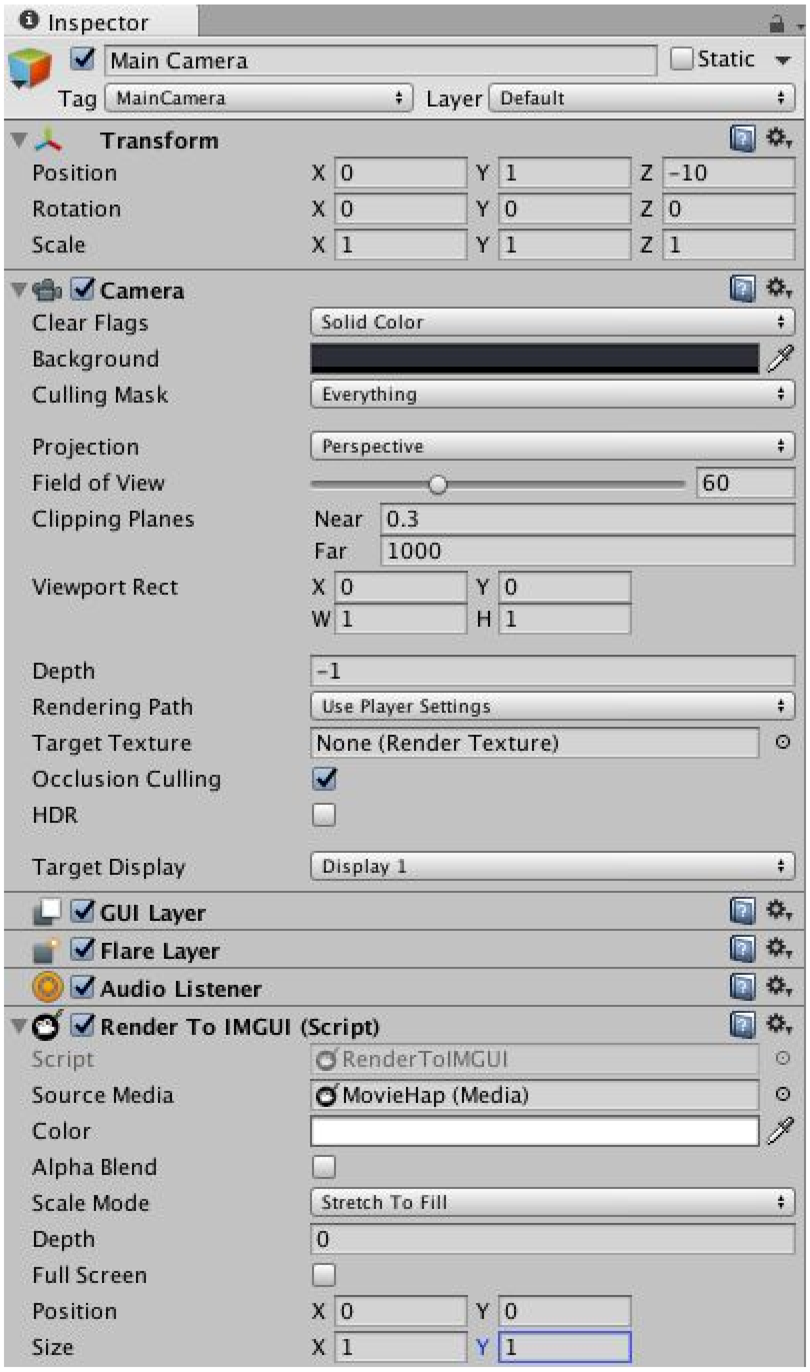
MediaManager
Tag Untagged Layer Default
Transform
Position X -19.49 Y -10.02 Z -13.83
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Media Manager (Script)
Script MediaManager

3. Again, add an empty object, attach `Media` script to it.
   Fill in `Media Url` (path to the video), along with the needed `Url Type`.



MovieHap
Tag Untagged Layer Default
Transform
Position X -13.41 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Media (Script)
Script Media
Media Url onMedia/SampleVideos/test_street_Hap.mov
Url Type Relative To Data Path
Enable Audio ✓
Use Native Audio Plugin ✓
Open On Start ✓
Play On Open ✓
Preload To Memory ☐

4. Attach `Render to IMGUI` script to Main Camera.
   Select the `Source Media` component of type `Media` created on step 3.



