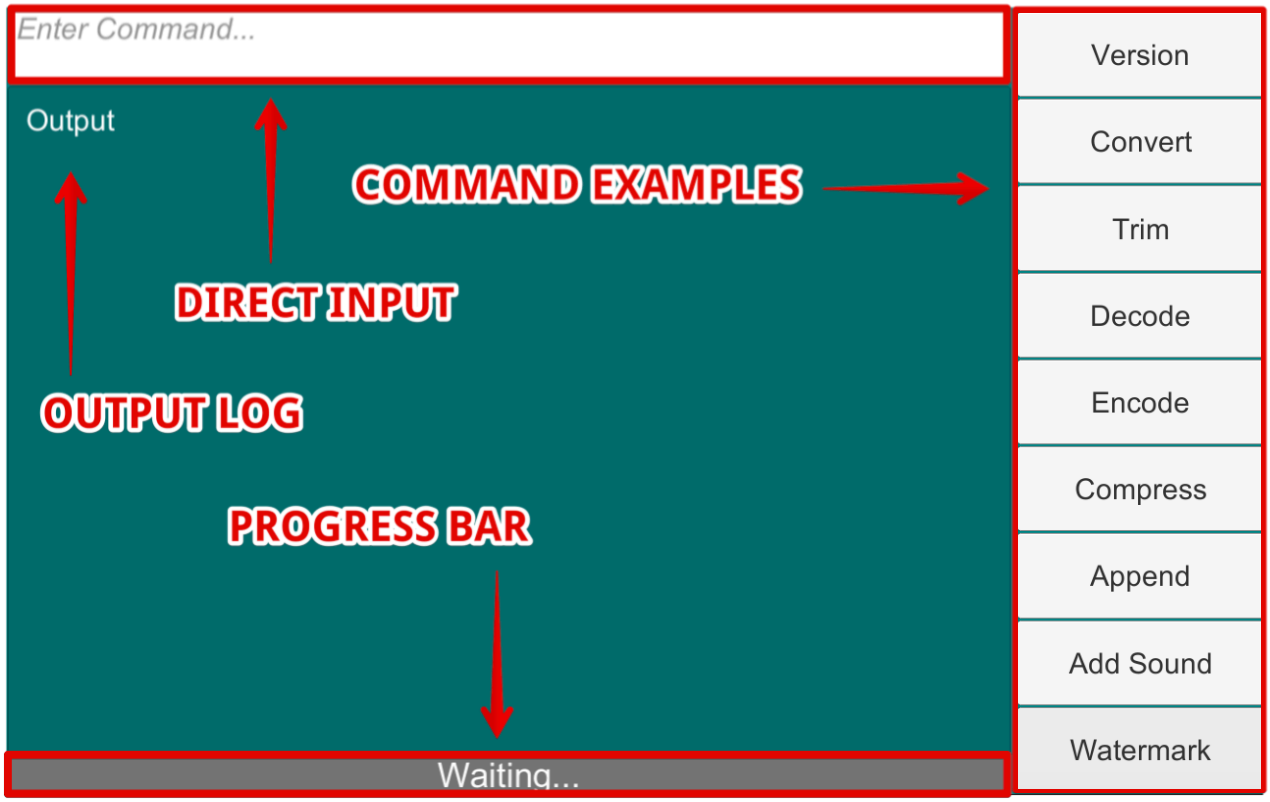
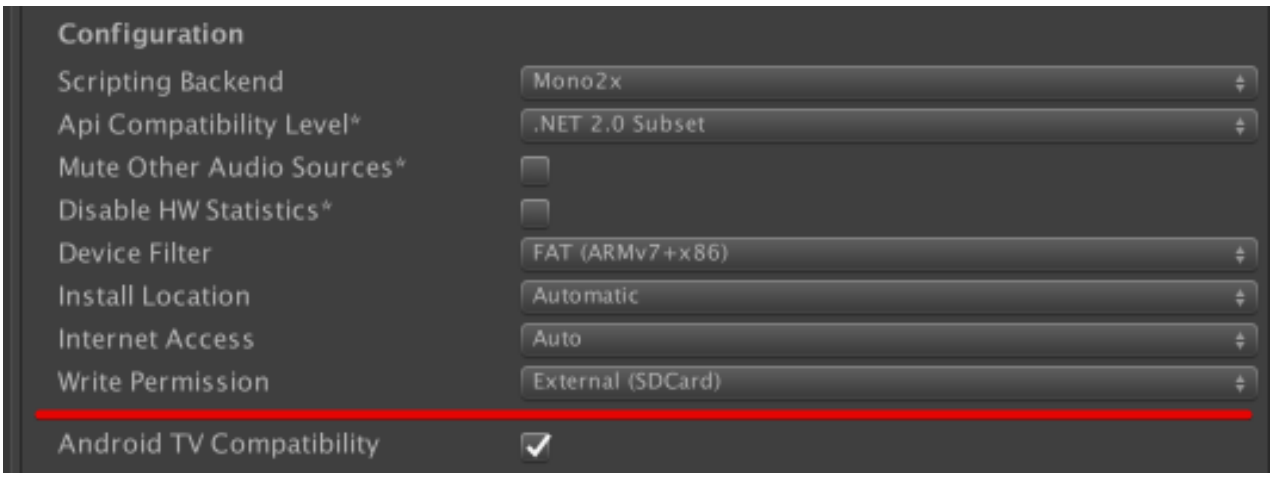
FFMPEG UNITY BIND 3.2

