Demolition Media Hap for Unity

version 3.2.0

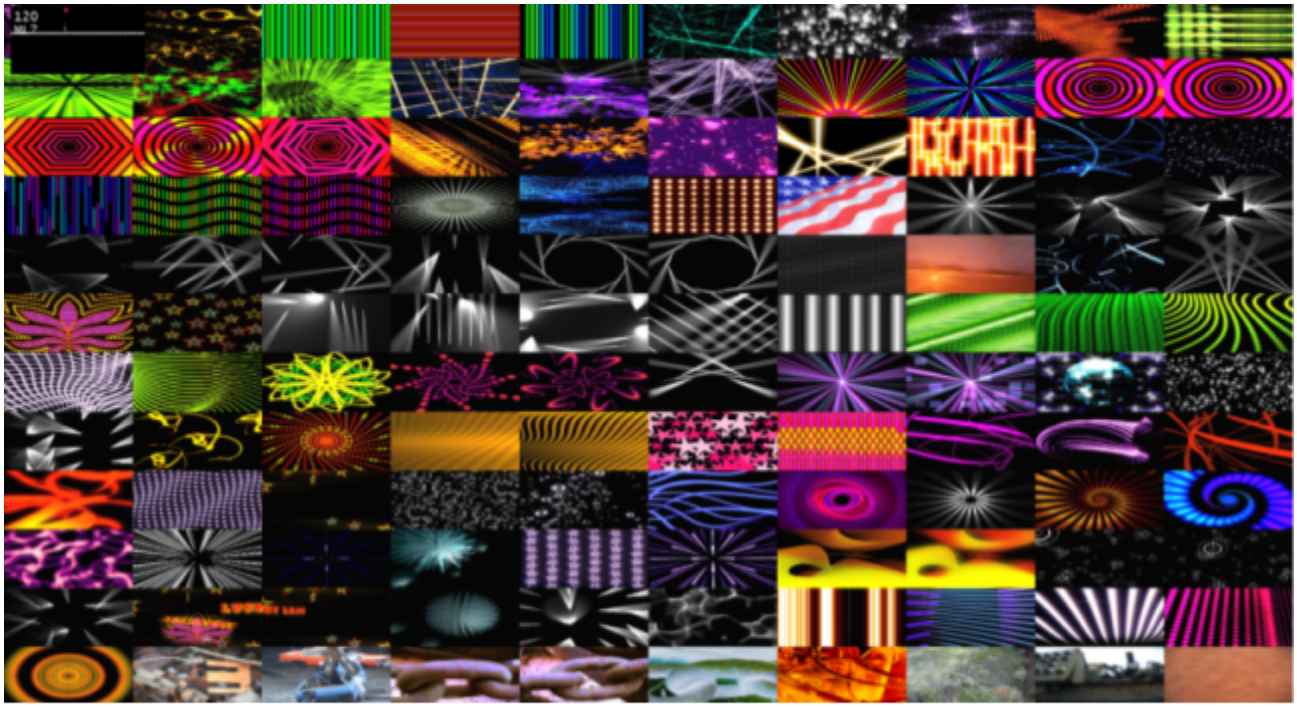
1. General information

1.1 The story

This plugin is the only one solution which managed to provide **lag-free high-frame rate 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 presented 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 a life-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 alpha channel transparency
— Reduced data throughput to graphics hardware



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

The 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, 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 @ 60 fps** videos, play multiple videos at once, with extremely fast frame-precise seeking
— Very high quality videos with the new **Hap R** codec
— **Transparent videos** with Hap Alpha, Hap Q Alpha and Hap R codecs.
— **Chunked Hap** support for even faster decoding
— 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 2019-2023
— Windows 8.1/10
— Linux 64-bit (glibc 2.35+, tested on Ubuntu 22.04)

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 black line watermark effect which is shown during the playback:

1.6 Getting example scenes to work

All the example scenes are located in the `Assets/DemolitionMedia/Examples` folder.

In order to get them working, download the sample videos archive via the link below and put the `SampleVideos` folder from it to `Assets/StreamingAssets`. Download link: https://mega.nz/folder/DAp1GThA#Iy-KDvLD-io6NOqW6WRKbQ

1.7 Native plugin upgrade instructions

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
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

1.8 Support

We offer support via e-mail: demolition.studios.rocks@gmail.com. If you have any problems just write to me and I will try to answer ASAP

You can also (optionally) post an issue on github.

