Jürgen Menge

Betreff:	Android Studio: migrating libraries with C/C++ from gradle-experimental with Model plugin to gradle + CMake / ndkBuild
Anlagen:	build.gradle; build.gradle; CMakeLists.txt; Android.mk; Application.mk

This was then …

For our technology we have a portfolio of c programs working in Windows (VisualStudio), iOS (Xcode), Android (Android Studio).

Available as SDK in form of a library for Xcode and Android development. (Also using the respective audio interface, especially, recording from the devices' microphone(s).)

Everything worked out nicely, and for Windows and iOS there was no problem migrating said library into 64 bit with the advent of iOS 10 and newer devices, say, starting with iPod Touch gen.6.

It became a bit of work with Android…

In the earlier days with Android Studio (before that we had Eclipse and Android enhancements) we were using the "model" plugin available in gradle-experimental for the NDK and JNI features. Everything was working nicely.

Now Android Studio 2.3.3 and gradle-experimental:0.9.3.

But then it turned out we could not build our c programs in 64 bit (using abifilters "ARM64-8a"), due to missing hooks:
Known limitations for the Gradle component "model" implementation, "No support for using a NDK modules like cpu_features" and "No Support for integrating external build systems".

OK, so I started updating the project(s) to (here) Android Studio v.3.4.1. However, to my dismay, using the latest gradle-experimental:0.11.1 I got pages of exceptions reports during Gradle sync! The offending line, calling the "model" plugin:

```
apply plugin: 'com.android.model.library'
```

I filed with Google issue # 133092980 (https://issuetracker.google.com/issues/133092980)

Nevertheless, between reporting a problem and eventually getting a solution could mean … well, days? Months? Never?

Even though the Gradle blog "*State and future of the Gradle Software Model*" (https://blog.gradle.org/state-and-future-of-the-gradle-software-model) suggests work is being done, that blog is as of August 2017, and Google is actually encouraging to migrate away from the "model" plugin, and none of the often great sample projects make use of "model".

This is now …

So I started my "happy migration" project, going from Android Studio 2.3.3 with gradle:2.3.3 to Android Studio 3.4.1 with gradle:3.4.1. And encountered somewhat documented syntax and structural changes for the build.gradle scripts. And some trial-and-error ones.

A good starting point the documentation from the Google Android team; here: "*Android Studio Project, Experimental Plugin User Guide*" (http://tools.android.com/tech-docs/new-build-system/gradle-experimental).

As the Android team wrote so nicely, "There are significant changes to the DSL of the plugin. We understand that many of the changes are frustrating and seem unnecessary, and our goal is to remove some of these current changes to minimize the migration process from the traditional plugin in the future."

(BTW, Gradle is now on release 5.4.1 as of April 2019; the one in Android Studio is 3.4.1.)

To now get the C/C++ programs compiled using JNI and NDK I created CMake and, as alternative, NDK-Build scripts, just for kicks. Right: Gradle without the "model" plugin does **not** call and build directly the JNI modules. The make files need to be separately created and called through either CMake or NDK-Build within build.gradle!

The modifications in detail …

app / build.gradle (from gradle:2.3.3 to gradle:3.4.1)

1. remove ending semicolons (;), now unnecessary

2. rename all "compile" entries in "dependencies" to "implementation", "provided" to "compileOnly" (a few of the many Gradle syntax changes…)

3. add line:
   flavorDimensions 'vox'

4. add to each selection in "productFlavors" the line:
   dimension 'vox'

5. change in "variant.outputs.every" to "variant.outputs.all"

6. declare following
   relativeRootDir = output.packageApplication.outputDirectory.toPath()
   relativize(rootDir.toPath()).toFile()
   and instead of
   output.outputFile = new File(output.outputFile.parent,
   output.outputFile.name.replace("app-voxChat", apkNameVoxChat)))
   now use
   output.outputFileName = new File("$relativeRootDir"+"/"+"app",
   "${appName}".replace("app-voxChat", apkNameVoxChat()))