Inspector

✓ Main Camera	☐ Static
Tag MainCamera	Layer Default

Transform

Position	X 0	Y 1	Z -10
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

✓ Camera

Clear Flags	Solid Color
Background	■
Culling Mask	Everything
Projection	Perspective
Field of View	60
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	-1
Rendering Path	Use Player Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
HDR	☐
Target Display	Display 1

✓ GUI Layer

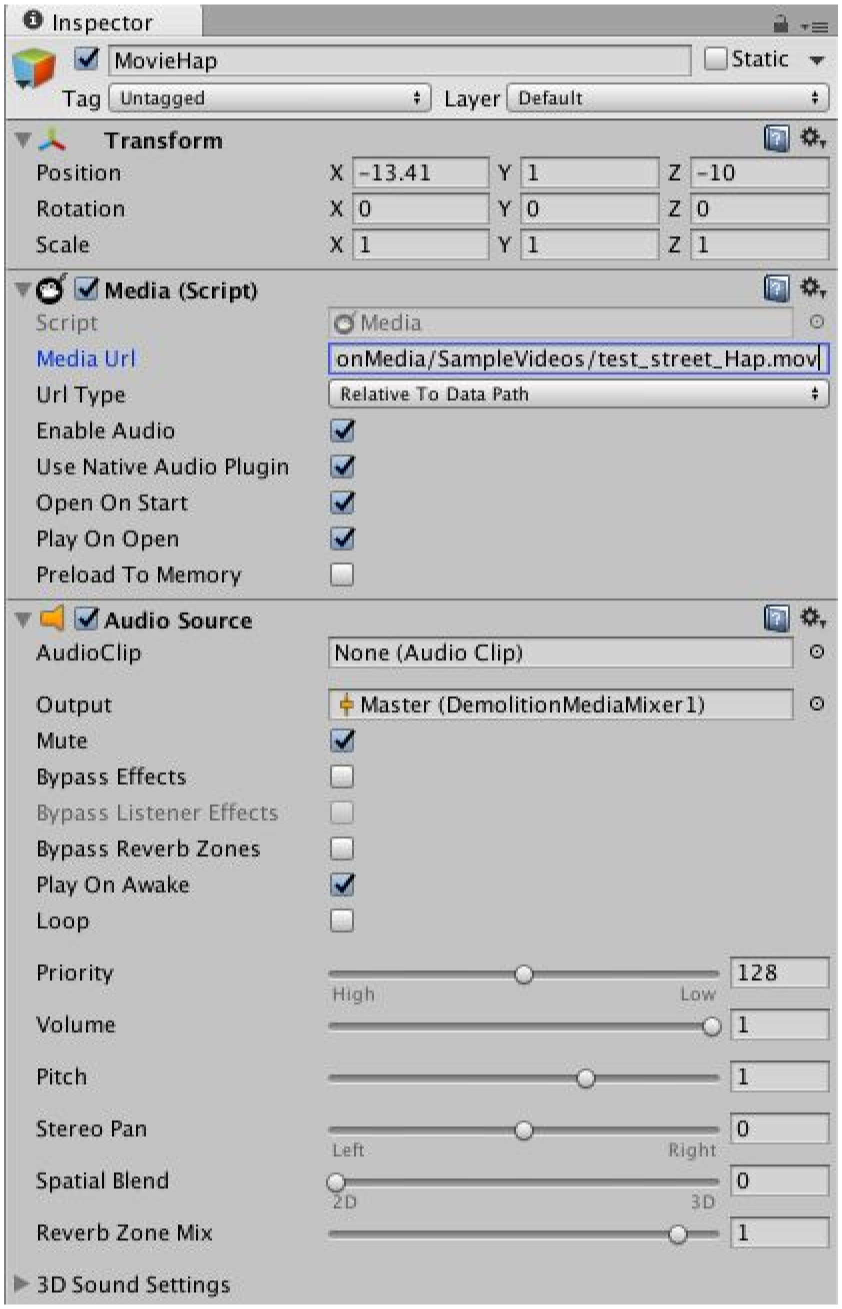
✓ Flare Layer

✓ Audio Listener

✓ Render To IMGUI (Script)