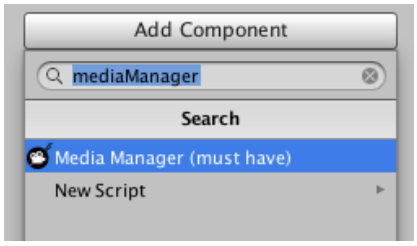
2. Usage

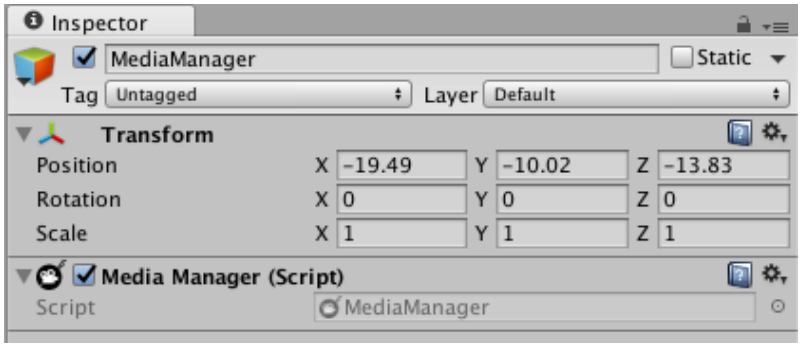
Watch this video for the step by step usage instructions!

2.1 Basic IMGUI usage

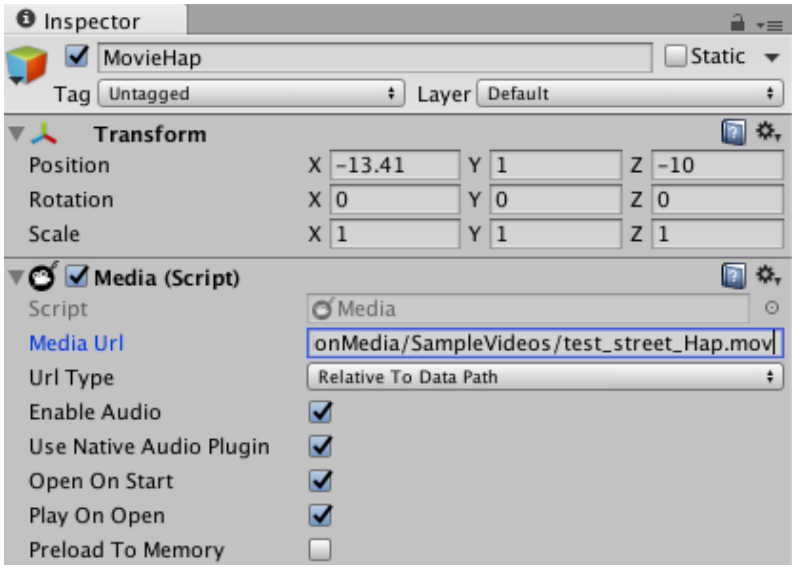
Follow these instructions to get a basic scene with IMGUI video player working:

