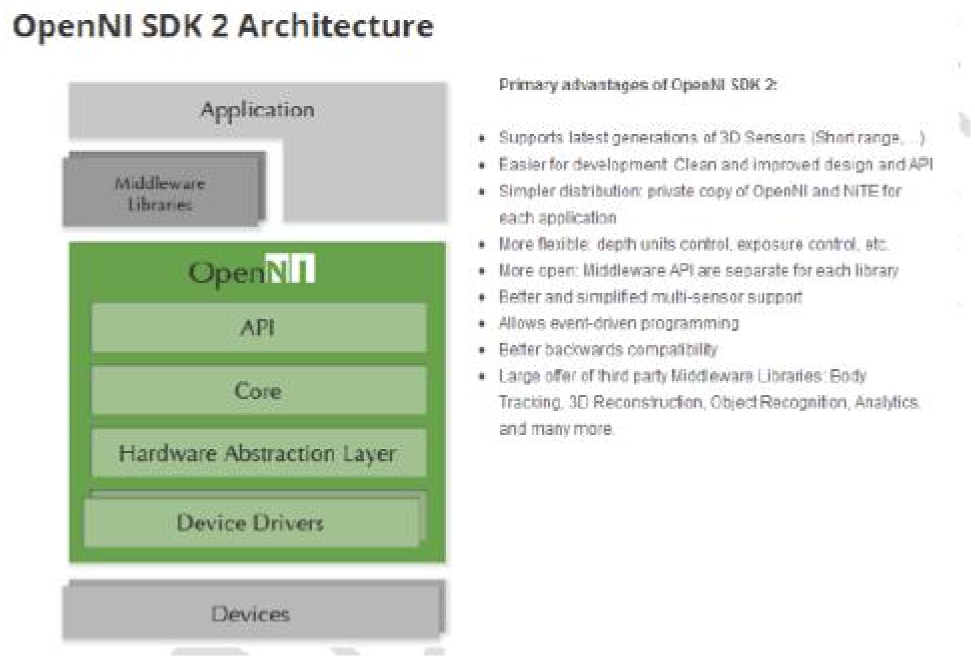
# Orbbec SDK development and use instructions

table of Contents

## 1 Introduction

Orbbec SDK is developed based on OpenNI2. OpenNI (Open Natural Interaction) is a multi-language, cross-platform software framework that defines the interface between applications, middleware and 3D sensing devices. Obi Zhongguang’s Astra series cameras fully support the OpenNI protocol, which means that any application developed using OpenNI can be seamlessly connected to Astra devices. OpenNI2 is the second generation version of OpenNI and the latest version. When using the Orbbec SDK, it is assumed that you have a certain understanding of OpenNI2. If you have not contacted OpenNI related development before, please read the "OpenNI2 Programming Brief Instructions" or refer to the official OpenNI2 help document.



Primary advantages of OpenNI SDK 2:

- Supports latest generations of 3D Sensors (Short range, ...)
- Easier for development: Clean and improved design and API
- Simpler distribution: private copy of OpenNI and NiTE for each application
- More flexible: depth units control, exposure control, etc.
- More open: Middleware API are separate for each library
- Better and simplified multi-sensor support
- Allows event-driven programming
- Better backwards compatibility
- Large offer of third party Middleware Libraries: Body Tracking, 3D Reconstruction, Object Recognition, Analytics, and many more

## 2. Scope of application

This document is applicable to Astra series cameras, including but not limited to all series of Astra, Astra Pro, and Astra Mini. Applicable platforms include windows 7 and above, Ubuntu 14.04 and above.

## 3. Windows platform

### 3.1 Windows environment configuration

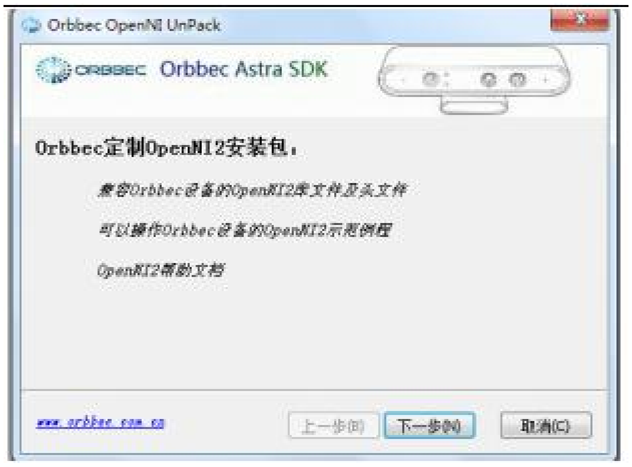
Before using OpenNI2 for development on Windows, please make sure that the relevant drivers have been installed, the Orbbec device can be viewed in the device manager, and the normal depth image can be seen using the NiViewer tool of OpenNI2. Please refer to the detailed installation process

"Astra Device Driver Installation and Device Diagnosis Guide".

### 3.2 SDK installation

The SDKs we provide for the Windows platform include OpenNI-Windows-x64-2.3.1 and OpenNI-Windows-x86-2.3.1

1) Click OpenNI-Windows-x64-2.3.1 / OpenNI-Windows-x86-2.3.1 according to the corresponding operating system to install.



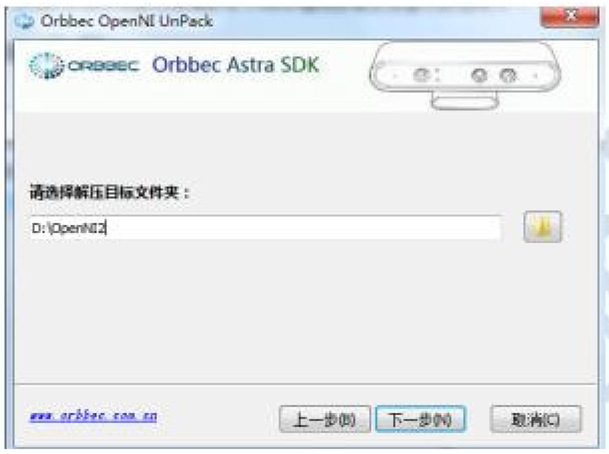
Orbbec定制OpenNI2安装包：

兼容Orbbec设备的OpenNI2库文件及头文件

可以操作Orbbec设备的OpenNI2示范例程

OpenNI2帮助文档

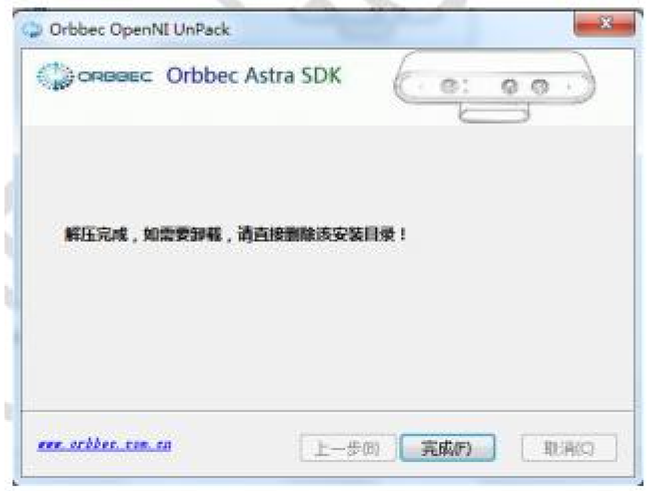
2) Click Next to select the installation directory.



