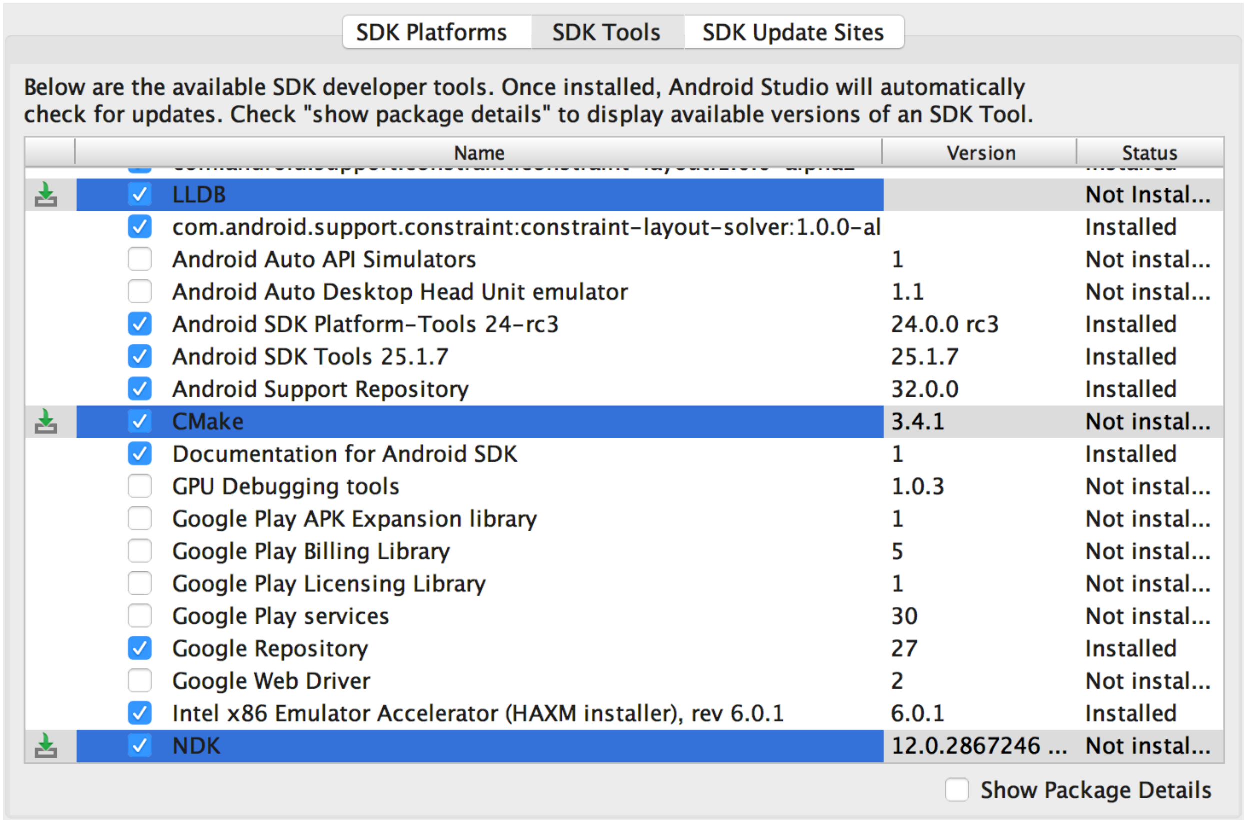
Android NDK + Open CV

– 製作者：王韋翔 (Wei-Xiang Wang)
– Email：40343108@gm.nfu.edu.tw

在 Android Studio 安裝 NDK

1. 開啟專案，在菜單列選擇 Tools -> Android -> SDK Manager 。
2. 點選 SDK Tools 分頁。
3. 如下圖所示，將LLDB、Cmake、NDK勾選。
4. 點擊應用(*Apply*)，然後點擊OK進行安裝。