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

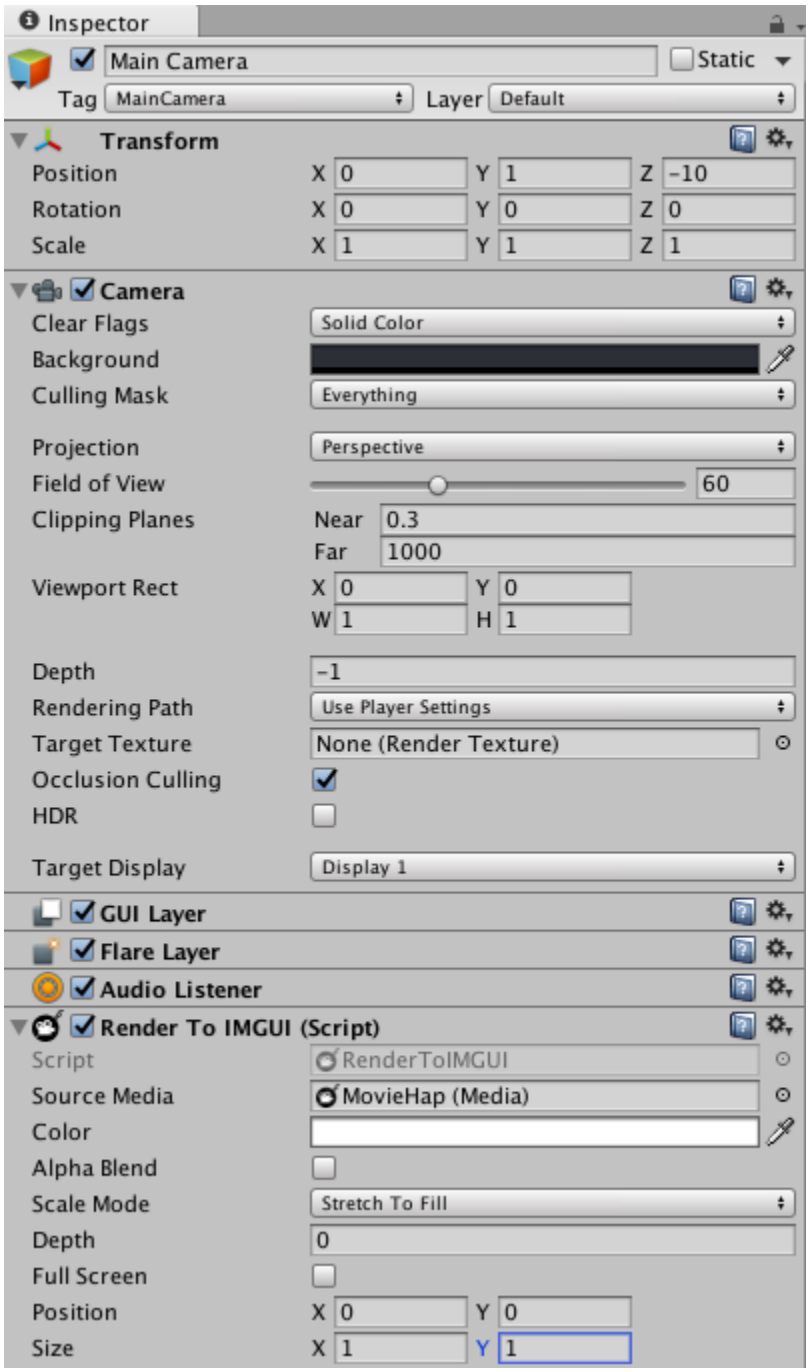




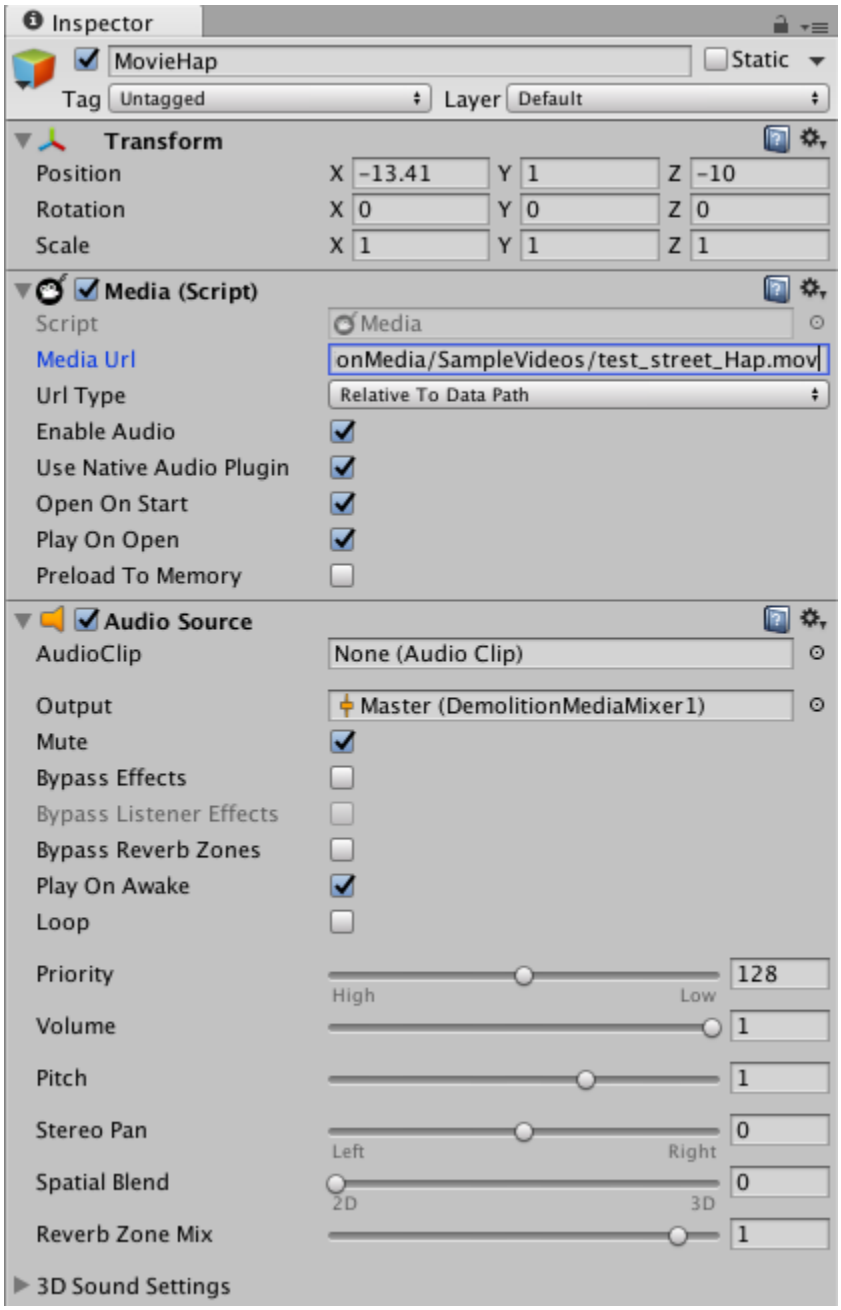
3. Again, add an empty object, attach `Media` script to it.
   Fill in `Media Url` (path to the video), along with the needed `Url Type`.