请选择解压目标文件夹：

D:\OpenNI2

3) The installer will decompress the corresponding library files and header files of OpenNI2 to the specified directory and set the environment variables. If you need to uninstall the SDK, just delete the directory directly.



解压完成，如需要卸载，请直接删除该安装目录！

4) Click Finish to end the SDK installation.

3.3 Development Instructions

The Orbbec SDK mainly includes the following contents:

Documentataion: OpenNI2 API documentation and development manual;

Include: The header files needed to develop OpenNI2 programs;

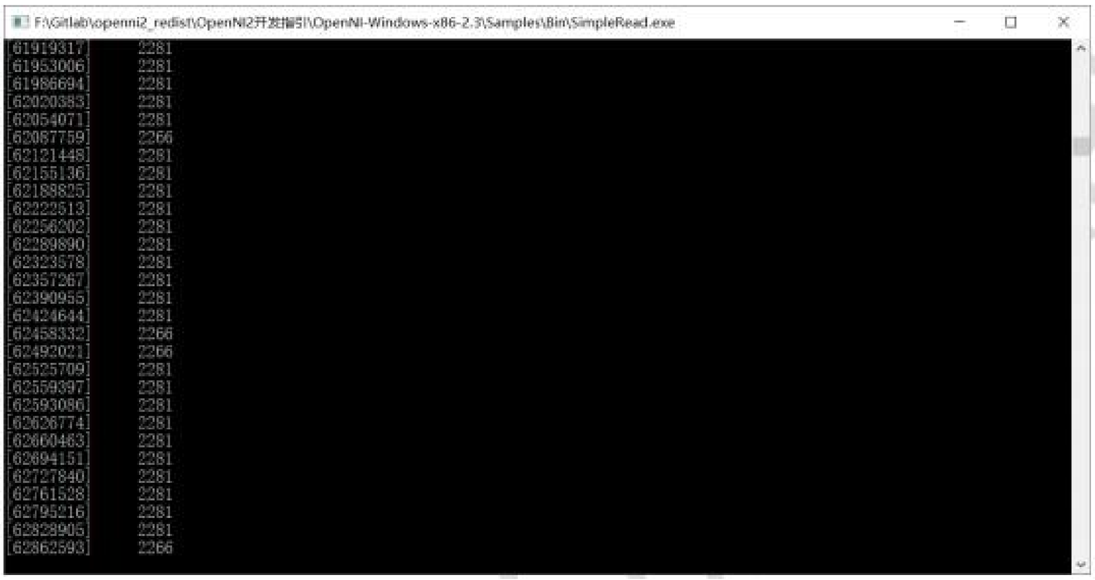
Lib: The static library that needs to be linked to develop OpenNI2 programs;

Redist: the dynamic library required to run the OpenNI2 program;

Samples: OpenNI2 related development routines;

The following describes the important routines SimpleRead and SimpleViewer in the SDK. SimpleRead is a read image middle

Example of the depth value of the point. After connecting the camera, run Samples\Bin\SimpleRead.exe to see the following effects:

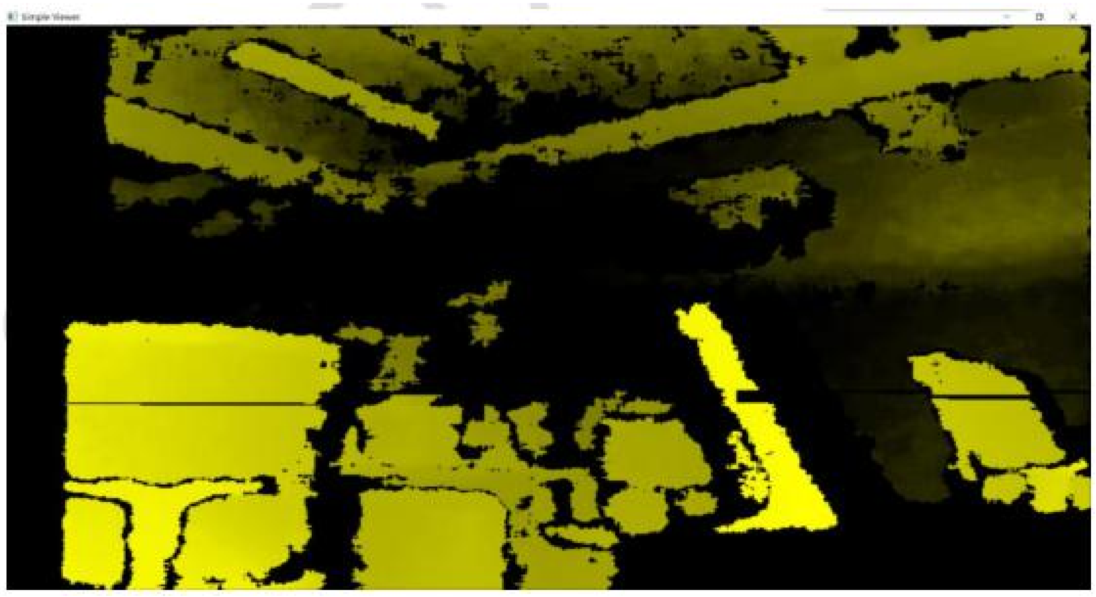

F:\Gitlab\openni2_redist\OpenNI2开发指引\OpenNI-Windows-x86-2.3\Samples\Bin\SimpleRead.exe

```
[61919317]      2281
[61953006]      2281
[61986694]      2281
[62020383]      2281
[62054071]      2281
[62087759]      2266
[62121448]      2281
[62155136]      2281
[62188825]      2281
[62222513]      2281
[62256202]      2281
[62289890]      2281
[62323578]      2281
[62357267]      2281
[62390955]      2281
[62424644]      2281
[62458332]      2266
[62492021]      2266
[62525709]      2281
[62559397]      2281
[62593086]      2281
[62626774]      2281
[62660463]      2281
[62694151]      2281
[62727840]      2281
[62761528]      2281
[62795216]      2281
[62828905]      2281
[62862593]      2266
```

For the specific implementation code, please refer to the project in the Samples\SimpleRead\ directory. You can try to compile it with VS yourself.