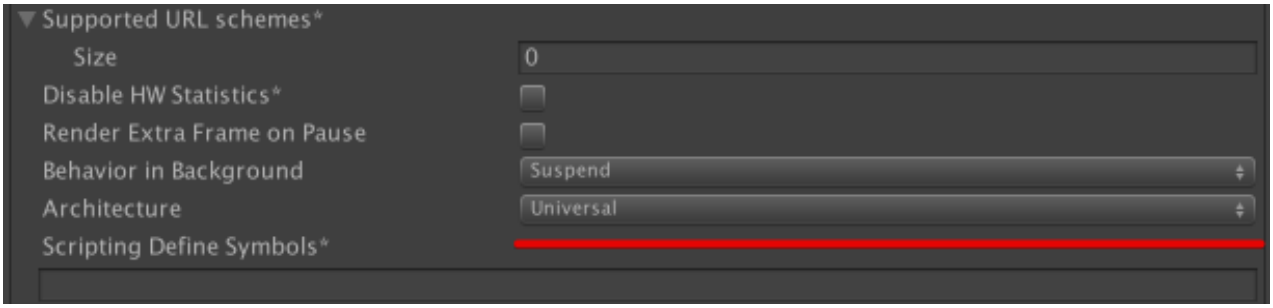
1. Take a look how it is organized in FFmpegDemo scene.



2. Build on device to test demo scene.
   Screen orientation is Landscape.
   On Android make sure that you've checked Write Permission -> External (SDCard).