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



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



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.

3. Q: I cannot set playback speed to a value above 1 or 2.
A: Try disabling VSync in Editor or Player

4. Q: I need to play the video in **reverse**.
A: Please consider buying Pro Sync asset version, reverse playback is supported there

2.3 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 Setting up VR scenes

Below are the steps to get the plugin running in the VR environment (example for Oculus HMD's and Unity 2021):

1. Set up `OpenXR Plugin` in the Package Manager (from Unity Registry)
2. Convert camera to `XR Rig` by right clicking in the hierarchy `XR` > `Convert Main Camera to XR Rig`
   If there is an error during this step, delete the existing gameobject with camera (e.g. "CameraRig"), manually add a new Camera component, tag it as main camera and then repeat.
   Tagging can be done like this: click on the Camera in the Hierarchy so its properties are brought up in the inspector, then change the Tag property right underneath the name field, so it says "MainCamera"
3. Open the Project Settings and in `XR Plugin Management` click `OpenXR`
4. Fix errors by clicking on yellow triangle and follow the steps

There are two VR scenes included in the Examples folder: `05_Hap_HMD_360` and `06_Hap_HMD_360_Stereoscopic`.

2.5 Advanced usage

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

3. Preparing HAP files

3.1 Hap formats

There are several Hap codec formats:

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 (don't use in Unity 2022+)
5. **Hap R**. Highly improved image quality compared to others, alpha channel support
6. **Hap H**. HDR floating-point format

In the next sections multiple ways to produce Hap-encoded videos are shown.

Note: Hap files are usually large and require for playback very fast storage – Solid State Drive or (even better) NVMe.

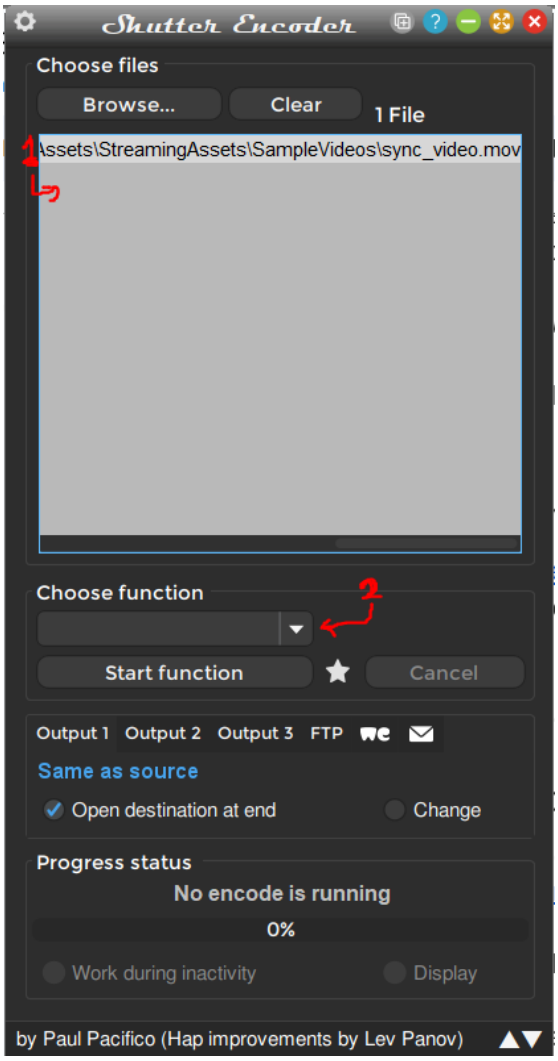
Note: To archive optimal playback smoothness, video file frame frequency must be multiple to projector / monitor scanning frequency, e.g. for a projector operating at 60Hz the perfect file FPS will be either of 30 or 60 frames per second, while a file with a frequency of 25 frames per second will ideally play at a scanning frequency either of 50Hz or 75Hz.

3.2 Encoding with Shutter Encoder

Note: for linux you can download the original version without advanced Hap support.