SimpleViewer is a routine to read and display the color map and depth map. After reading the data, use OpenGL to draw the data.

After connecting the camera, run Samples\Bin\SimpleViewer.exe to see the following effects:



There are several shortcut keys you can try:

1-Depth and color overlay display;
2-Only show the depth map;
3-Only display color images;
m-switch mirror display;
For the specific implementation code, please refer to the project in the Samples\SimpleViewer\ directory. You can try to compile it with VS yourself.

## 4. Linux Platform

### 4.1 Linux environment configuration

1) Before using OpenNI2 SDK for related development on the Linux platform, please execute the following commands to install and configure the related environment.
Take the x64 platform as an example, in order to run related programs, you first need to install the following programs

```
$ sudo apt-get install build-essential freeglut3 freeglut3-dev
```

2) Check the udev version, Orbbec's driver needs to depend on libudev.so.1, you can execute the following command to find out, if not, you need to manually link libudev.so.x.x to libudev.so.1

```
$ldconfig -p | grep libudev.so.1
$cd /lib/x86_64-linux-gnu
$sudo ln -s libudev.so.x.x.x libudev.so.1
```

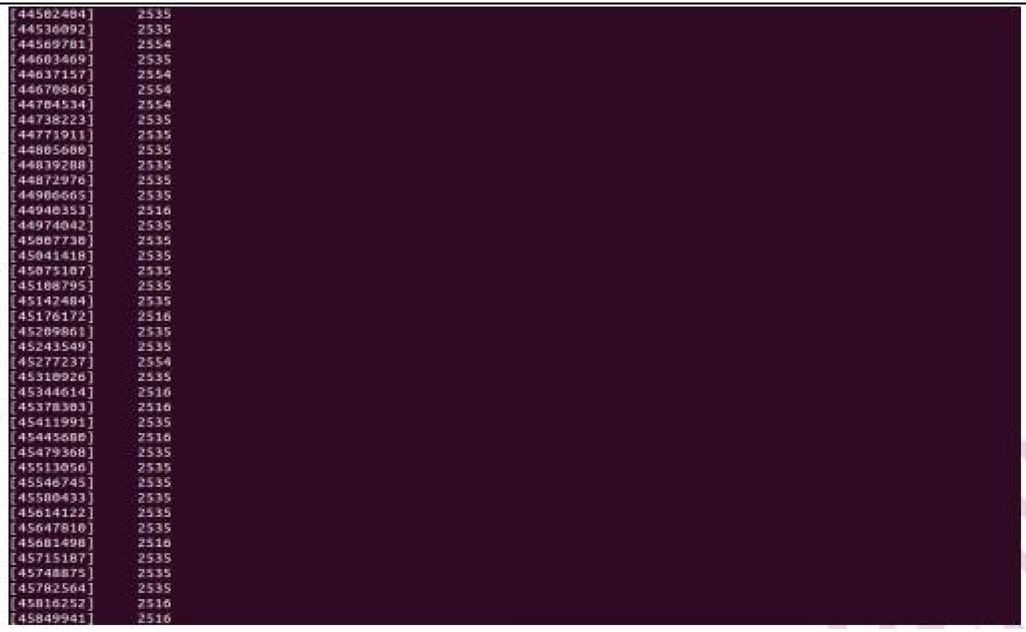
3) Copy the Linux OpenNI2 SDK package to a certain path, unzip it, and then run the install script in the unzipped directory to generate OpenNI2 related environment variables

```
$ tar zxvf OpenNI-Linux-x64-2.2-0118.tgz
$ cd OpenNI-Linux-x64-2.2
$ sudo ./install.sh
$ source OpenNIDevEnvironment
```

4) Plug and unplug the device again for the setting to take effect.

### 4.2 Development Instructions

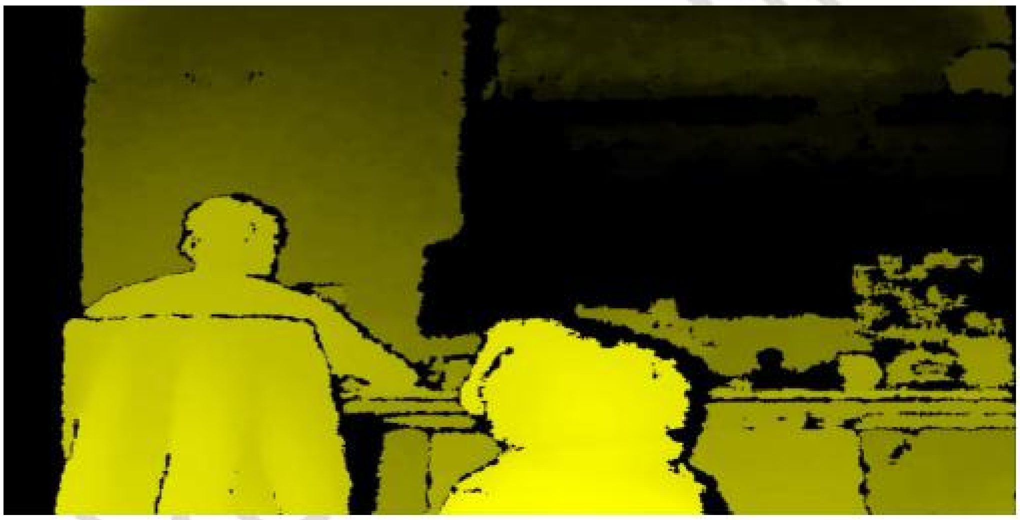
The SDK we provide for Linux platform includes OpenNI-Linux-Arm-2.3, OpenNI-Linux-Arm64-2.3, OpenNI-Linux-x86-2.3
And OpenNI-Linux-x64-2.3, mainly includes the following content:
Docementation: OpenNI2 API manual;
Include: the header files needed to develop OpenNI2 programs;
Redist: library files needed to compile OpenNI2 programs;
Samples: OpenNI2 related development routines;
Tools: NiViewer, a tool provided by OpenNI2;
The following describes the important routines SimpleRead and SimpleViewer in the SDK. We use the x64 platform Ubuntu 14.04 system
For example, SimpleRead is a routine that reads the depth value of the middle point of the image.
After connecting the camera, run it under Shell
Samples/Bin/SimpleRead can see the following effects:



```
[44502404]      2535
[44536092]      2535
[44569781]      2554
[44603469]      2535
[44637157]      2554
[44670846]      2554
[44704534]      2554
[44738223]      2535
[44771911]      2535
[44805600]      2535
[44839288]      2535
[44872976]      2535
[44906665]      2535
[44940353]      2516
[44974042]      2535
[45007730]      2535
[45041418]      2535
[45075107]      2535
[45108795]      2535
[45142484]      2535
[45176172]      2516
[45209861]      2535
[45243549]      2535
[45277237]      2554
[45310926]      2535
[45344614]      2516
[45378303]      2516
[45411991]      2535
[45445680]      2516
[45479368]      2535
[45513056]      2535
[45546745]      2535
[45580433]      2535
[45614122]      2535
[45647810]      2535
[45681498]      2516
[45715187]      2535
[45748875]      2535
[45782564]      2535
[45816252]      2516
[45849941]      2516
```

For the specific implementation code, please refer to the source code in the Samples/SimpleRead/ directory. You can try to compile with the make command yourself.