Script	RenderToIMGUI
Source Media	MovieHap (Media)
Color	□
Alpha Blend	☐
Scale Mode	Stretch To Fill
Depth	0
Full Screen	☐
Position	X 0 Y 0
Size	X 1 Y 1

5. To get the audio working, attach an `Audio Source` component to the object with `Media`. Output should be set to “Master (DemolitionMediaMixer1)”



MovieHap ☐ Static
Tag: Untagged   Layer: Default

Transform

	X	Y	Z
Position	-13.41	1	-10
Rotation	0	0	0
Scale	1	1	1

Media (Script)

Script	Media
Media Url	onMedia/SampleVideos/test_street_Hap.mov
Url Type	Relative To Data Path
Enable Audio	✓
Use Native Audio Plugin	✓
Open On Start	✓
Play On Open	✓
Preload To Memory	☐

Audio Source

AudioClip	None (Audio Clip)
Output	Master (DemolitionMediaMixer1)
Mute	✓
Bypass Effects	☐
Bypass Listener Effects	☐
Bypass Reverb Zones	☐
Play On Awake	✓
Loop	☐
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

3D Sound Settings

This is the recommended way, since it provides the high-performance audio output.

6. Optionally attach the `Keyboard controls` script to get the basic keyboard controls working (space = play/pause, left/right arrows = jump forward/backward)

2.2 Playback speed

To control the playback speed use the `Media.PlaybackSpeed` property in the C# API.
See `RenderToIMGUIWithControls.cs` for an usage example.

Please consider the following when using this feature:
1. **It's recommended to disable the audio** if the playback speed differs from the default value of 1.0, otherwise the audio/video won't be in sync (audio stream is currently always being played with default speed), and you will need to wait until the audio is fully played until a new loop will be started.

2. Valid playback speed range is from 0.5 to X, where X depends on your PC capabilities and video bitrate. To determine the situation where the playback speed is a no-go for your PC, it's recommended to have `Media.FramedropEnabled` property set to true (default behavior), and check the number of dropped frames with `Media.GetFramedropCount`. If the framedrop count increases while the video is being played, this means that your computer isn't capable of playing the video with the active playback speed and some frames are being skipped by the decoder.

2.3 (new) Looping

For infinitely looping video playback just set Media.Loops property to a negative value, i.e.:
```
Media media = getMedia();
media.Loops = -1; // Play video in infinite loop
```

To play the video only once, set the loop count to 1:
```
media.Loops = 1; // Play video once
```

You can also play the video an arbitrary number of times before the playback stops:
```
media.Loops = 5; // Play video 5 times, then stop
```

2.4 Advanced usage

For more advanced usage scenarios (uGUI, render to material) check out out the example scenes provided with the asset!

3. Preparing HAP files

3.1 Hap formats

There are three main Hap codecs:

1. **Hap**. Offers the lowest data rates for playing back the most clips at a time. Gives reasonable image quality
2. **Hap Q**. Offers improved image quality at a higher data-rate (larger file sizes)
3. **Hap Alpha**. Has similar image quality to Hap, and supports an alpha channel
4. **Hap Q Alpha**. Improved image quality as well as alpha channel support

Hap files are usually large and require a fast hard drive or solid state drive (SSD) for playback. Read more about hap codecs here and here.

There are several ways how to produce Hap-encoded video files.

3.2 Encoding with QuickTime codec

Download and install the "official" QuickTime Hap codec from https://github.com/Vidvox/hap-qt-codec/releases (on Windows install QuickTime in prior)

Please note that you don't have to do it to be able to play Hap in Unity using our plugin.

If you were lucky and managed to install both QuickTime and Hap QuickTime codec, you can use software like Adobe After Effects, Adobe Premiere or QuickTime (Pro version needed) for encoding your videos (just select the needed Hap codec in output parameters).

Possible problem: when exporting video from After Effects with audio stream, playback may be inconsistent.
Why: We need to resample the audio data to a format suitable for Unity audio engine, and that may be a CPU-heavy operation.
Solution: please re-encode with FFmpeg using the "`-vcodec copy`" option (will only process the audio stream, leaving video frames as is). It will be played fine then. See 3.3 for details.