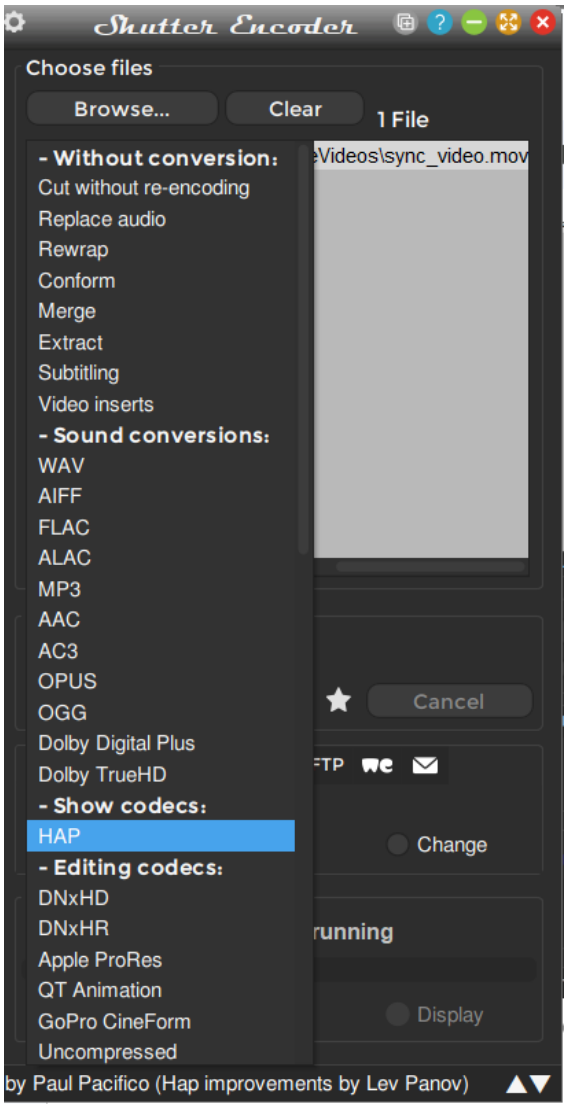
Go to https://github.com/DemolitionStudios/shutter-encoder/releases and download the latest release. It's a custom open source Shutter Encoder version with improvements made specially for Hap codec.
Please conside donating to support further encoder Hap improvements development.

1. Drag and drop or add using "Brouse…" the source videos

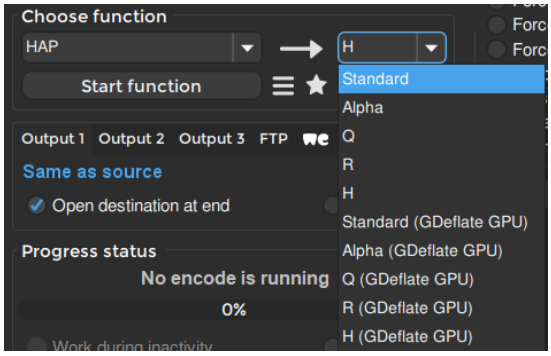


2. Click the function selector as shown above

3. In the pop up list click on HAP



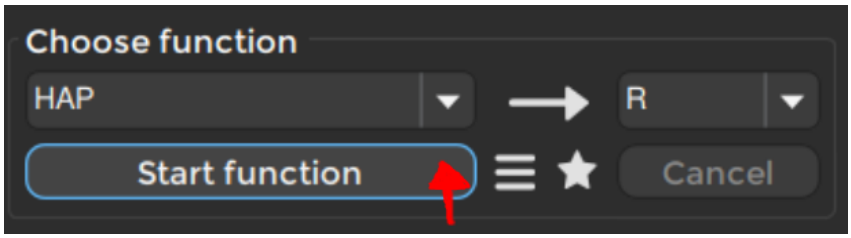
1. Select the desired Hap format on right:



We recommend R for quality (supports alpha channel), Standard for performance & smaller file size (no alpha channel), Alpha for performance (supports alpha channel), H for HDR data (e.g. .EXR image sequences)

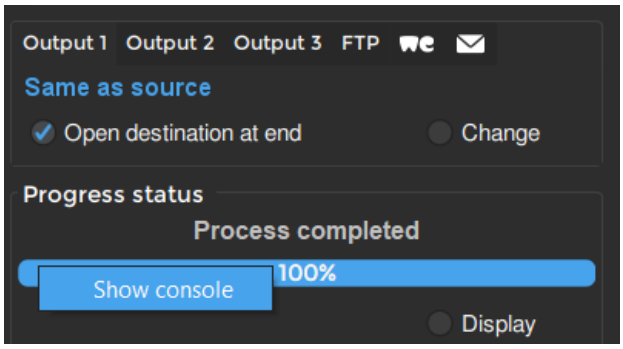
If you're going to use **GDeflate GPU Compute decoding (works only with [PRO SYNC asset]** with Unity 2022 with DirectX 12), choose the desired format from the 5 last list items. It has the advantage of both dramatically reduced CPU usage & 20-40% smaller file size compared to Snappy (usual compressor). Sometimes the file can be 90% smaller (10Mb vs 120Mb).