SimpleViewer is a routine to read and display the color map and depth map. After reading the data, use OpenGL to draw the data. After connecting the camera, run Samples/Bin/SimpleViewer under Shell to see the following effects:



There are several shortcut keys you can try:

1-Depth and color overlay display;

2-Only show the depth map;

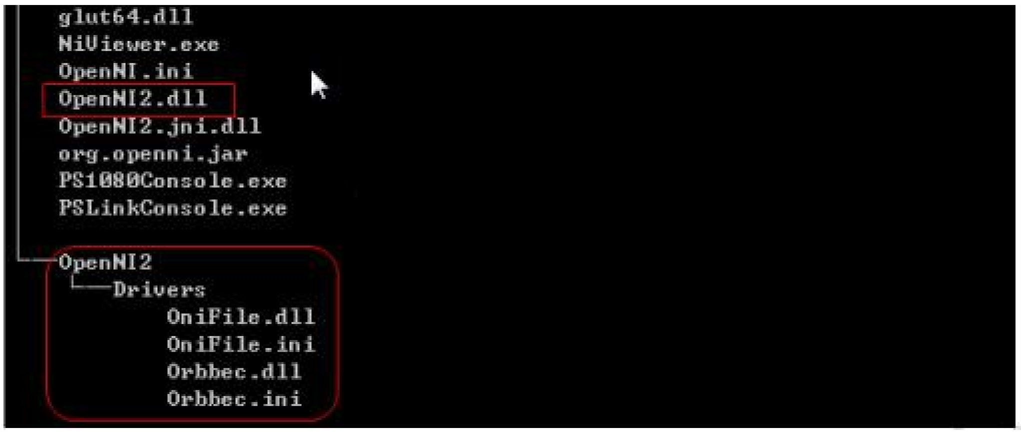
3-Only display color images;

m-switch mirror display;

For the specific implementation code, please refer to the project in the Samples\SimpleViewer\ directory. You can try to compile it with the make command yourself.

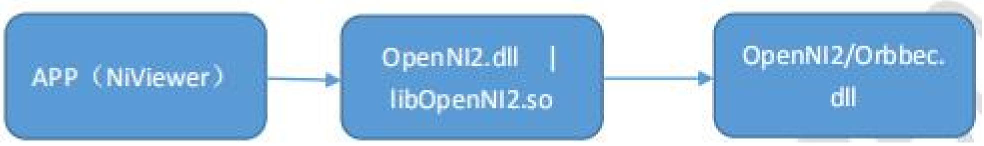
5. OpenNI2 program structure

A standard OpenNI2 program always follows the following file directory structure, or you can also change the directory structure by changing OpenNI.ini
Structure.



```
    glut64.dll
    NiViewer.exe
    OpenNI.ini
    OpenNI2.dll
    OpenNI2.jni.dll
    org.openni.jar
    PS1080Console.exe
    PSLinkConsole.exe

└───OpenNI2
    └───Drivers
            OniFile.dll
            OniFile.ini
            Orbbec.dll
            Orbbec.ini
```

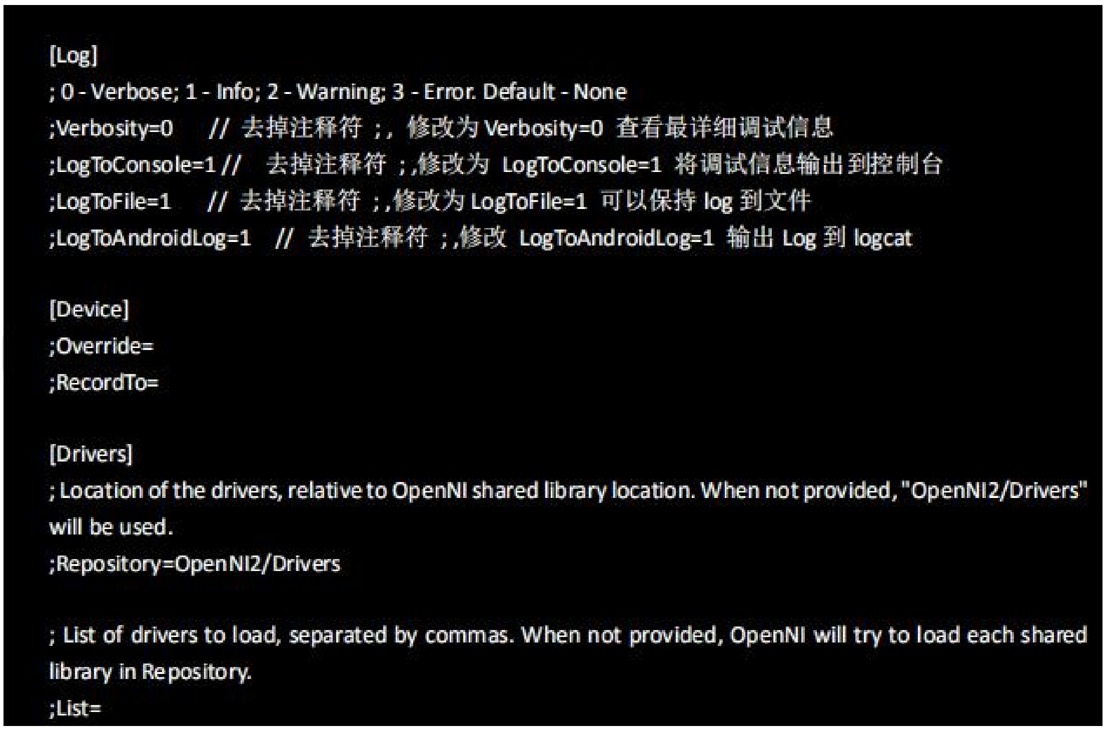
The hierarchy sequence of program call is as follows:



Orbbec.ini is saved as the initial configuration of Astra, such as resolution, output format, whether to mirror or not. In addition to changing these configurations through the code, you can also directly modify the corresponding parameters of the file to configure the Astra camera.