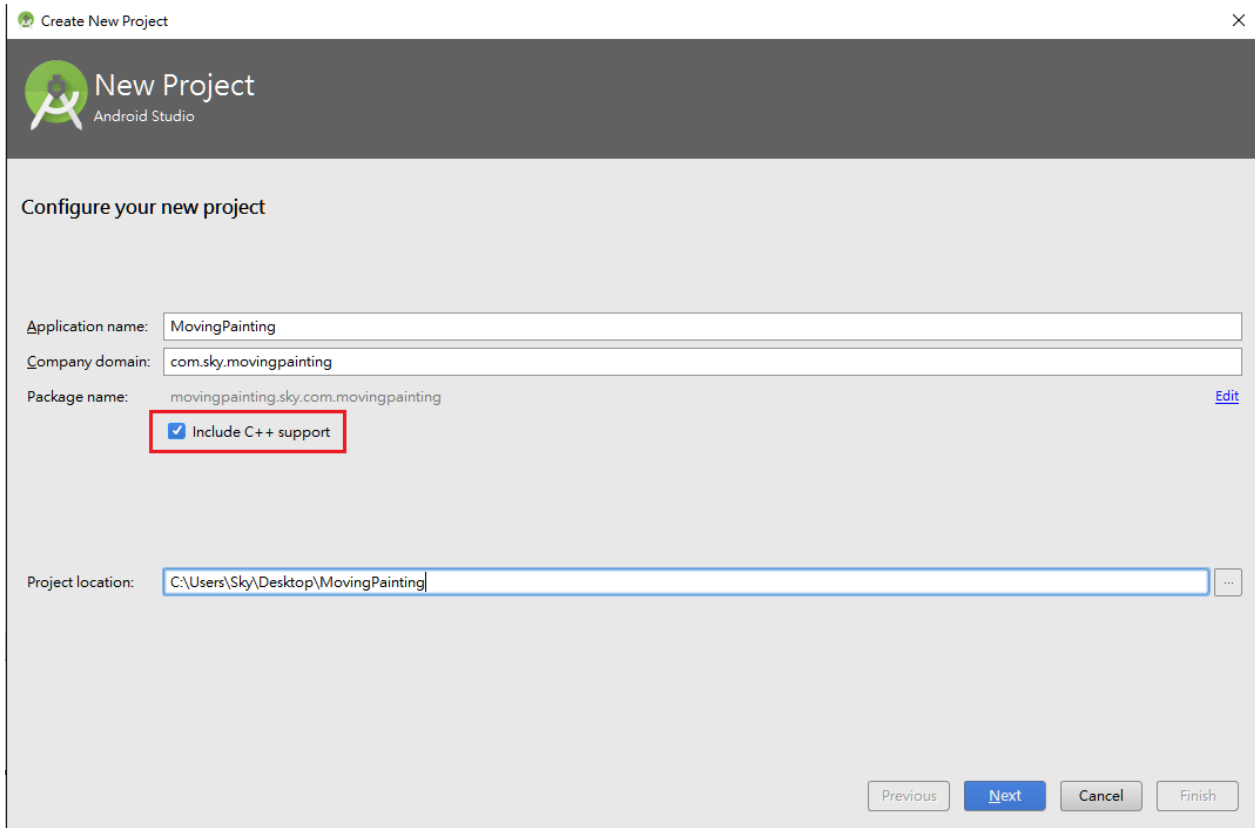
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

		Name	Version	Status
	✓	LLDB		Not Instal...
	✓	com.android.support.constraint:constraint-layout-solver:1.0.0-al		Installed
		Android Auto API Simulators	1	Not instal...
		Android Auto Desktop Head Unit emulator	1.1	Not instal...
	✓	Android SDK Platform-Tools 24-rc3	24.0.0 rc3	Installed
	✓	Android SDK Tools 25.1.7	25.1.7	Installed
	✓	Android Support Repository	32.0.0	Installed
	✓	CMake	3.4.1	Not instal...
	✓	Documentation for Android SDK	1	Installed
		GPU Debugging tools	1.0.3	Not instal...
		Google Play APK Expansion library	1	Not instal...
		Google Play Billing Library	5	Not instal...
		Google Play Licensing Library	1	Not instal...
		Google Play services	30	Not instal...
	✓	Google Repository	27	Installed
		Google Web Driver	2	Not instal...
	✓	Intel x86 Emulator Accelerator (HAXM installer), rev 6.0.1	6.0.1	Installed
	✓	NDK	12.0.2867246 ...	Not instal...