4. When ready, click the "Start function" button. The encoding process will start.



The output directory is the same as source file location by default, but you can change by selecting the "Change" option below the "Start function" button.

5. If encoding fails you can get the output log by doing right click on the progress bar:



Logs can be pasted to the github Issues page (the source video file should be uploaded somewhere also)

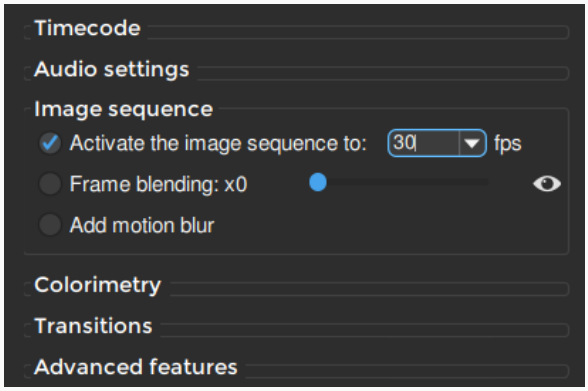
If you get: `Error initializing output stream 0:0 -- Error while opening encoder for output stream #0:0 - maybe incorrect parameters such as bit_rate, rate, width or height`

then first of all make sure your video dimensions are multiple of 4. If not, you can adjust cropping with ffmpeg:

```
ffmpeg -i vid.mp4 -filter:v "crop=WIDTH:HEIGHT:0:0" -c:a copy -c:v libx264 -crf 23  vid_crop.mp4
```

Replace WIDTH and HEIGHT according to your needs to multiples of 4.
`crf` parameter controls the quality. Less value gives better quality (23 is default)

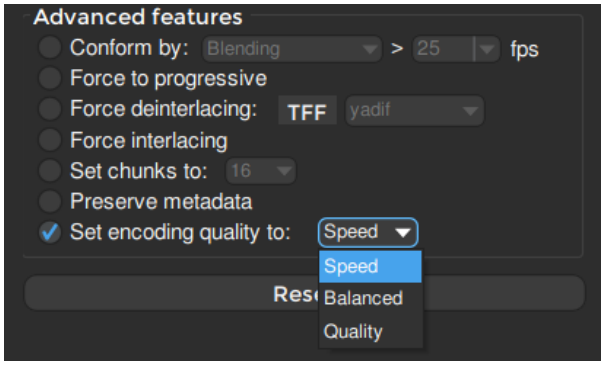
6. To encode an image sequence (particurally useful for encoding HDR .EXR sequences) you need to additionally specify the input sequence framerate in the "Image sequence" section:



7. Additional settings: expand the “Advanced features” on the right.

Hap R

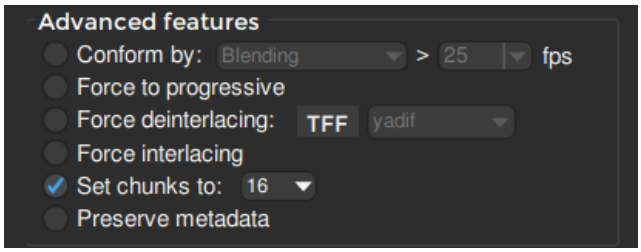
- Change the encoding quality / speed tradeoff



Hap H

- Change the encoding quality / speed tradeoff
- Change the normalization factor.
  Each input pixel value is being divided by that factor

All
- Change the chunk count



The default is 16 and should be optimal for 4/8 core processors.
For GDeflate formats, always leave that setting untouched.

For other functions, please refer to the official Shutter Encoder docs

3.3 Encoding with community-supported exporter

Download and install the exporter for Adobe CC applications from
https://github.com/disguise-one/hap-encoder-adobe-cc/releases

Further instructions for After Effects / Premiere Pro / Media Encoder.

It's the most "official" way to encode Hap videos, although it's not always the best.

3.4 Encoding using the new Jokyo Hap Encoder (paid)

Download trial version or buy full encoder version at https://jokyohapencoder.com/
Install it, and use it in After Effects / Media Encoder for exporting videos.

All Hap flavours are supported, better quality compared to the community exporter. You can also configure the quality / encoding speed ratio.

3.5 Encoding with FFmpeg (windows, linux)

The other way (and likely most convenient for those who just want to transcode existing videos to Hap) is to encode videos using FFmpeg.

