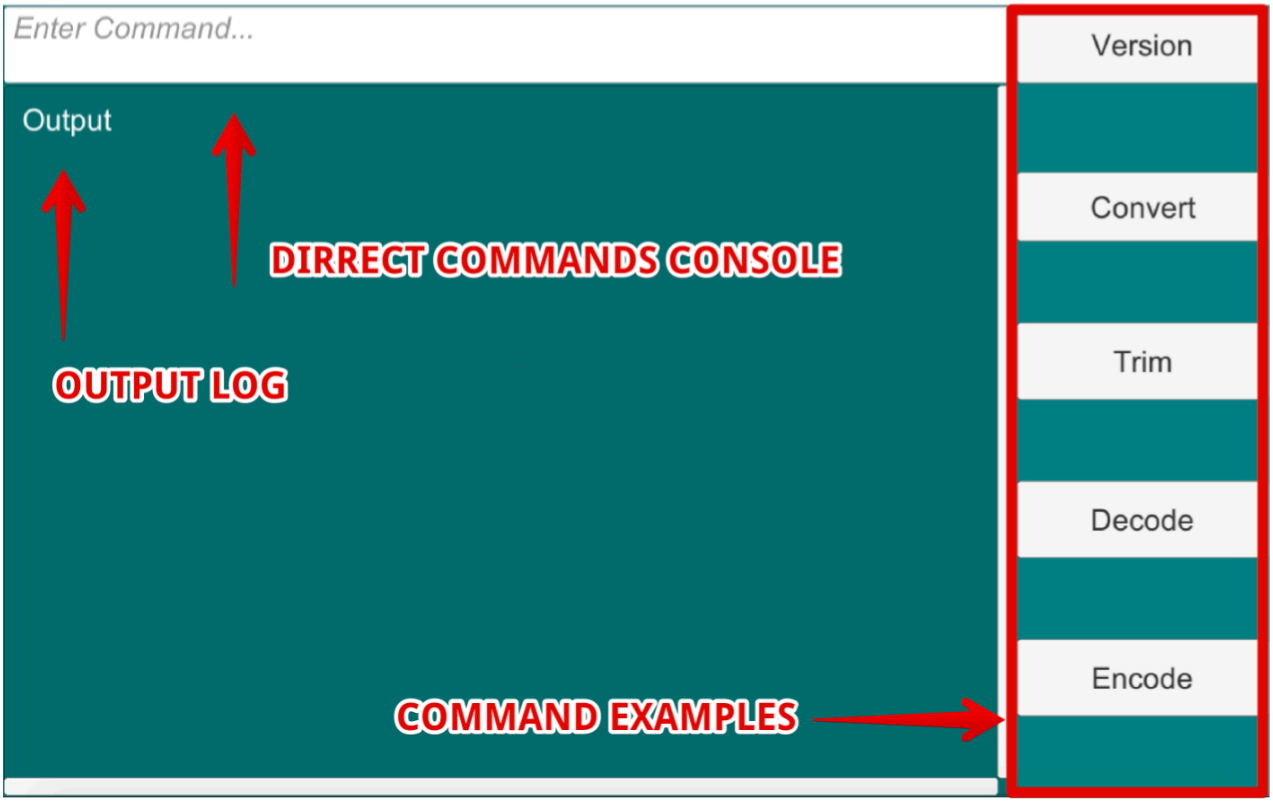
1. Take a look how it is organized in FFmpegDemo scene.



2. Build on device to test demo scene. Make sure that you've checked Write Permission -> External (SDCard).

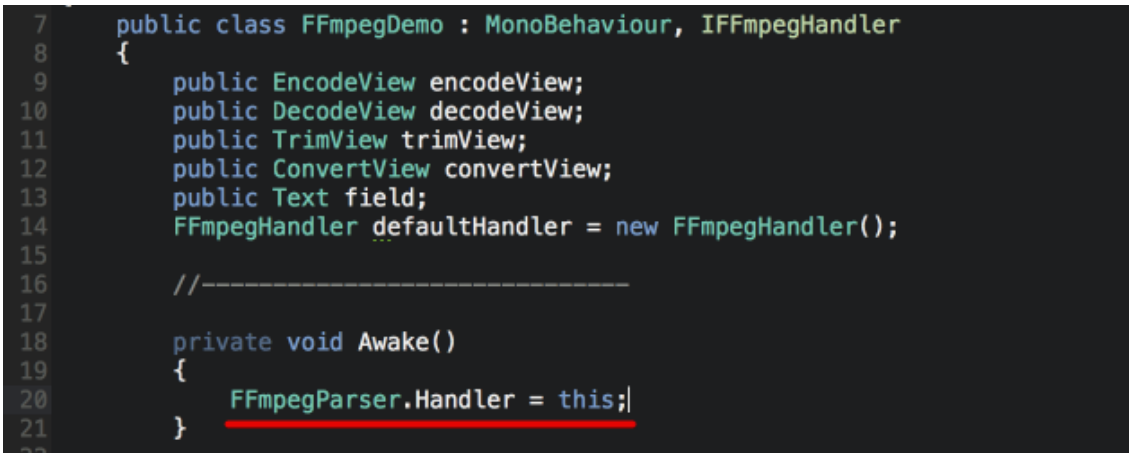
3. Fill example fields in specified format. Play with direct input console to do the same on your own.

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 this events:



- 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: https://writingminds.github.io/ffmpeg-android/
2. This specific assembly: https://github.com/WritingMinds/ffmpeg-android-java
3. Learn FFmpeg cross-platform api and have all power of it functionality: https://ffmpeg.org/documentation.html

Support

1. FFmpeg Unity Binding offers just a platforms binding.
2. To get help with your specific task contact us via mail: max-bot@outlook.com
   Skype: superninja28