6. Open OpenNI2 debug log

OpenNI2 provides a wealth of debugging logs. When you turn on the corresponding level of Log, you can view the current depth, color image frame rate and other information on the screen or file. This function is mainly realized by modifying the OpenNI.ini file.



```
[Log]
; 0 - Verbose; 1 - Info; 2 - Warning; 3 - Error. Default - None
;Verbosity=0     // 去掉注释符 ;，修改为 Verbosity=0 查看最详细调试信息
;LogToConsole=1 //   去掉注释符 ;,修改为 LogToConsole=1 将调试信息输出到控制台
;LogToFile=1     // 去掉注释符 ;,修改为 LogToFile=1 可以保持 log 到文件
;LogToAndroidLog=1   // 去掉注释符 ;,修改 LogToAndroidLog=1 输出 Log 到 logcat

[Device]
;Override=
;RecordTo=

[Drivers]
; Location of the drivers, relative to OpenNI shared library location. When not provided, "OpenNI2/Drivers"
will be used.
;Repository=OpenNI2/Drivers

; List of drivers to load, separated by commas. When not provided, OpenNI will try to load each shared
library in Repository.
;List=
```

7. Windows/Linux sample code description

Orbbec SDK provides a series of sample codes based on OpenNI2 to help developers understand OpenNI related development, including how to turn on and off the device, obtain depth map, color map data, etc.

Note: There are two ways to obtain frame data in the data stream, polling and event callback. The polling method requires the user to manually control the data acquisition in the loop. The callback method is registered by OpenNI for the corresponding event, and the event is triggered when new data is available. It is recommended to use the polling method, the following code is not

Explain that the callback method is used, and the polling method is used to obtain data.

SimpleRead-This code is the most basic code for using OpenNI2. It is recommended that users view this code first. The program is a console program,

The main functions are how to open the device, create a depth stream, how to obtain the depth frame data, and print the depth value without the center of the frame.

MultipleStreamRead-console program, similar to SimpleRead, but added to create a color stream, and demonstrate the function of simultaneously obtaining color stream data and depth stream data.

MultipleStreamRead-Callback-console program, the function is the same as MultipleStreamRead, but to get the number of color and depth frames

The method of data is changed from polling to event callback, and it demonstrates how to add device connection event monitoring function, frame synchronization function, etc.

SimpleViewer-Demonstrate how to write an OpenGL-based graphics program to open the device, create depth and color streams, and display the depth map and color map at the same time, and show how to histogram the depth data and display it graphically.

MultiDepthViewer – Demonstrate how to write a program to display the depth data of multiple RGBD under the same system, and display it on the screen graphically. You can switch

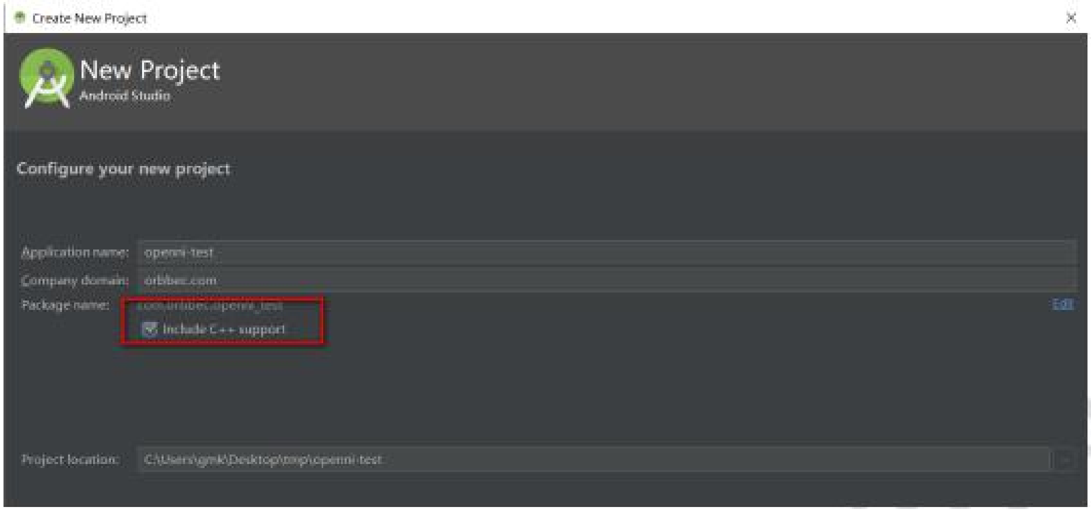
between different devices through the digital keys to view the depth map of the corresponding device.
EventBasedRead-The function is similar to SimpleRead, but the method of obtaining frame data is changed from polling method to event-based callback method.
MWClosestPoint-Demonstrate how to write a middleware library file based on OpenNI. The function of the middleware is used to analyze frame data and find the closest point to the camera.
MWClosestPointApp-Demonstrate how to use the library generated by MWClosestPoint to write a console program to print the point closest to the camera.
ClosestPointViewer – Demonstrate how to use the library generated by MWClosestPoint to write a GL graphics program to display the point closest to the camera.
Simpleviewer.Java – Demonstrate how to use Java to write a graphics program and open the color stream, the depth stream and display them together with ExtendedAPI-orbbec sdk's extended functions, demonstrate how to get the device serial number, device type, and read the calibration parameters saved by the camera
Data (Astra S only), save the calibration parameters to the camera, ir gain acquisition and adjustment, ir exposure acquisition and adjustment, switch LDP (laser protection equipment), switch laser

## 8. Android platform

### 8.1 Android Studio environment configuration

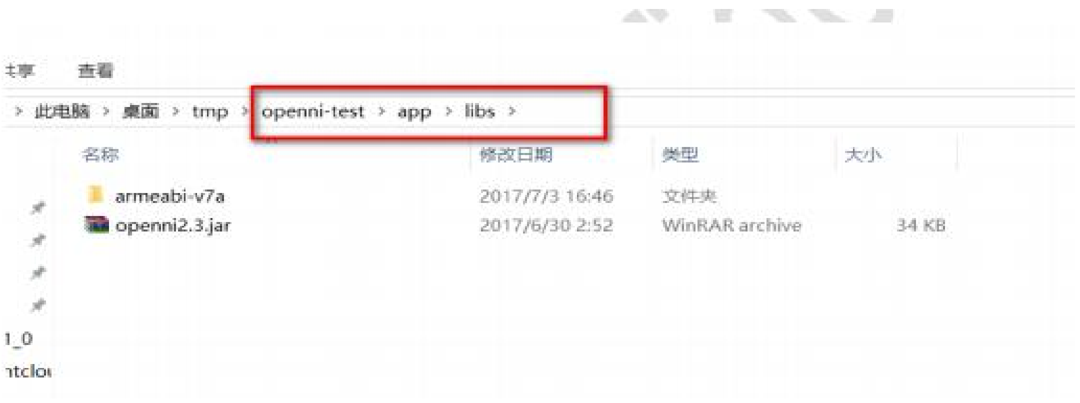
When you need to integrate the OpenNI2 SDK into your Android project, please refer to the following operations:

