GETTING STARTED

With FMOD Ex Programmer's API for Android

UPDATED : MAY 2012

LEGAL NOTICE

CONTENTS

Contents

Introduction

Welcome to the FMOD Ex Programmer’s API for Android, the quickest and easiest way to get great sound and music into your Android games. This document will show you how to get started implementing FMOD Ex in your game by pointing you in the direction of detailed API documentation and support resources. While the FMOD Ex Programmer’s API presents the same interface on all platforms, each platform does have its own unique features and limitations - Android-specific features/limitations will be listed here along with any hints and tips for getting the most out of FMOD Ex on the Android.

Have fun implementing great audio and drop us a line some time,

The FMOD Team
Melbourne, Australia
www.fmod.org

Support Resources

API documentation

Detailed API documentation can be found in the “documentation” directory/folder of your FMOD Ex Programmer’s API installation. This documentation is your main reference for information on FMOD Ex API classes and functions.

Forums

http://www.fmod.org/forum

This should be your first port of call for further FMOD information and questions on implementation. If you have a question related to FMOD, chances are someone else has already asked it. The FMOD forums are free for all FMOD users and are monitored by the FMOD team as well as being home to a strong community of FMOD developers, from student first-timers to top-level professionals working on games that are household names.

Email

support@fmod.org

This is our main technical support line. It’s monitored directly by the FMOD team and we aim to answer all emails within 24 hours. It’s free for all FMOD users and your issues will be addressed directly by the guys who wrote the code. If you can’t find an answer to your problem on the FMOD forums, shoot us an email and we’ll get right onto it.

Videos

http://www.youtube.com/FMODTV

The FMOD YouTube channel contains a growing number of videos of tutorials relating to FMOD and FMOD Designer. This channel is being added to all the time, so be sure to check back regularly.

Installation

Libraries

FMOD comes with C/C++ libraries which can be called via JNI, or linked to the C/C++ component of your application:

- **libfmodex.so** for general development.
- **libfmodexL.so** for the same library, but with debug logging which can help to determine any problems if they exist.

We currently provide FMOD libraries for the ***armeabi*** and ***armeabi-v7a*** ABIs which are built against the **android-3** API level for maximum compatibility. The library location follows the format **api/lib/$(ABI)**

NOTE: Previously we provided an **android-9** library for OpenSL ES support, this has since been made a dynamic component in our **android-3** library simplifying your choice of libs.

If you wish to use the Audio Track output mode you will also need an FMOD jar file, it must to be added to your Java application to initialize and drive the FMOD audio output:

- **fmodex.jar**

Java Initialization

To enable audio output through the Audio Track output mode you must include and initialize the FMOD Java audio driver in your application (not required if you use the OpenSL output mode). To do this you will need to ensure **fmodex.jar** is referenced in your Java project, and you have imported the **org.fmod.FMODAudioDevice** package. The FMOD Java audio driver also requires that your application loads the fmodex library.

The **FMODAudioDevice** class has two functions you must call for audio to play, they can be called at any time, but we recommend you put the **start()** and **stop()** function calls in the **onStart()** and **onStop()** overrides of your Activity. Below is a fragment from the 'playsound' example that illustrates this (please refer to the examples for a complete demonstration).

```
import org.fmod.FMODAudioDevice;

public class Example extends Activity
{
    private FMODAudioDevice mFMODAudioDevice = new FMODAudioDevice();

    @Override
    public void onStart()
    {
        super.onStart();
        mFMODAudioDevice.start();
    }

    @Override
    public void onStop()
    {
        mFMODAudioDevice.stop();
        super.onStop();
    }

    static
    {
        System.loadLibrary("fmodex");
    }
}
```

Running Examples

The FMOD examples are designed to follow a similar process to the official NDK samples, utilizing **ndk-build**. Please make sure the NDK is set up correct and official examples work before proceeding (see http://developer.android.com/sdk/ndk for details). We have made some attempts to streamline the usage of the examples by providing Eclipse projects, however some setup is still required to get them working. Please note examples won't build if there are spaces in the project path (this is a GNU make limitation).

1. Import the desired example into your workspace:
   File -> Import -> General -> Existing...
2. Resolve the *FMOD_LIB* reference by setting a class path variable to the installed file:
   Window -> Preferences -> Java -> Build Path -> Classpath Variables -> New
   FMOD_LIB = "C:\FMOD\api\lib\fmodex.jar"
   FMOD install path ('C:\FMOD') may vary depending on your setup.
3. Ensure appropriate environment variables are set on your machine or in the project via:
   Project -> Properties -> C/C++ Build -> Environment -> Add
   ANDROID_NDK_ROOT = C:\Android\ndk
   NDK install path ('C:\Android\ndk') may vary depending on your setup.
4. Copy all the files from the two sets of example media "examples\media\*" and "fmoddesignerapi\examples\media\*" to a folder called '*fmod*' in the root of the SD card.

Audio Latency

Reducing the amount of audio latency between calling an API function and hearing its effect is generally controlled via System::setDSPBufferSize. However it should be noted that on this platform there is significantly more OS latency (which is out of the control of developers). It is currently not mandatory for device manufactures to adhere to audio latency guidelines (section 5.3 Audio Latency of the Android CDD). Only devices which report 'android.hardware.audio.low_latency' from 'adb shell pm list features' will be able to achieve a true low latency audio experience. Testing for low latency can be done at runtime by checking FEATURE_AUDIO_LOW_LATENCY.

Troubleshooting

Find solutions for common platform specific issues here:

A quick note. Use the logging version of FMOD to get information in the tty or output log file.

Pulsating tone is suddenly audible

This is a fatal warning from FMOD's mixer. It means the mixer tried to allocate some memory and failed. Because of unexpected behavior at this point, the mixer sends a pulsating sine wave out through the speakers to let you know of this fact.

The solution for this is to reduce memory usage or provide more memory to FMOD, then restart the application.

Note that the tty/log output will display out of memory error messages, and System::setCallback can be used in the API to catch out of memory errors with `FMOD_SYSTEM_CALLBACKTYPE_MEMORYALLOCATIONFAILED`.