SiaPlayerDemo – Release Document

# Introduction

To source code for this project can be found at:

<https://github.com/sergiorf/SiaPlayerDemo.git>

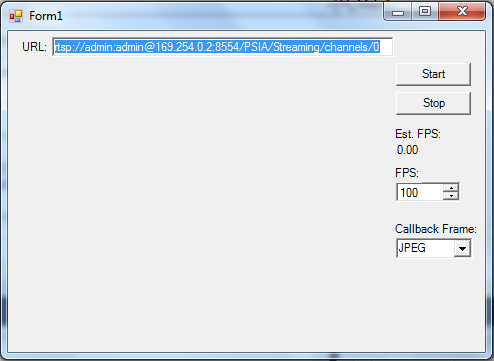
This project consists on the following folders:

* ***3rdParty***: Contains the source code of the 3rd party dependencies:
  + **FFmpeg**: Used for H.264 stream decoding and JPEG re-encoding.
  + **SDL**: Used for threads and mutex services.
* ***fmpeg*-dev**: Contains the FFmpeg headers and libraries (both x86 and x64) needed to compile and link our project. Note that the x86 libraries have been compiled from the sources[[1]](#footnote-1) in 3rdParty/FFmpeg. Unfortunately I’ve been unable to compile the x64 version, so I downloaded them pre-built from this site: ffmpeg.zeranoe.com/builds.
* ***SDL*-1.2.15:** Contains the SDL headers and libraries (both x86 and x64) needed to compile and link our project. Both x86 and x64 libraries have been built from sources, using the VS project file found in 3rdParty/SDL-1.2.15.
* ***MinGW-*RT:** The FFmpeg x86 libraries (found in fmpeg-dev folder), have been compiled from sources (found in 3rdParty/FFmpeg) using MinGW’s GCC compiler. Thus, in order to execute our project we need the MinGW runtime libraries available in this folder.
* ***SiaPlayer:*** This is essentially a dll wrapper around FFmpeg, which offers the ability to do H.264 stream decoding and transcoding, and sends the transcoded frames to a user registered callback at a user defined FPS rate.
* ***SiaViewer:*** This is a small C# GUI app example, demonstrating how to use the SiaPlayer API.

# How to test SiaViewer?

Open the solution file SiaPlayerDemo.sln with Visual Studio Professional 2013 (other VS versions haven’t been tested). Choose the right configuration for the solution: x64/x86 and Debug/Release and build. Note that a Post-build event will copy the necessary runtime dependencies to the folder where the .exe resides. Finally set SiaViewer as a StartUp Project and run.

A Form will show up as follows:



**URL**: Enter here the stream address from the Video Server. Don’t forget to put the credentials.

**Start/Stop Buttons**: Start/Stop the decoding and rendering process. If the URL information is right, after pressing on the Start button, a video stream should be visible in the central area of the form.

**Est. FPS**: This is the video frame’s rate received from the source stream. The SiaPlayer will use chunks of 50 received frames to (re)calculate this estimate in real-time. This value is for informative purposes only.

**FPS**: The user can specify here the rate of video frames that she wishes to receive on the callback. A number higher than the value from Est. FPS (original’s frame rate) will have no consequence whatsoever on the decoding and frame rendering. A number that is less than Est. FPS will result in the SiaPlayer dropping some frames, thus sending fewer of them to the client via the callback.

**Callback Frame**: The user can specify here the encoding of the frames received through the callback. Two formats are supported: RGB (24bpp) and JPEG.

# Annex 1 – How to compile FFmpeg from sources?

These are the steps to compile FFmpeg from sources:

1. FFmpeg can be compiled on Windows using the MinGW’s GCC compiler. Thus we need to download the MinGW Installation Manager and install all packages in Basic Setup. Make sure the GCC version is >= 3.4.5. By default MinGW will be installed at c:\MinGW. Add c:\MinGW\bin to the system’s PATH.
2. Download yasm-1.2.0-win32.exe or yasm-1.2.0-win64.exe into c:\yasm.
3. Open up Visual Studio Developer Command Prompt and navigate to c:\MinGW\msys\1.0. Run msys.bat from there.
4. Change your current home directory to ffmpeg’s dir, i.e.: “cd ${ProjectDir}\3rdParty\FFmpeg”.
5. Rename MinGW link.exe to avoid conflict with MS link.exe with “mv /bin/link.exe /bin/mingw\_link.exe”. Now running link.exe should point at MS link. Double check with the command “which link” it should respond something like "/c/Program Files (x86)/Microsoft Visual Studio 11.0/VC/Bin/link.exe".
6. Extend default path with the command "export PATH=$PATH:$(ProjectRoot)\3rdParty\FFmpeg".
7. Type the commands below. (a) and (b) take some time to complete (about 5-10 minutes)
   1. "./configure --prefix=ffmpeg/ --enable-shared --extra-cflags=-I/c/MinGW/include --yasmexe='C:/yasm/yasm-1.2.0-win32.exe' --enable-gpl --cpu=i686" ... or ... "./configure --prefix=ffmpeg/ --enable-shared --extra-cflags=-I/c/MinGW/include --yasmexe='C:/yasm/yasm-1.2.0-win64.exe' --enable-gpl --cpu=x86\_64" for 64bits
   2. "make"
   3. "make install"
8. Now under "$(ProjectRoot)\3rdParty\FFmpeg \lib" you should see some \*.def and \*.a files. It’s time to convert \*.a to standard MS \*.lib files. Change directory using the command "cd $(ProjectRoot)\3rdParty\FFmpeg \lib", and do:
   1. lib /machine:i386 /def:avcodec-55.def /out:avcodec.lib
   2. lib /machine:i386 /def:avformat-55.def /out:avformat.lib
   3. lib /machine:i386 /def:avutil-52.def /out:avutil.lib
   4. lib /machine:i386 /def:swscale-2.def /out:swscale.lib

I produced the binaries successfully for a x86 configuration, unfortunately I couldn’t do the same with the x64 built, due to some issue during make.

1. See Annex 1, for information on how to compile FFmpeg from sources [↑](#footnote-ref-1)