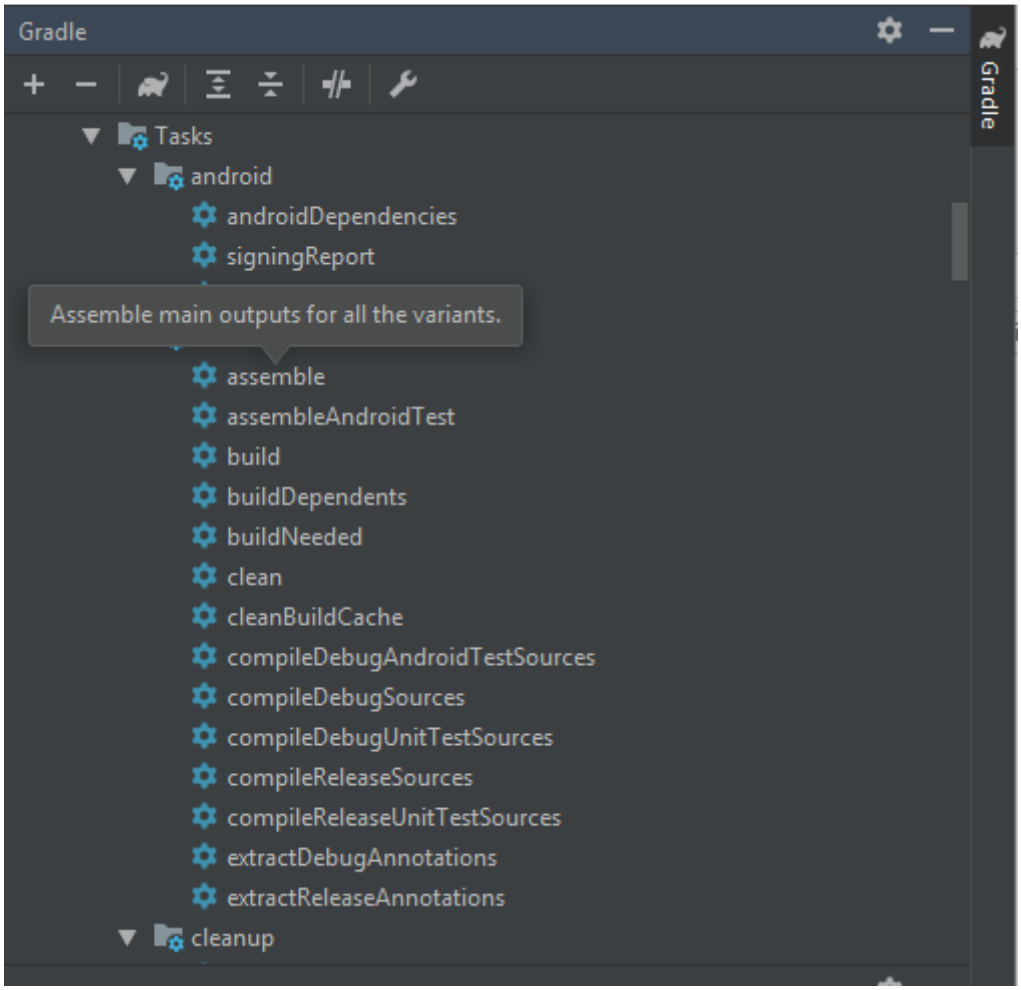
wavvoxlibrary / build.gradle (from gradle:2.3.3 / experimental:0.9.3 to gradle:3.4.1)

1. remove ending semicolons (;), now unnecessary
2. rename all “compile” entries in “dependencies” to “implementation”, “provided” to “compileOnly” (a few of the many Gradle syntax changes…)
    a. convert gradle-experimental to gradle syntax
    b. change “apply plugin: 'com.android.model.library' ” to “apply plugin: 'com.android.library' ”
    c. remove outside wrapped “model { }“ block
3. revert all changes marked in red per “*Android Studio Project, Experimental Plugin User Guide*”
4. rewrite the “ndk “ block without “cFlags” and “platformVersion”;
   remove “sources” or “sourceSets” block
5. add “externalNativeBuild” blocks to specify external build systems and their respective make files
    a. CMake: src/main/jni/CMakeLists.txt
    b. NDK-Build: src/main/jni/Android.mk + src/main/jni/ Application.mk (implicitly)
6. create said make file(s), also include CPU model(s) and compiler options

What’s left …

The Build – Signed .. APK is working nicely with the link of the compiled “.so” library files (here, both for “armeabi-v7a” and “arm64-v8a”), and the app runs on 32bit environment (say, Nexus 7 with Android 6.0.1) as well on 64bit environments (say, Xaomi M1 with Android 9).

To create the “.aar” library, you’ll need to explicitely execute the Gradle task, here, “wavvoxlibrary / Tasks / build / assemble”



the Android library is now available at ./wavvoxlibrary/build/outputs/aar/… and can be used and linked to other apps

Attached are …

1. ./wavvoxengine/build.gradle
    a. for Android Studio 2.3.3 (before) – using the component “Model”
    b. for Android Studio 3.4.1 (after) – set buildKind() to either “cmake” or “ndkBuild”
2. ./wavvoxlibrary/src/main/jni/
    a. CMakeLists.txt — buildKind() “cmake”
    b. Android.mk + Application.mk — buildKind() “ndkBuild”
3. This document …

Enjoy!

Cheers,

jm.

Jürgen Menge

San José, California