3.3 Encoding with FFmpeg

The other way (and probably the simplest one) is to encode videos using FFmpeg.
Get Windows binaries: https://ffmpeg.zeranoe.com/builds/
Get OS X binaries: http://evermeet.cx/ffmpeg/ or http://www.ffmpegmac.net/

After downloading the FFmpeg builds, use the following commands to encode Hap videos.

- To encode with **Hap**
  ffmpeg -i movie.mov -vcodec hap -format hap movie_hap.mov
- To encode with **Hap Q**
  ffmpeg -i movie.mov -vcodec hap -format hap_q movie_hapq.mov
- To encode with **Hap Alpha**
  ffmpeg -i movie.mov -vcodec hap -format hap_alpha movie_hap_alpha.mov
- **Important:** when encoding high-resolution videos it's recommended to use the so called chunked encoding mode:
  ffmpeg -i movie.mov -vcodec hap -format hap -chunks 8 movie_hap_chunked.mov

  This will convert the input movie to Hap Q-encoded movie using 8 chunks, which means that Unity can decode the movie using 8 threads simultaneously, giving a major performance gain, as long as the storage is able to provide needed throughput.

- (New) To **re-encode audio stream only** (make sure the video codec is already Hap)
  ffmpeg -i hap_in.mov -vcodec copy -acodec aac -b:a 320k hap_out.mov

  This will give you 320kbps AAC audio stream in the output file. The video stream quality will be preserved as in the input video.

- (New) How to achieve the **maximum performance** out of the player on high resolutions

  Try encoding the video without 2nd stage compression algorithm. You can save CPU time with that, at the cost of higher data-rate (it also depends on the content) -- your SSD should be fast!
  ffmpeg -i movie.mov -vcodec hap -format hap -compressor none movie_hap.mov

  I've managed to play 10k@50fps with this approach on my laptop. Will also work for the rest of Hap formats (Hap Q, Hap Alpha, Hap Q Alpha)

3.4 More encoding ways

You can use these applications to encode Hap videos as well:

1. HapInAVFoundation (Mac only batch converter, can produce Hap Q Alpha!)
2. TouchDesigner (you can even bake some real-time footage with it! Read more)
3. More on Hap Alpha encoding

4. (new) FAQ/Troubleshooting

Q0: I got **Open Failed: FindCodecError** when opening the video file.
A: Are you trying to open mkv / mp4 h264 / h265 / vp8 / vp9 / prores / network stream? It's not possible with this plugin! We only play Hap-encoded videos inside mov or mp4 container (mov is preferred) with it, which allow to use GPU acceleration and fast frame-precise seeking. Please see section 3 to find out how to encode Hap videos!

Q1: QuicktimeTools.exe is crashing while importing the Hap videos.
A: Don't worry, this is expected and won't affect the playback. Just press OK to exit the error reporting dialog. The videos ain't being played using QuickTime or Unity internal player facilities, we have our own codecs for them included.

Q2: I build one of the **example scenes**, but the video isn't playing when running the build! E.g. in 02_Hap_ControlsIMGUI scene I see `OpenFailed: OpenInputError`
A: Since Unity Asset submission rules are kinda strict, we couldn't place the videos to the `StreamingAssets` folder. To build workable example scenes, you have at least 2 options:

I. Using StreamingAssets
   1. Copy `<project_root>/Assets/DemolitionMedia/SampleVideos` folder to `<project_root>/Assets/StreamingAssets/DemolitionMedia/SampleVideos`
   2. In the Editor change the `Url Type` of the `MovieHap` object's `Media` component to `Relative To Streaming Assets Path`
   3. Build the scene

II. Manually copy files after the build
   1. In the Editor change the `Url Type` of the `MovieHap` object's `Media` component to `Absolute`
   2. Build the scene
   3. Copy `<project_root>/Assets/DemolitionMedia/SampleVideos` folder to `<build_root>/DemolitionMedia/SampleVideos`

Q3: I couldn't open the example scenes in my Unity Editor.
A: Are you using an old Unity version? 5.5 or older? The scenes are using the new file format from Unity 5.6 (yeah, that sucks that it's being broken from version to version), but the plugin itself and all the scripts should work (just follow the usage instructions). If something doesn't work (shaders, materials), please let me know (you can post an issue on github or write me a e-mail).

