How to support OpenCV&NDK?

Step1: Support OpenCV

Ref: http://stackoverflow.com/questions/17767557/how-to-use-opencv-in-android-studio-using-gradle-build-tool/27356635#27356635

As per OpenCV docs(1), below steps using *OpenCV manager* is the recommended way to use OpenCV for **production** runs. But, OpenCV manager(2) is an additional install from Google play store. So, if you prefer a self contained apk(not using OpenCV manager) or is currently in*development/testing* phase, I suggest answer at http://stackoverflow.com/a/27421494/1180117.
Recommended steps for using OpenCV in Android Studio with OpenCV manager.

1.Unzip OpenCV Android sdk downloaded from OpenCV.org(3)
2.From `File -> Import Module`, choose `sdk/java` folder in the unzipped opencv archive.
3.Update `build.gradle` under imported OpenCV module to update 4 fields to match your project's `build.gradle` a) compileSdkVersion b) buildToolsVersion c) minSdkVersion and 4) targetSdkVersion.
4.Add module dependency by `Application -> Module Settings`, and select the `Dependencies` tab. Click `+` icon at bottom(or right), choose `Module Dependency` and select the imported OpenCV module.

As the final step, in your Activity class, add snippet below.

Note: *You could only make OpenCV calls after you receive success callback on `onManagerConnected` method*. During run, you will be prompted for installation of OpenCV manager from play store, if it is not already installed. During development, if you don't have access to play store or is on emualtor, use appropriate OpenCV manager apk present in `apk` folder under downloaded OpenCV sdk archive .

In the final test, we can write a java file using OpenCV's build-in prototype like `Mat` or other functions. In this way, we can know that it works well.

Step2: Support NDK at the same time

First, use the guide here: https://codelabs.developers.google.com/codelabs/android-studio-jni/index.html?index=..%2F..%2Findex#0

But even with the step by step guide, we cannot get it working, due to:

1) studio's version must match(v2.1) and the gradle's version(2.8)

2) OpenCV's gradle issue.

Then we direct to this presentation: http://www.slideshare.net/ph0b/mastering-the-ndk-with-android-studio-and-the-gradleexperimental-plugin

This slide gives some details about how to use the **experimental-gradle** for using ndk, because

it's not officially merged, and we cannot use it(so many errors when I just use a hello-world example)

In this post: http://stackoverflow.com/questions/17727645/how-to-update-gradle-in-android-studio
I get to know how to update gradle, but even so using newest studio and gradle still not works.

At last, according to the slides and presentation, I get to know how to use the **expermental-gradle**, actually android studio has a **different** mechanism from eclipse.
In eclipse, when we want to use plugin, we have to install it via the installation menu, like ADT, pyDev or other plugins. But here in android studio, we use the **experimental-gradle** by changing the gradle file, and then syncing, this will help it download and applying the new configurations.

But this step is not very simple, it involves:
1) modify the location of new gradle, as following project's topmost `build.gradle` shows:

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProjec
```

2) Change the gradle's location using a newer version(**v2.8 -> v2.10**, which is for experimental plugin's use), see the project - gradle - wrapper - gradle-wrapper.properties:

```
#Wed Oct 21 11:34:03 PDT 2015
distributionBase=GRADLE_USER_HOME
```

```
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-2.10-all.zip
```

3) Change **each subproject's** build.gradle, here our project is quite complex, because it involves the OpenCV subproject, it's not application but library, that we have to support at the same time, use new **experimental-gradle**, we no longer use old syntax of gradle,

Before:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.1"

    defaultConfig {
        applicationId "com.google.sample.helloandroidjni"
        minSdkVersion 22
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
// others below this line: no change
```

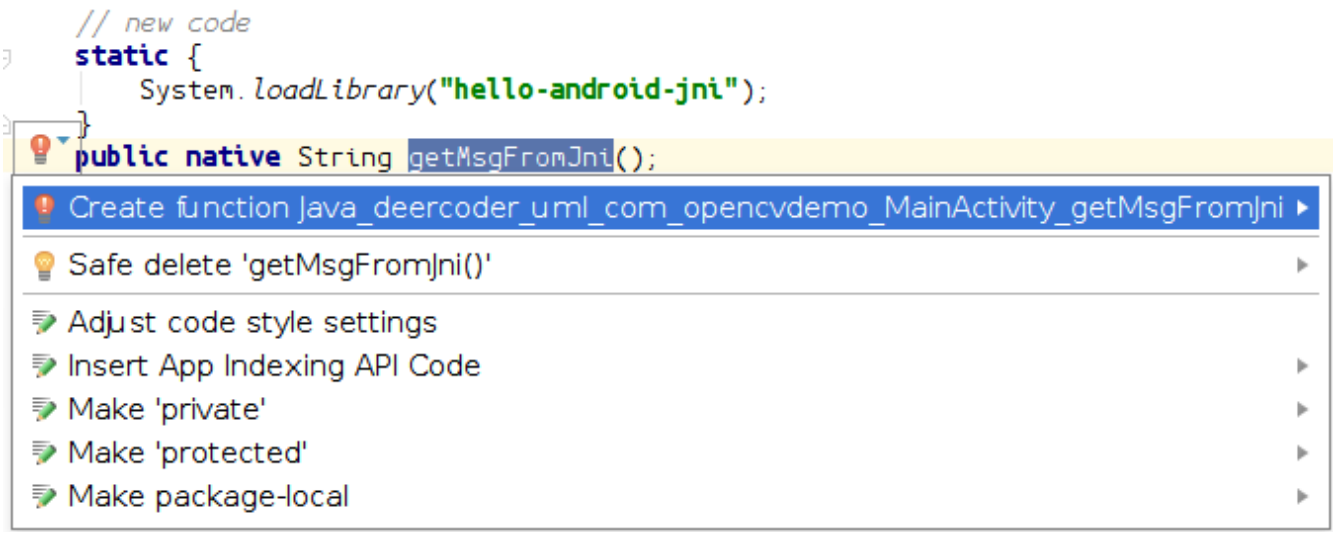
Now:

