



JULIAN LÖHR

AR/MR/VR/HoloLens Developer

TAG: NDK

18. DECEMBER 2014

OpenAL-MOB with openFrameworks on Android

Introduction

On work i am currently working on a mobile game project. We are using openFrameworks v0.8.4 in combination with OpenAL for Audio output because we need the HRTF for binaural sounds. The OpenAL implementation is OpenAL-MOB, since it's an OpenAL-Soft fork with HRTF support and especially supports mobile platforms.

To get OpenAL-MOB integrated into openFrameworks we are using our own wrapper, but the openFrameworks Addon ofxALSoft will do just fine. Just merge the OpenAL-MOB folder into the ofxALSoft addon folder and you are ready to go.

iOS

Our main target platform is iOS, so my development system is a Mac OS X. I am therefore referring to the Mac Terminal, but for Linux and Windows the steps should be the same.

To get OpenAL-MOB running and deployed on an iOS Device is quite easy. Just add the ofxALSoft addon and the OpenAL-MOB-ios XCode project to your openFrameworks project in XCode. XCode compiles OpenAL-MOB and links it into your app on its own. So everything is fine.

Android

Our second target platform is Android and the last couple of workdays i tried to get the project running on our Android device.

Currently openFrameworks supported Android IDE is still Eclipse, so i am referring to Eclipse. If you use Android Studio you might to have adjust some additional things.

The problem with openFrameworks for Android is it uses custom makefiles to compile your openFrameworks project and create a shared library. Additionally it is ignoring the Android.mk and Application.mk generating them each build.

OpenAL-MOB provides Android makefiles but since openFrameworks ignores its Android.mk you can't create a dependency. Even worse due to openFrameworks custom makefiles the addons are compiled as well and everything is linked into one shared library. In this linking process the ofxALSoft addon needs something to be linked again, which is the OpenAL-MOB lib, that needs to be built beforehand.

Solution

I came up with a way to get it running anyway. I didn't create a dependency to automatically build OpenAL-MOB each deploy, but rather build the library manually. However if you won't change OpenAL-MOB you just to need to build it once.

The first step is to build a static library of OpenAL-MOB and then include it into the linkage of the openFrameworks shared lib of your project.

Building OpenAL-MOB

Adjusting the Makefile

Open **Android.mk** in "**Path/to/of/addons/ofxALSoft/build_android/jni/**".

First we need to add one compiler flag otherwise we will get some linker errors later when building the openFrameworks project. So add beneath the "**ifreq ... else ... endif**" block around the

LOCAL_CFLAGS:

```
LOCAL_CFLAGS += -fno-stack-protector
```

Second change it to build a static library instead of a shared one. To do so change the last line from

```
include $(BUILD_SHARED_LIBRARY)
```

into

```
include $(BUILD_STATIC_LIBRARY)
```

At the end the bottom half of the file should look like:

```
# set the platform flags
ifeq ($(APP_ABI),x86)
  LOCAL_CFLAGS += -D HAVE_SSE
else
  LOCAL_CFLAGS += -D HAVE_NEON -mfloat-abi=softfp -mfpu=neon -marm
endif

LOCAL_CFLAGS += -fno-stack-protector

#LOCAL_SHARED_LIBRARIES += libOpenSLES
LOCAL_LDFLAGS += -lOpenSLES

#include $(BUILD_SHARED_LIBRARY)
include $(BUILD_STATIC_LIBRARY)
```

Compiling

Now we are ready to compile it. The ndk-build tool is a little bit broken (or it was in revision 9b) because it evaluates the **ifreq ...** block just once in a call so we need to run two compile passes. First compile for ARM and then in a second run for x86.

Open **Application.mk** in **"Path/to/of/addons/ofxALSoft/build_android/jni/"** and set

```
APP_ABI := armeabi armeabi-v7a
```

Now open a Terminal and navigate with `cd` to “**Path/to/of/addons/ofxALSoft/build_android/jni/**”. Run “**Path/to/ndk/ndk-build**” in this directory to compile it.

If you get compile errors because he can't find some header files, specify the **APP_PLATFORM** argument. You can either add it to your **Application.mk** file or pass it as command line argument to `ndk-build`. And make sure the specified Android Version is included in the NDK package.

After building the static libraries for ARM open “**/addons/ofxALSoft/build_android/obj/**” and rename the folder **local** to **android**. OpenFrameworks will later search for the static lib in a directory structure similar to “***/android/\$(abi)/libname.a**”. Furthermore `ndk-build` cleans previously build static libs for all target in the **local** directory when run. So it would delete our previously build libs on the second build pass.

For the second build pass set

```
APP_ABI :=x86
```

in the **Application.mk** and run `ndk-build` again.

Copy the “**local/x86**” folder into “**android/x86**”.

We now have three static libraries of OpenAL-MOB for each architecture. And can start to add them to the openFrameworks project.

Adding it to your Project

At last we need to include the OpenAL-MOB library into the openFrameworks make process.

Navigate to your openFrameworks project folder and open **config.make**.

First we need to add the library headers to be included for the addon. Add

```
PROJECT_EXTERNAL_SOURCE_PATHS += $(OF_ROOT)/addons/ofxALSoft-MOB/include
```

at the appropriate position.

Then we need to include the static lib into the linkage. Add

```
PROJECT_LDFLAGS += $(OF_ROOT)/addons/ofxALSoft-MOB/build_android/obj/$(ABI_LIB_
```



for the linker flag.

Now hit **run** or **build all** in eclipse and your project should successfully compile and deploy to your device.

Conclusion

As said the OpenAL-MOB library is built manually, but unless you modify it you just need to build it once and can add it to your repository as prebuilt library. There might be a way to include the build into the openFrameworks makesystem but since i don't need that i am skipping that step.