Q4: When I export video from **After Effects with audio stream**, playback is broken/inconsistent.
A: We need to resample the audio data to a format suitable for Unity audio engine, and that may be a CPU-heavy operation. Please re-encode the video file with FFmpeg using the `-vcodec copy` option (will only process the audio stream, leaving video frames as is). It will be played fine then. See 3.3 for details (re-encode audio stream only section)

Q5: I need to synchronize the playback across multiple network render clients. I want to control the playback by using my own external clock source (e.g. LTC timecode). I want to manually specify index of the frame to display.
A: This pro feature isn't included to the Asset Store package. Please write us, if you're interested in this.

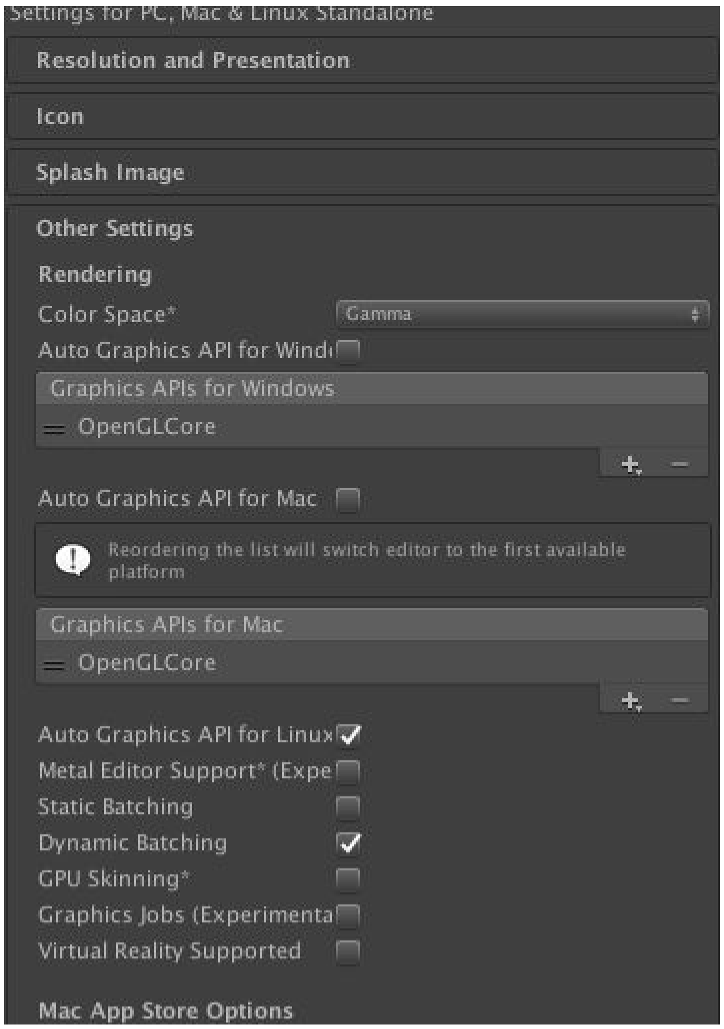
Q6: Plugin not works on OS X 10.9 or 10.10
A: Since the Metal graphics API is only shipped with OS X 10.11 SDK and above, the plugin version provided on the Asset Store may not work on older versions. Please send me a mail, so I could send you a plugin version for older OS X.

Q7: On OS X, the video playback works only in editor. If I build the Unity scene, the video texture is always black.
A: You're probably running OS X 10.12 or there is other Metal-related problem. The solution is the following:

1. In the Editor go to `Edit -> Project Settings -> Player`
2. Expand the Other Settings section
3. Uncheck Auto Graphics API for Mac and put OpenGLCore on top of Metal.

Update: may freeze Unity for some Macs, currently investigating

You can also safely remove Metal:



Settings for PC, Mac & Linux Standalone

Resolution and Presentation

Icon

Splash Image

Other Settings

Rendering

Color Space* Gamma

Auto Graphics API for Wind[ ]

Graphics APIs for Windows

OpenGLCore

Auto Graphics API for Mac [ ]

Reordering the list will switch editor to the first available platform

Graphics APIs for Mac

OpenGLCore

Auto Graphics API for Linux [✓]

Metal Editor Support* (Expe [ ]

Static Batching [ ]

Dynamic Batching [✓]

GPU Skinning* [ ]

Graphics Jobs (Experimenta [ ]

Virtual Reality Supported [ ]

Mac App Store Options

Now it should be fine in the build!