Use Android Studio to create a new project that supports C++ as openni-test

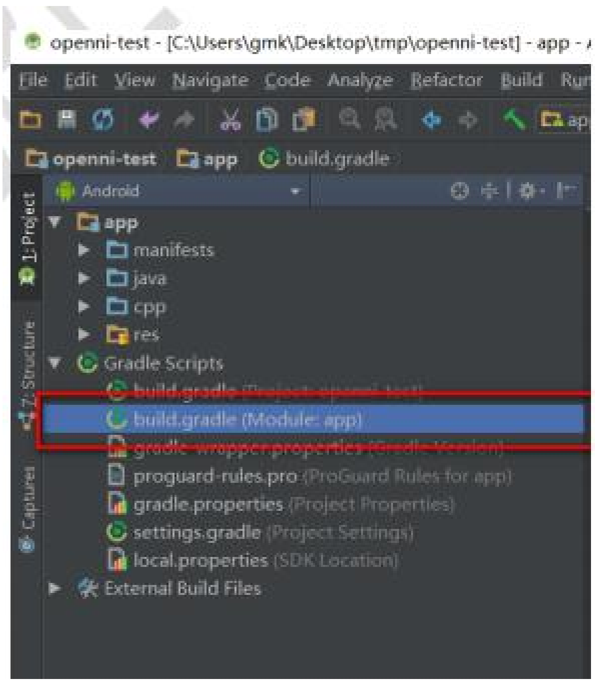


Application name: openni-test
Company domain: orbbec.com
Package name: com.orbbec.openni_test
[✓ Include C++ support]
Project location: C:\Users\gmk\Desktop\tmp\openni-test

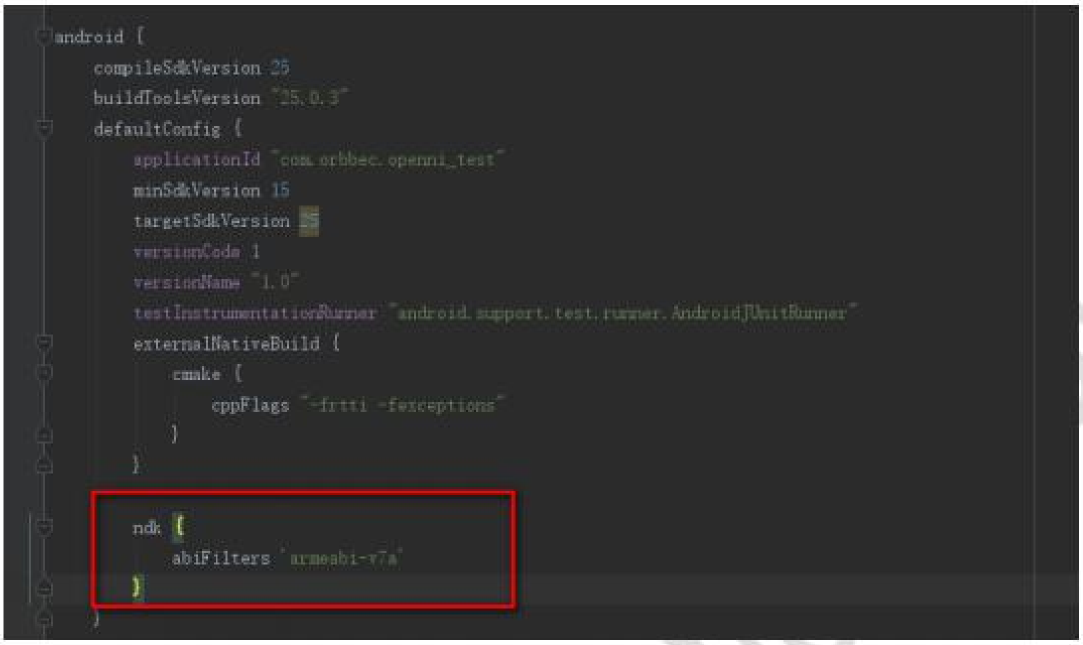
Copy the contents of the libs directory in the OpenNI_Android_32bit_2.3 package to the openni-test/app/libs directory:


共享 查看
> 此电脑 > 桌面 > tmp > openni-test > app > libs >
名称 修改日期 类型 大小
armeabi-v7a 2017/7/3 16:46 文件夹
openni2.3.jar 2017/6/30 2:52 WinRAR archive 34 KB

Do the corresponding project configuration in the Gradle file of the openni-test project, and the Jar configuration used

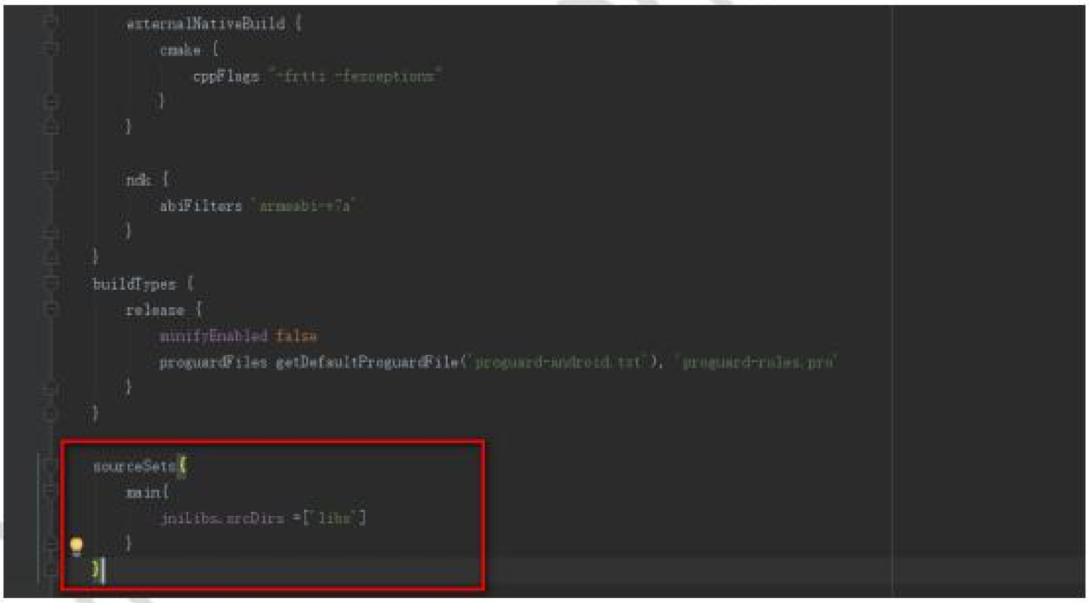

openni-test - [C:\Users\gmk\Desktop\tmp\openni-test] - app
File Edit View Navigate Code Analyze Refactor Build Run
openni-test app build.gradle
Android
app
manifests
java
cpp
res
Gradle Scripts
build.gradle (Project: openni-test)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
External Build Files

