Demolition Media Hap for Unity

version 0.9.3

1. General information

1.1 The story

This plugin proved to be the only one solution which managed to preserve **lag-free high-framerate video playback** in The Complex mixed reality playscape with Unity running on **8 FullHD projectors (8k video resolution)**. We tried other solutions, but they just didn't work. Finally, we made this plugin.

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 resolutions with fairly low CPU usage, since it provides 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.

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 technical description of how to prepare the Hap files in section 3.

1.3 Plugin features

— **Hardware accelerated** **HAP** video playback **without any external codecs** installation needed.
— Low CPU/memory usage. Frames are **decompressed on the GPU**.
— Play **8k @ 60 fps** 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 the graphics APIs (**DX9, DX11, OpenGL, Metal**) on desktop platforms.
— **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 5.x (currently being tested on 5.4, but should work on older versions — contact me if it doesn't)
— 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 possible on request. Contact me if you need it.

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 Support

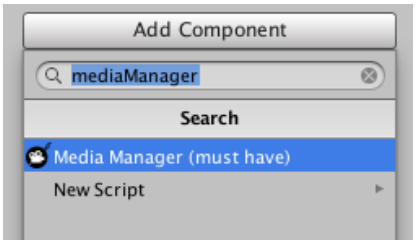
We offer support using e-mail address demolition.studios.rocks@gmail.com. Really, if you have any problems just write me and I will try to answer ASAP. If you've bought the plugin on the asset store (thank you!) don't hesitate to include your invoice number in the letter text. You can ask anything about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

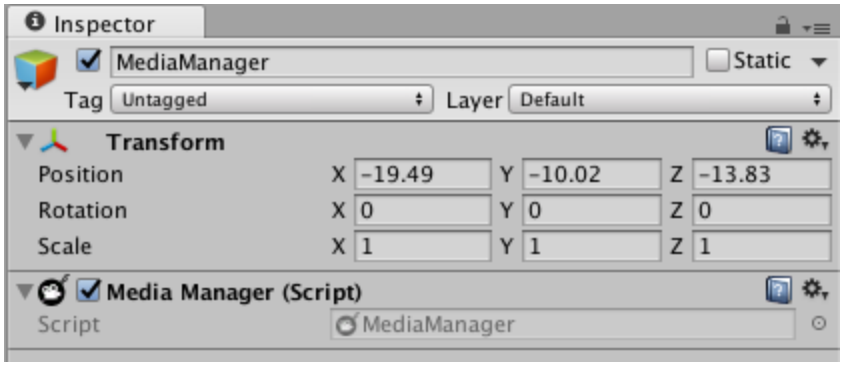
2. Usage

2.1 Basic IMGUI usage

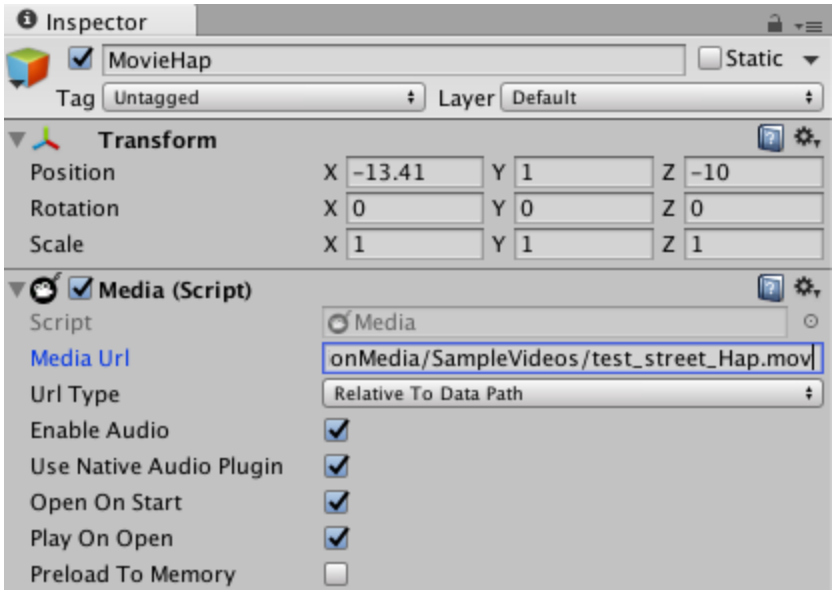
Follow these instructions to get a basic scene with IMGUI video player working:

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

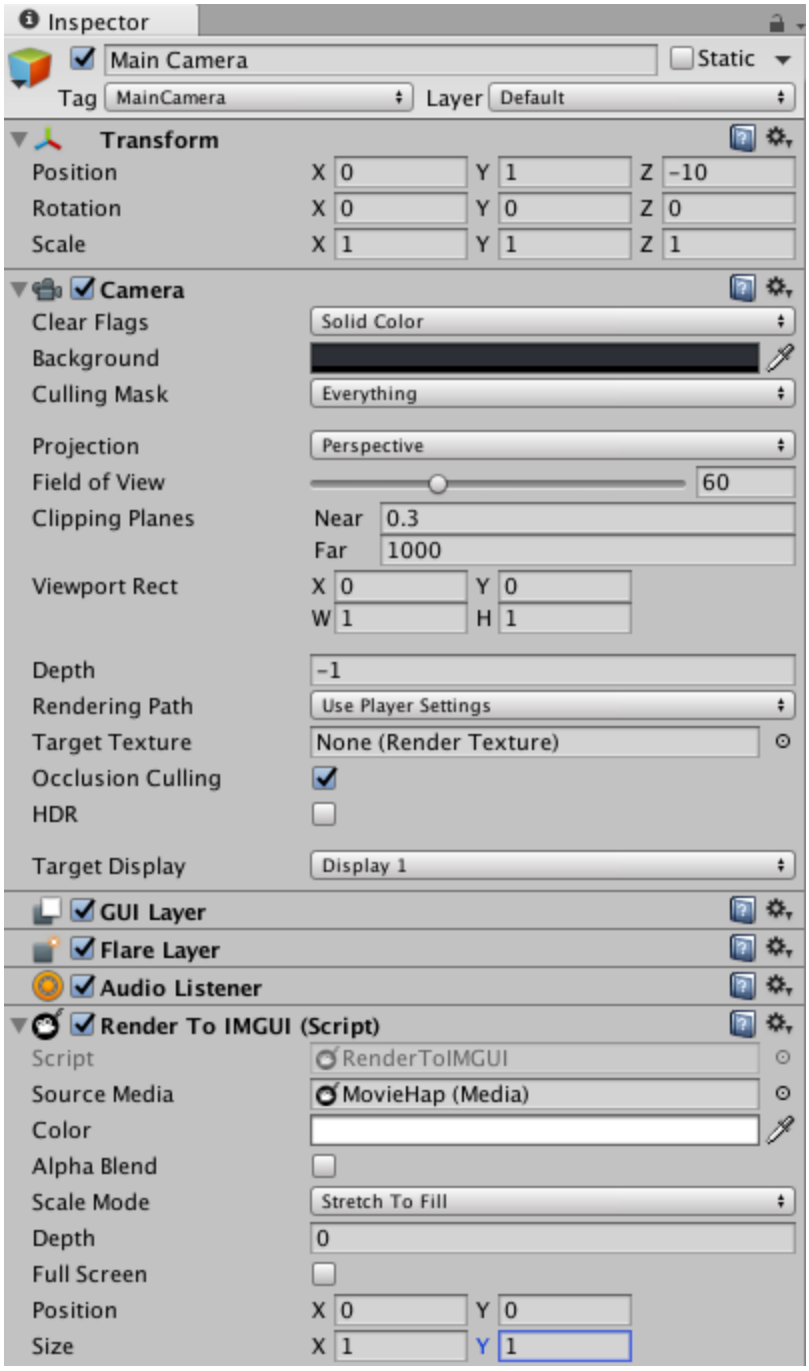




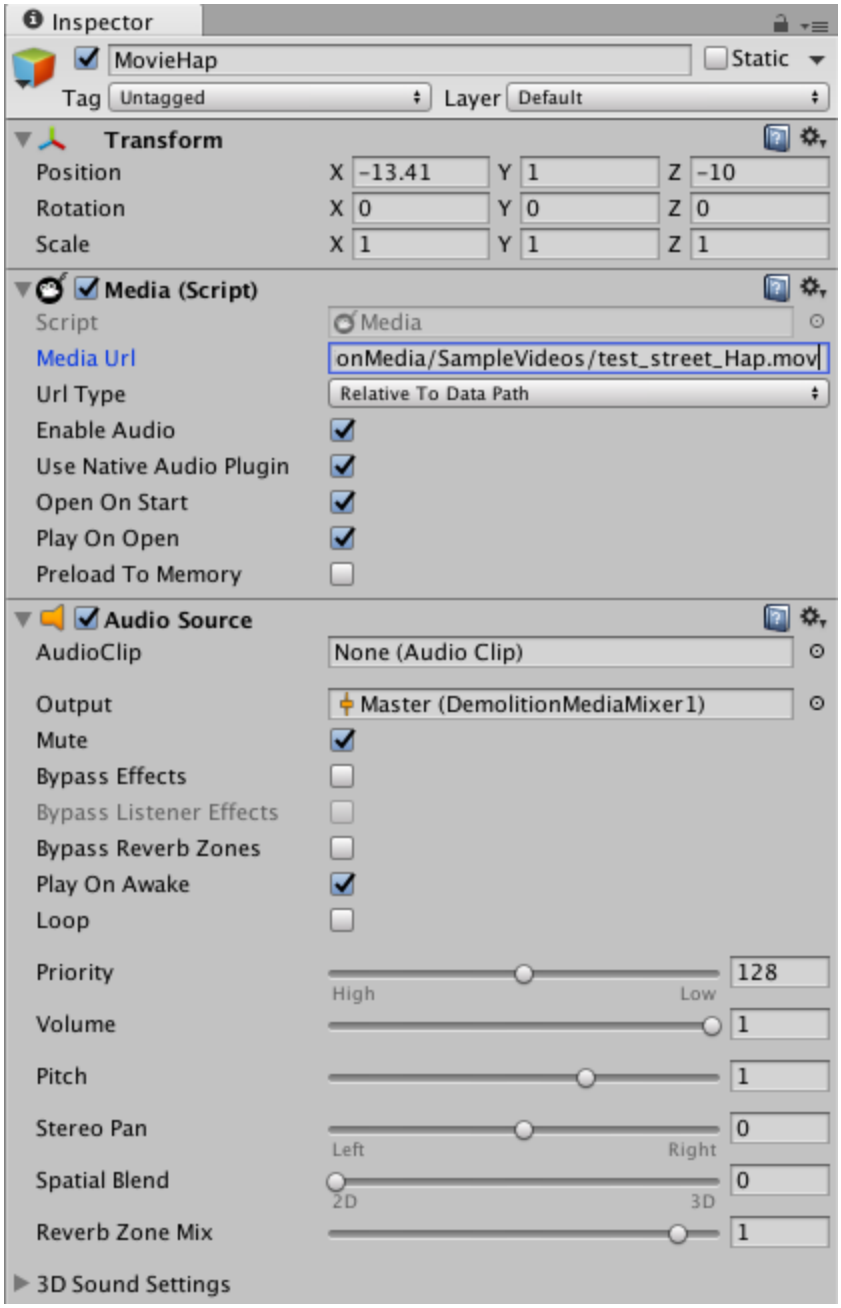
3. Again, add an empty object, attach “Media” script to it.
   Fill in Media Url (path to the video), along with the needed Url Type.



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 same object as in step 3. Output should be: “Master (DemolitionMediaMixer1)”.



This is the advised way of getting the high-performance audio output.

6. Optionally attach the “Keyboard controls” script to get the basic keyboard controls working.

2.2 Playback speed

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

Please consider the following if using this feature:
1. **It's recommended to disable the audio** if the playback speed differs from 1.0 (the default one).
Otherwise the audio/video won't be in sync (audio stream 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 computer capabilities and video bitrate. To determine the situation where the playback speed is too high, 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 Advanced usage

For more advanced usage scenarios checkout out the example scenes provided with the asset.

Also watch this video for the step by step usage instructions:
https://www.youtube.com/watch?v=SRSYF4atzTs

2.4 Older OS X versions

Since the Metal graphics API is only shipped with OS X 10.11 SDK, the plugin version provided on the Asset Store may not work on older OS X versions. If you need to use the plugin on these OS X versions, please request the plugin build with older libraries via the support e-mail.

2.5 Other notices

Since Unity Asset submission rules are kinda strict, to build the example scenes, you're responsible for placing the sample videos to the StreamingAssets folder and making the correct paths to them on yourself!

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.

3.2 Encoding with QuickTime codec

There are several options how you can produce hap encoded video files. First one is to download and install the “official” QuickTime 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).

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_q -chunks 8 movie_hapq_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.

3.4 More encoding ways

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

1. HapInAVFoundation (batch converter, Mac only)
   It can produce Hap Q Alpha!
2. TouchDesigner (you can even bake some real-time footage with it! Read more here)
3. More on Hap Alpha encoding:
   http://vdmx.vidvox.net/tutorials/working-with-hap-alpha-encoded-movies-in-vdmx

4. 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