Q8: I have several videos and want to change using C# the one which is being rendered using `RenderToMeshMaterial`. When I change the `SourceMedia`, I see a blank white or black frame before my video is actually being played. The code is [similar to this](#).

A: ~~Please make sure that the `RenderToMeshMatereal`'s `FallbackTexture` property is set to a transparent texture (a texture with alpha = 0). This is the texture which is used when no video frame is available for render yet. You can either create it programatically or specify a file texture from the Unity Editor.~~

`FallbackTexture` is now transparent by default. The only thing you have to do to to ensure that the video will be transparent when it doesn't have a frame to show, is check that the mesh material shader is set to `Unlit/Transparent`.

Q9: Video playback isn’t smooth
A: First of all, it's important to find the bottleneck if the playback isn't smooth: it could be either CPU or SSD speed.

To quckly check whether you exceed your SSD capabilities, take the overall video file size and divide it by its duration in seconds: you will get the required read speed in Mb/s. Then compare it to the benchmarked values: CrystalDiskMark or any other tool should work.

Also I recommend using the following criteria to decide:
1. If you enable "Preload To Memory" for the Media component and then things work well, then it's a SSD problem and you need either preload to memory or buy faster SSD or use lower data-rate (e.g. switch to HapQ->Hap). Tip: use a file that will fit into the memory, i.e. a shorter version
2. Good indication of CPU bottleneck is constantly increasing number of dropped frames (both kinds) -- see the “IMGUI with controls” example for that.

If the bottelneck is SSD then you can do following:
- switch from Hap Q to Hap
- lower the resolution/framerate
- use NVMe SSD which give 3Gb/s+ speed

If the bottleneck is CPU:
- disable 2nd stage compression (snappy)
  example of FFmpeg command line for encode
  ffmpeg.exe -i in.mov -vcodec hap -format hap -compressor none out.mov
  The data-rate will be constant and also higher compared to snappy compressor, but at least twice less CPU work will be required before the frame gets into GPU memory! We’ve managed to play 10k@50fps and 12k@25fps using this trick on a hi-end gaming laptop
- switch from Hap Q to Hap
- lower the resolution/framerate

Q10: I want to play 8192x8192 8k video, why isn’t it working?
A: 8192x8192 isn't exactly 8k, it's 2x8k actually (considered that 8k is 7680x4320)
It's very close to the practical limits of what's possible with the plugin. But could be possible depending on your PC specs. You need really fast CPU in addition to GPU for that, and also speedy SSD drive (I recommend using NVMe, it can give up to 3Gb/s). Also see Q9. Maybe you want to try 8k@30fps first, just for sanity check.

Q10: There is a visible color banding on the video after converting to Hap.
A: Try adding noise or texture to the video until banding breaks down.
https://github.com/Vidvox/hap-qt-codec/issues/35

5. Changelog

0.9.1 (released 13.02.2017)
— Initial version

0.9.3 (released 27.03.2017)
— Playback speed can be changed now
— Framedrop can be enabled or disabled using the C# API
— Overall stability improvements
— Fix resource leaks (GPU and CPU)
— Updated the IMGUI example scene: set active segment, playback speed, etc
— New Hap Alpha example video

0.9.5 (released 31.08.2017)
— Unity 2017 support
— Support for videos with non-multiple of 2 resolutions
— Added a note how to encode a video for 10k@50fps playback
— Frame-precise seeking
— Fixed wrong computation of frame index in some cases
— Fixed native plugin not working on x86 architecture in some cases
— Playback will stop on the last frame is looping is disabled, not on the first one
— `RenderToMeshMaterial` reworked and improved
— `RenderToMaterial` and `RenderToMeshMaterial FallbackTexture` is transparent by default (can change videos one on top of another without blanking)
— Documentation is now somewhat more complete (new section: FAQ)
— Added native plugin upgrade instructions for Windows

0.9.6 (released 26.09.2019)
— Unity 2018/2019 support