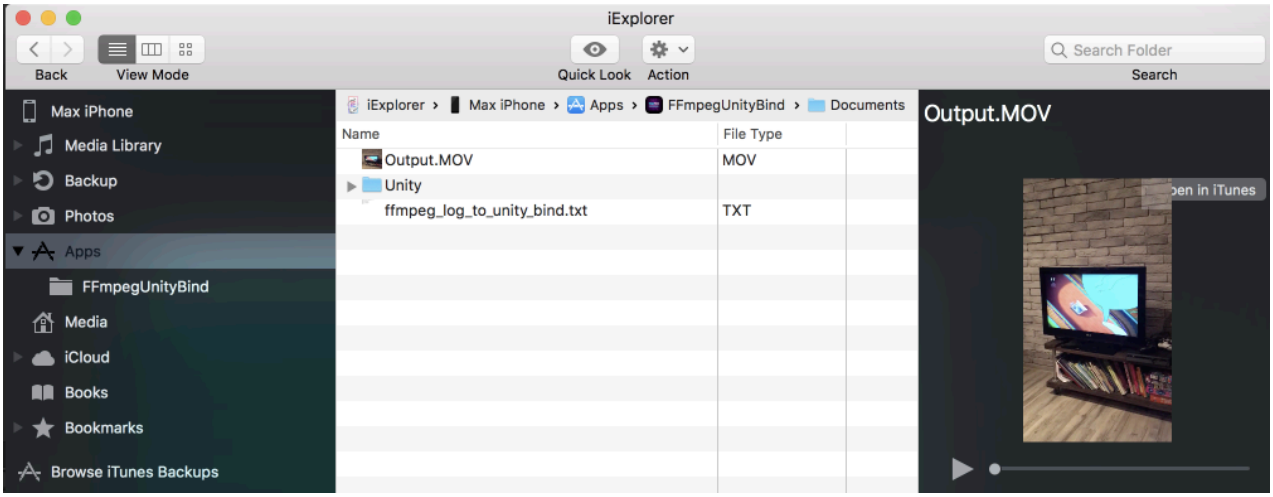
On IOS it is important to check if Universal build architecture selected. Rest of the job should do IOSPostBuild.cs