Get Windows binaries: https://www.gyan.dev/ffmpeg/builds/
Recommended file: https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-full.7z (7-zip archiver)
Linux: install using your distro instructions (e.g. *sudo apt-get install ffmpeg*)

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

- To encode with **Hap**
  ffmpeg -i movie.mov -vcodec hap -format hap -chunks 8 movie_hap.mov
- To encode with **Hap Q**
  ffmpeg -i movie.mov -vcodec hap -format hap_q -chunks 8 movie_hapq.mov
- To encode with **Hap Alpha**
  ffmpeg -i movie.mov -vcodec hap -format hap_alpha -chunks 8 movie_hapalpha.mov
- **Hap R** isn't supported by ffmpeg to the date this document was written (update: supported in the version that is shipped with the custom Shutter Encoder)
- **Important:** when encoding high-resolution videos it's recommended to use the so-called chunked encoding mode. We use 8 chunks in the examples above - which means that Unity can decode the movie using 8 threads simultaneously, giving a major performance gain. Chunks number can be in the 1-64 range (default is 1) that does not exceed the number of CPU cores in the playback system.
- (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 a 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 – probably only NVMe will work!
  ffmpeg -i movie.mov -vcodec hap -format hap -compressor none movie_hap.mov

  I've managed to play 10k@50fps Hap on my average-level laptop with this approach. You can use the compressor flag with other Hap formats as well.

- Sometimes you can get an error if your video size isn't multiple of 4, e.g.
  `Video size 2274x1280 is not multiple of 4`
  To fix that, you can crop the video with the following command:
  ffmpeg -i vid.mp4 -filter:v "crop=WIDTH:HEIGHT:0:0" -c:a copy -c:v libx264 -crf 23 vid_crop.mp4
  Replace WIDTH and HEIGHT according to your needs to multiples of 4.
  `crf` parameter controls the quality. Less value gives better quality (23 is default)

- Convert vertical video to horizontal with blurred version in the background:
  *ffmpeg -i in.mp4 -filter_complex [0:v]scale=ih*16/9:-1,boxblur=luma_radius=min(h\,w)/20:luma_power=1:chroma_radius=min(cw\,ch)/20:chroma_power=1[bg];[bg][0:v]overlay=(W-w)/2:(H-h)/2,crop=h=iw*9/16 out.mp4*

3.6 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. 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 allows us to use GPU acceleration and fast frame-precise seeking. Please see section 3 to find out how to encode Hap videos!

In case of problems opening a video, make sure you're opening a valid Hap file:
cross-check with a reference player, JokyoHapPlayer. You may need to re-encode the video

Q1: I can see some visual artifacts or color banding in the encoded videos
A: If you're using a simple Hap format, it can be the case. Even Hap Q can have some problematic areas in the video. The best quality can be achieved with the new Hap R format (use Jokyo Hap Encoder for encoding) – see section 3 for encoding instructions.
If you really need to use simple Hap format (e.g. for performance reasons), try adding noise or texture to the video until banding breaks down. See
https://github.com/Vidvox/hap-qt-codec/issues/35

Q2: 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.

Q3: 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`

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 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 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`.

Q6: 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 to either preload to memory or buy a faster SSD or use a 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 the 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:

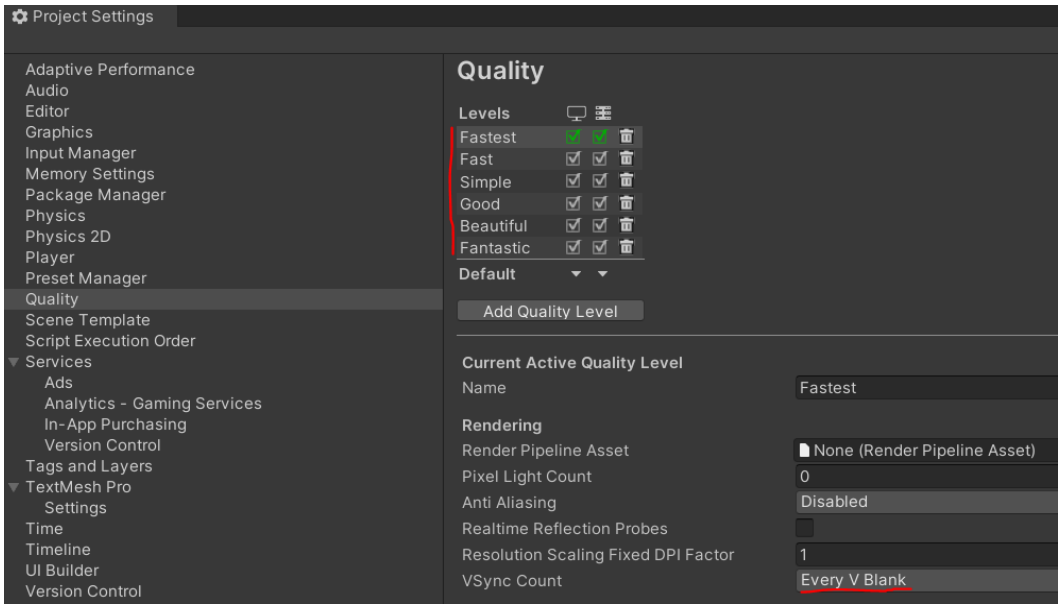
- switch from Hap Q to Hap
- lower the resolution / frame rate
- use NVMe SSD which give 3Gb/s+ speed