Add the following to the Gradle file, and add the abiFilters of ndk to the defaultConfig:



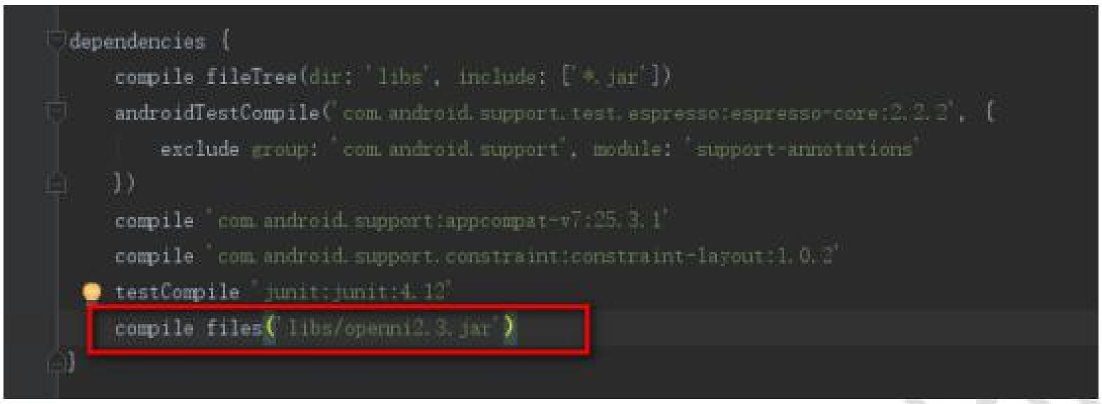
```
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.3"
    defaultConfig {
        applicationId "com.orbbec.openni_test"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        externalNativeBuild {
            cmake {
                cppFlags "-frtti -fexceptions"
            }
        }

        ndk {
            abiFilters 'armeabi-v7a'
        }
    }
```

Add the sourceSets configuration library file directory after buildTypes:



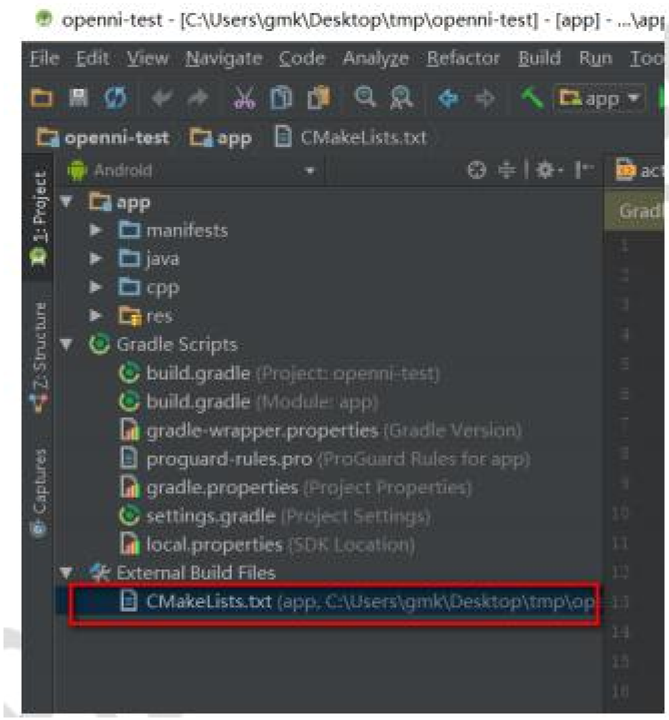
```
        externalNativeBuild {
            cmake {
                cppFlags "-frtti -fexceptions"
            }
        }

        ndk {
            abiFilters 'armeabi-v7a'
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    sourceSets{
        main{
            jniLibs.srcDirs =['libs']
        }
    }
```

In the dependencies, add the openni library file path that needs to be compiled

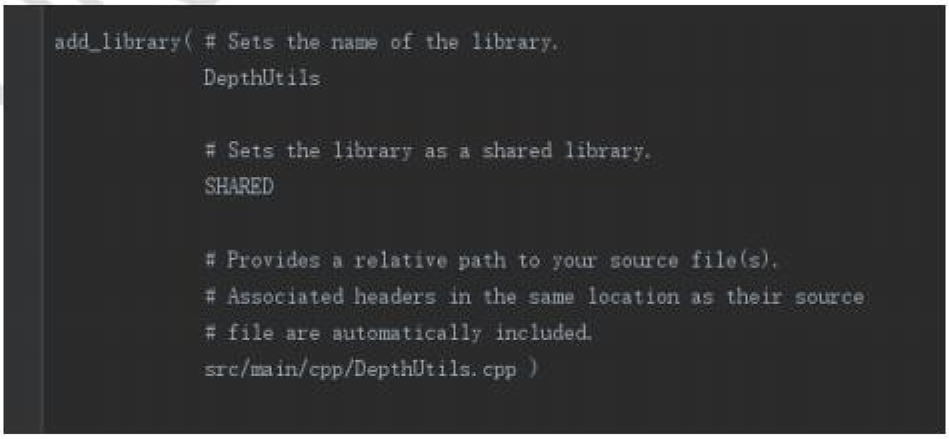


```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
    compile files('libs/openni2.3.jar')
}
```

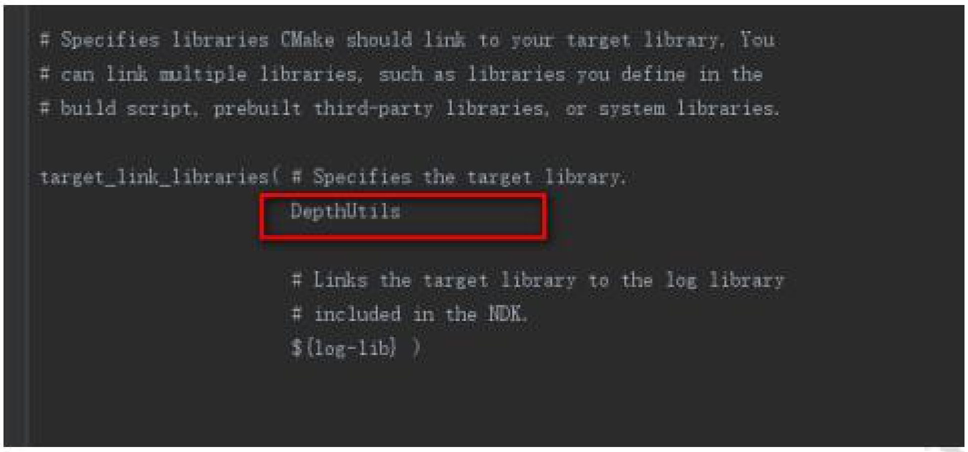
Do the corresponding cpp configuration in the cmake file of openi-test:


openni-test - [C:\Users\gmk\Desktop\tmp\openni-test] - [app] - ...\app
File Edit View Navigate Code Analyze Refactor Build Run Too
openni-test app CMakeLists.txt
Android
app
manifests
java
cpp
res
Gradle Scripts
build.gradle (Project: openni-test)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
External Build Files
CMakeLists.txt (app, C:\Users\gmk\Desktop\tmp\op