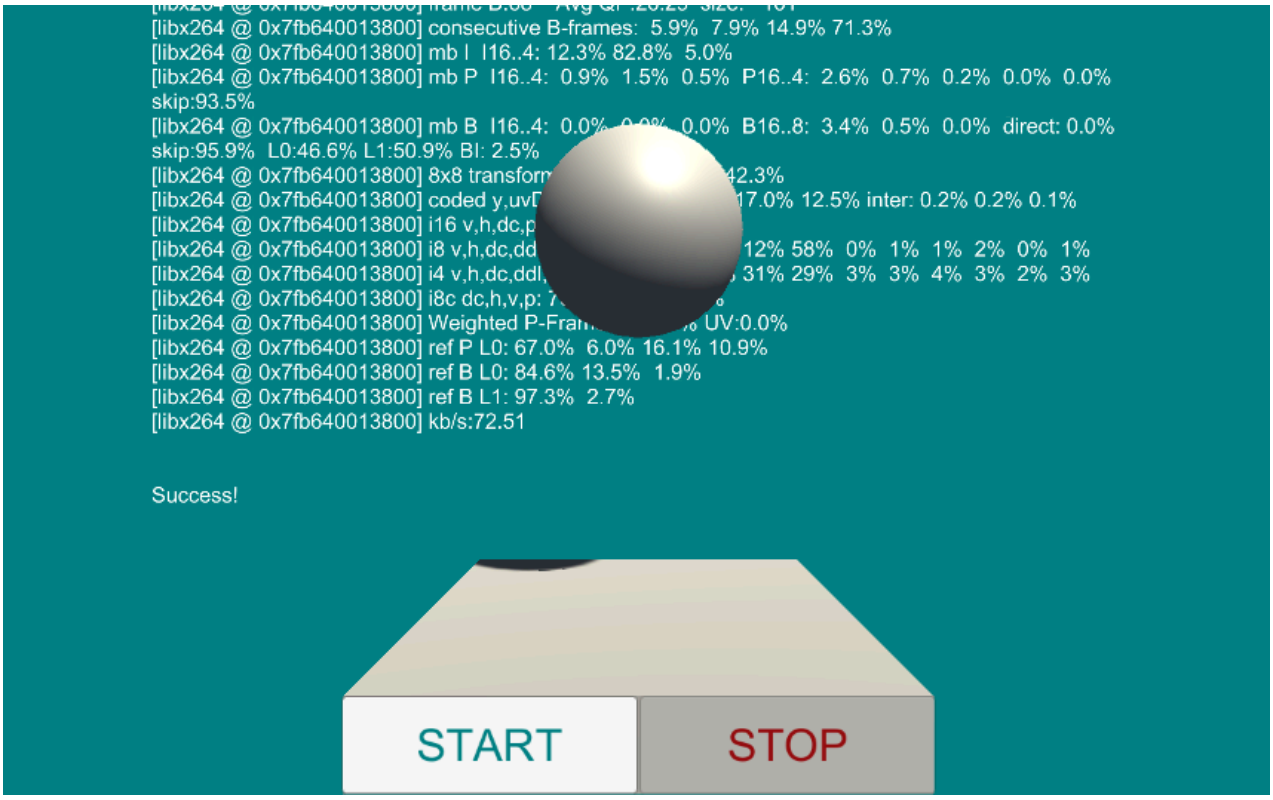


On Mac OS please make sure you have all rights to execute ffmpeg binary file in Assets/FFmpeg/Standalone/Mac folder. Usually script tries to grant permissions automatic. You can also manually navigate to this directory in terminal and execute: chmod 700 ffmpeg

3. Fill example fields in specified format end inspect output results in Android file manager. Play with direct input console to do the same by your unique commands.

On IOS there is a Video Picker helper which simplifies things. You can inspect results via external software like iExplorer. Here is an Educational (FREE) version of it for Mac.





4. Build on device and test FFmpegREC scene. It is absolutely cross-platform. Every frame a screenshot is taken and in the end all of them encoded to the video.
On slow devices you can reduce capturing FPS (which improves performance of screenshots making) or / and reduce resolution (boost all).

Set up own scene

1. Put FFmpeg.prefab to your scene. That's it.
2. Usage: Make copy of FFmpegDemo.cs rename it and change according to needs of your application.

Understanding code

1. FFmpegWrapper implements 2 simple methods needed for all operations (initialization and execute). Initialization is performed in unity Start() method and there is nothing special about it (NOTE: you should call other operation after initialization). Execute(string[] cmd) is a console interface for all FFmpeg operations. You can work with that directly or using Helpers additionally included into this package.

2. Helpers:
   - FFmpegCommands: Encapsulates commands construction to have simple call from application logic (Convert, Trim etc). Constructed commands are sent to FFmpegWrapper. You can send commands directly to FFmpegWrapper.Execute(string[] cmd).
   - FFmpegParsers: Gets FFmpeg events from response string and calls them. Make sure that you've assigned handler for events receiving:

```
using UnityEngine;

namespace FFmpeg.Demo
{
    public class FFmpegDemo : MonoBehaviour, IFFmpegHandler
    {
        public ProgressView progressView;
        public ConvertView convertView;
        public TrimView trimView;
        public DecodeView decodeView;
        public EncodeView encodeView;
        public CompressView compressView;
        public AppendView appendView;
        public AddSoundView addSoundView;
        public WatermarkView watermarkView;

        FFmpegHandler defaultHandler = new FFmpegHandler();

        //------------------------------

        void Awake()
        {
            FFmpegParser.Handler = this;
        }

        //------------------------------

```

- IFFmpegHandler: Implement it to know when video operations was finished and when it is processing (see FFmpegDemo.cs).
- FFmpegConfigs: Simple data structures with commands construction parameters. It is used by FFmpegCommands.

On your own

1. This version of FFmpeg library is used for binding on Android: https://writingminds.github.io/ffmpeg-android/
2. This is specific Android assembly: https://github.com/WritingMinds/ffmpeg-android-java
3. On IOS FFmpeg was built like this: https://github.com/kewlbear/FFmpeg-iOS-build-script
4. Learn FFmpeg cross-platform api and have all power of it functionality: https://ffmpeg.org/documentation.html

Support

1. FFmpeg Unity Bind offers just a platforms binding.
2. F.A.Q. Support: biz@giganeo.com