If the bottleneck is CPU:

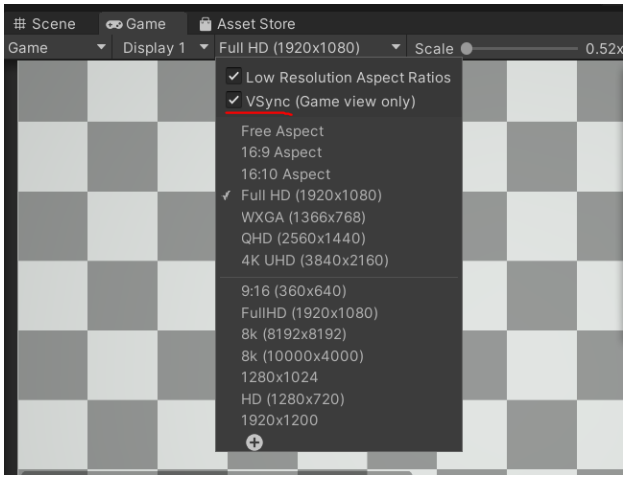
- disable 2nd stage compression (snappy)
  example of FFmpeg command line for encode
  ffmpeg.exe -i in.mov -vcodec hap -format hap -compressor none out.mov
  The data-rate will be constant and also higher compared to snappy compressor, but at least twice less CPU work will be required before the frame 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 / frame rate

Q7: I want to play 8192x8192 8k video, why isn’t it playing smoothly?
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 a 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 a sanity check.

Q8: The CPU usage is high, even though the video is paused
A: Make sure `VSync` setting is set to `Every V Blank` in the Unity `Project Settings`:



and also in the Game settings:



Q9: I get “no usable version of libssl was found” error on Linux (Ubuntu)
A: Execute the following in the terminal:

```
wget
http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.0g-2ubuntu4_amd64.deb
sudo dpkg -i libssl1.1_1.1.0g-2ubuntu4_amd64.deb
```

Q10: On Linux Vulkan + Linear color space + IMGUI rendering is too bright
A: That's a general Unity problem. Use other method then IMGUI

Q11: On Linux I cannot import an AudioClip
A: That's a general Unity problem. I haven't found a way to import mp3/wav/ogg from Editor. But you can create an audio clip from mp3 programmatically, see LFOClockNetworkEnetClient.cs for Pro Sync version for an example (inside `foreach (var path in Playlist.AudioFiles)` loop), or

```
using UnityEngine;

using UnityEngine.Networking;

var audioClipUrl =
"file:///home/user/project/Assets/audio.mp3";

UnityWebRequest req =
UnityWebRequestMultimedia.GetAudioClip(audioClipUrl,
AudioType.MPEG);

req.SendWebRequest();

while (!req.isDone)

    Thread.Sleep(5);

var audioClip = DownloadHandlerAudioClip.GetContent(req);
```

5. Changelog

3.2.0
— Added Linux support
— No more memory leaks at all! Now it can be used in 24/7 installations
— Fixed problems in scripts in some Editor versions
— Fixed Hap Alpha only and Hap Q Alpha playback for Unity textures
— Removed problematic DisableAudio method (use Audio Mixer capabilities instead)
— FFmpeg modernization (remove flush_pkt)
— Pro Sync (windows): can play multiple videos with GDeflate GPU decoding
— Pro Sync (windows): can use GDeflate compressed videos in playlist

3.1.0 (released 21.10.2023)
— More robust C# / native plugin interop
— Improve all the Render components, so user can specify custom scale/offset/flipX/flipY
— Fixed an error when building projects
— Now possible to encode Hap H HDR data (e.g. .EXR image sequences) using our custom Shutter Encoder version
— Updated docs

3.0.0 (released 7.05.2023)
— Update to FFmpeg 6.0
— New encoder app available for download (link on plugin GitHub page)
— Fix flickering in some cases

1.0.1 (released 15.09.2020)
— Incorrect seeking fixed
— Added configurable GPU/RAM memory limits

1.0.0 (released 14.07.2020)
— Reworked playback loop - even more smooth playback
— Improved stability and performance - can play even more videos now
— Unlocked decode threads number: it scales with the the CPU concurrency capabilities
— Frees Unity render and main threads: texture updating is done completely in background threads
— Better playback speed control
— Multichannel audio support
— Only 10Mb required for the plugin DLLs

0.9.6 (released 26.09.2019)
— Unity 2018/2019 support

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

0.9.3 (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

0.9.1 (released 13.02.2017)
— Initial version