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

model {
    android {
        compileSdkVersion = 23
        buildToolsVersion = "23.0.1"

        defaultConfig {
            applicationId = "com.google.sample.helloandroidjni"
            minSdkVersion.apiLevel = 22
            targetSdkVersion.apiLevel = 23
            versionCode = 1
```

```
        versionName = "1.0"
    }
    buildTypes {
        release {
            minifyEnabled = false
            proguardFiles.add(file('proguard-android.txt'))
        }
    }
}
// others below this line: no change
```

We can see that the `apply plugin` and `android` or `model` have all been changed, follow the guide to change it.
At last, use this configuration to sync and build with gradle(Don't add any ndk() here, it will make it fail...)

We have to first make it build pass with the experimental plugin, and then add ndk syntax and rebuild with new modification.



After the above configuration, we have get the **experimental-gradle** build, then we can work on adding support for ndk, as this one
shows: https://codelabs.developers.google.com/codelabs/android-studio-jni/index.html?index=..%2F..%2Findex#3

1) modify `build.gradle` to add support of ndk:

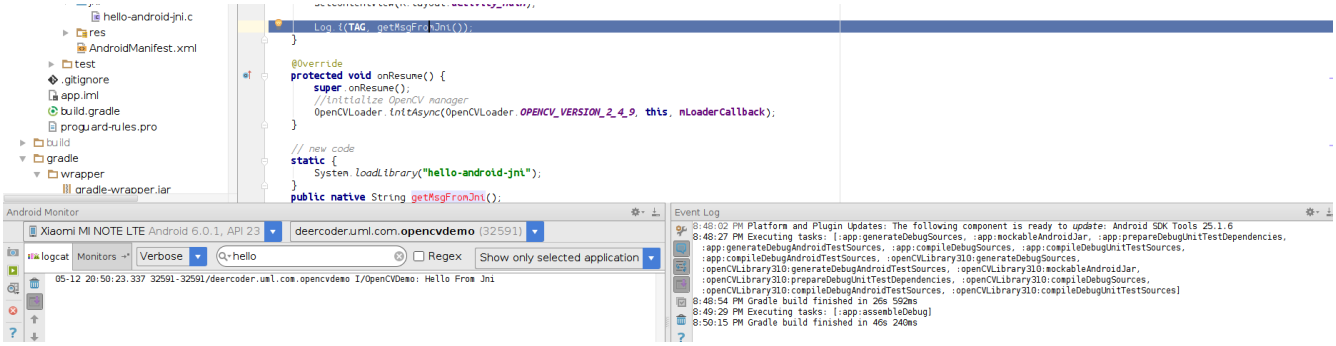
```
buildTypes {
...
}
// New code
ndk {
    moduleName = "hello-android-jni"
}
```

```
// New code finished
```

2) add code in our body, which is XXXActivity.java, and also the .c and .h files

```
...
    // new code
    static {
        System.loadLibrary("hello-android-jni");
    }
    public native String getMsgFromJni();
    // new code done
} // class MainActivity
```

3) sync and build, if there is some error like the above figure shows, then we can use the hint to automatically insert the code. **NOTE: the above figure shows that we have installed the experimental-gradle successfully, that's a good indication, otherwise, if we don't have the hint, then our previous steps are failed, we have no idea why ndk and jni not working in that case, keep repeating the steps until we're sure each step is OK and shows the correct information**



as the above figure shows, we have successfully build the `hello-world` example as well as the **OpenCV** support. It will output the `hello world` string, but using the JNI call method, and it also outputs the **OpenCVDemo** string, which means that OpenCV's `Mat` structure is also used at the same. We have made it by using OpenCV & ndk at the same time!

Next, we can do more by porting the OpenCV's sample and our algorithm based on this. Currently we can support OpenCV and NDK at the same time, which is the basis for our future project and extension.