If you created the cpp file by yourself, you can refer to the configuration of the depthforopenni2 project,
The depthforopenni2 project created the DepthUtils.cpp file in the cpp directory, and the corresponding cmake configuration is

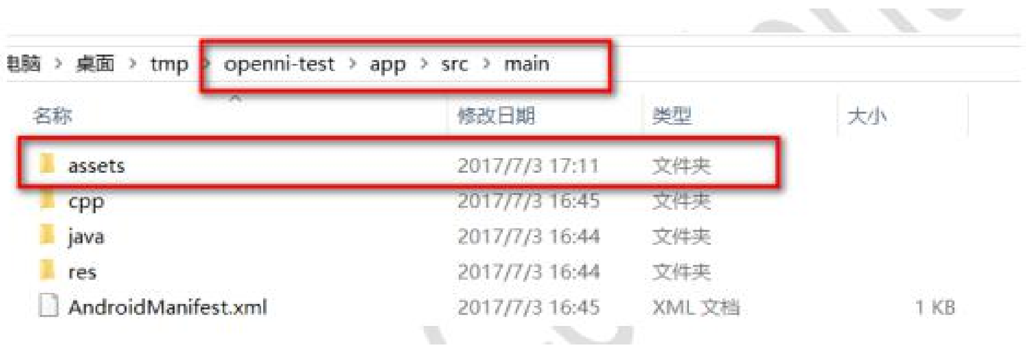


```
add_library( # Sets the name of the library.
             DepthUtils

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
             # Associated headers in the same location as their source
             # file are automatically included.
             src/main/cpp/DepthUtils.cpp )
```



```
# Specifies libraries CMake should link to your target library. You
# can link multiple libraries, such as libraries you define in the
# build script, prebuilt third-party libraries, or system libraries.

target_link_libraries( # Specifies the target library.
                       DepthUtils

                       # Links the target library to the log library
                       # included in the NDK.
                       ${log-lib} )
```

Copy the asserts directory in the OpenNI_Android_32bit_2.3 package to the openni-test/app/src/main directory:



电脑 > 桌面 > tmp > openni-test > app > src > main

| 名称 | 修改日期 | 类型 | 大小 |
|---|---|---|---|
| assets | 2017/7/3 17:11 | 文件夹 | |
| cpp | 2017/7/3 16:45 | 文件夹 | |
| java | 2017/7/3 16:44 | 文件夹 | |
| res | 2017/7/3 16:44 | 文件夹 | |
| AndroidManifest.xml | 2017/7/3 16:45 | XML 文档 | 1 KB |

After completing the project configuration of the above steps, you can read and display your own color, depth, and IR images in your newly created project by imitating the code in the Demos directory in the OpenNI_Android_32bit_2.3 package.

8.2 Development Instructions

For the Android platform SDK, we provide two packages, OpenNI_Android_32bit_2.3 and OpenNI_Android_Native_2.3,

OpenNI_Android_32bit_2.3 is the Java layer SDK, including the following:

Demos: The Android platform uses the Java layer API to view the source code of the color, depth, and IR diagram routines;

Apks: compiled APK file for viewing color, depth, and IR map;

Asserts: OpenNI2 related configuration files;

Libs: jar package and so library of Android platform OpenNI2;

OpenNI_Android_Native_2.3 is the Native layer SDK, including the following:

Demos: The Android platform uses the Native layer API to view the source code of the color, depth, and IR diagram routines;

Apks: compiled APK file for viewing color, depth, and IR map;

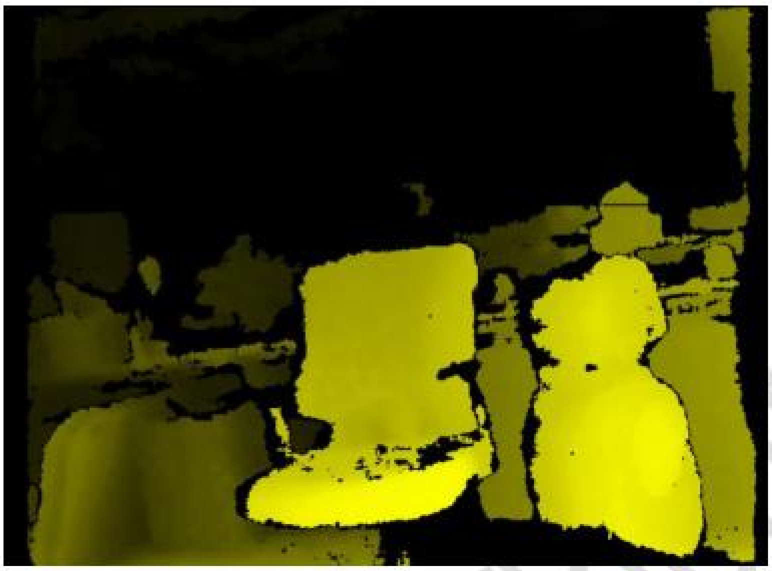
Asserts: OpenNI2 related configuration files;

Libs: the so library of OpenNI2 on the Android platform;

The important routine depthforopenni2 in the Java layer SDK is explained below. The main function of this example is to read the depth of the camera.

The figure also shows that the camera is connected and the program is opened and the normal

operation is shown in the figure below. Please note that in the current version of the OpenNI2 SDK,
You need to make sure that the Android system has root permissions and SeLinux is set to Permissive mode to obtain the system USB permissions normally, and get the corresponding color, depth and IR data:



For the specific implementation source code, please refer to the project depthforopenni2 in the Demos directory and compile it with Android Studio.
When running related programs of the Android platform, plug in the camera and open the APK program, a USB authorization pop-up window will appear. If there is no authorization pop-up window and the USB permission is not obtained, the relevant data cannot be read. At this time, you need to check the system USB Authorize related content.