Show Package Details

在 Android Studio 使用 NDK

1. 點選 File -> New -> New Project
2. 要使用 Android Studio 的 C++ 功能，需勾選 Include C++ support。
3. 點選 Next 進行下一步



Configure your new project

Application name: MovingPainting

Company domain: com.sky.movingpainting

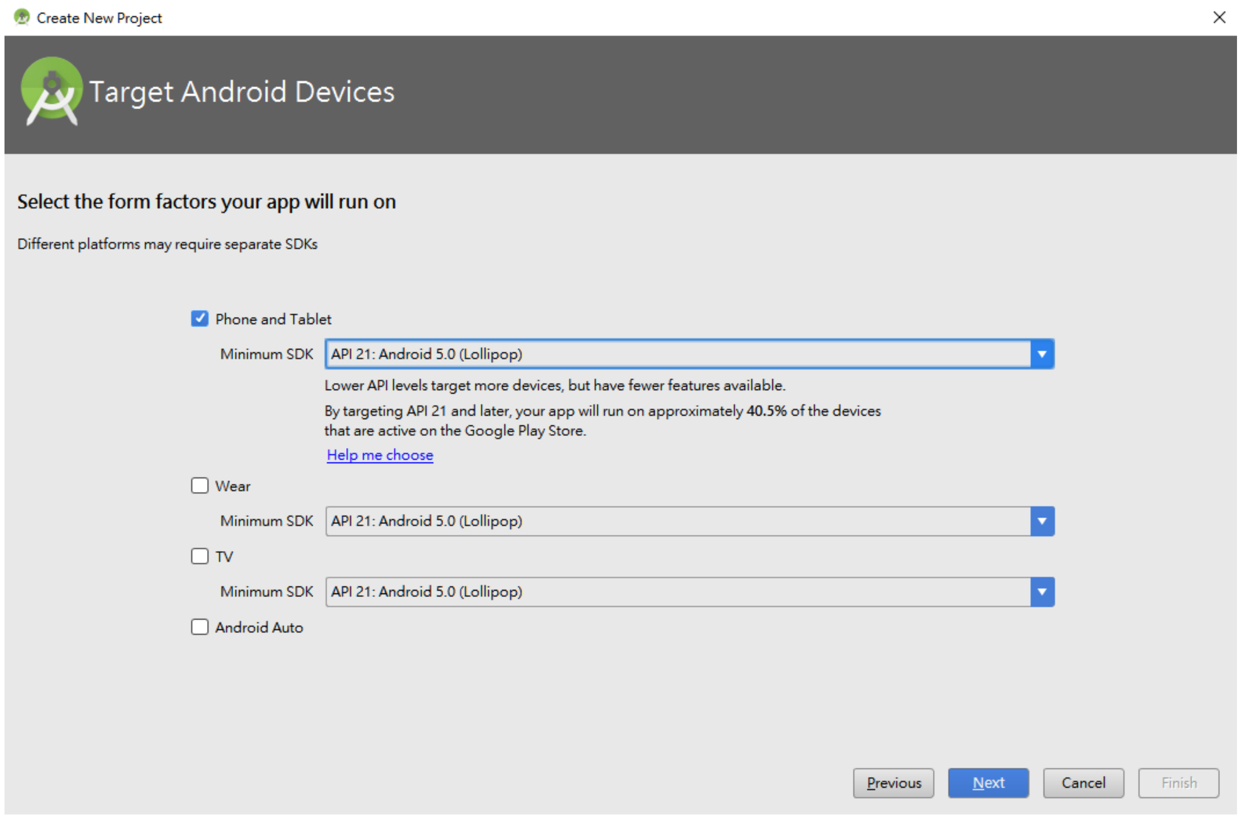
Package name: movingpainting.sky.com.movingpainting

☑ Include C++ support

Project location: C:\Users\Sky\Desktop\MovingPainting

在 Android Studio 使用 NDK

4. 勾選 Phone and Tablet
5. 依照需求選擇最低SDK版本
6. 點選 Next 進行下一步



Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet
Minimum SDK: API 21: Android 5.0 (Lollipop)
Lower API levels target more devices, but have fewer features available.
By targeting API 21 and later, your app will run on approximately **40.5%** of the devices that are active on the Google Play Store.
Help me choose

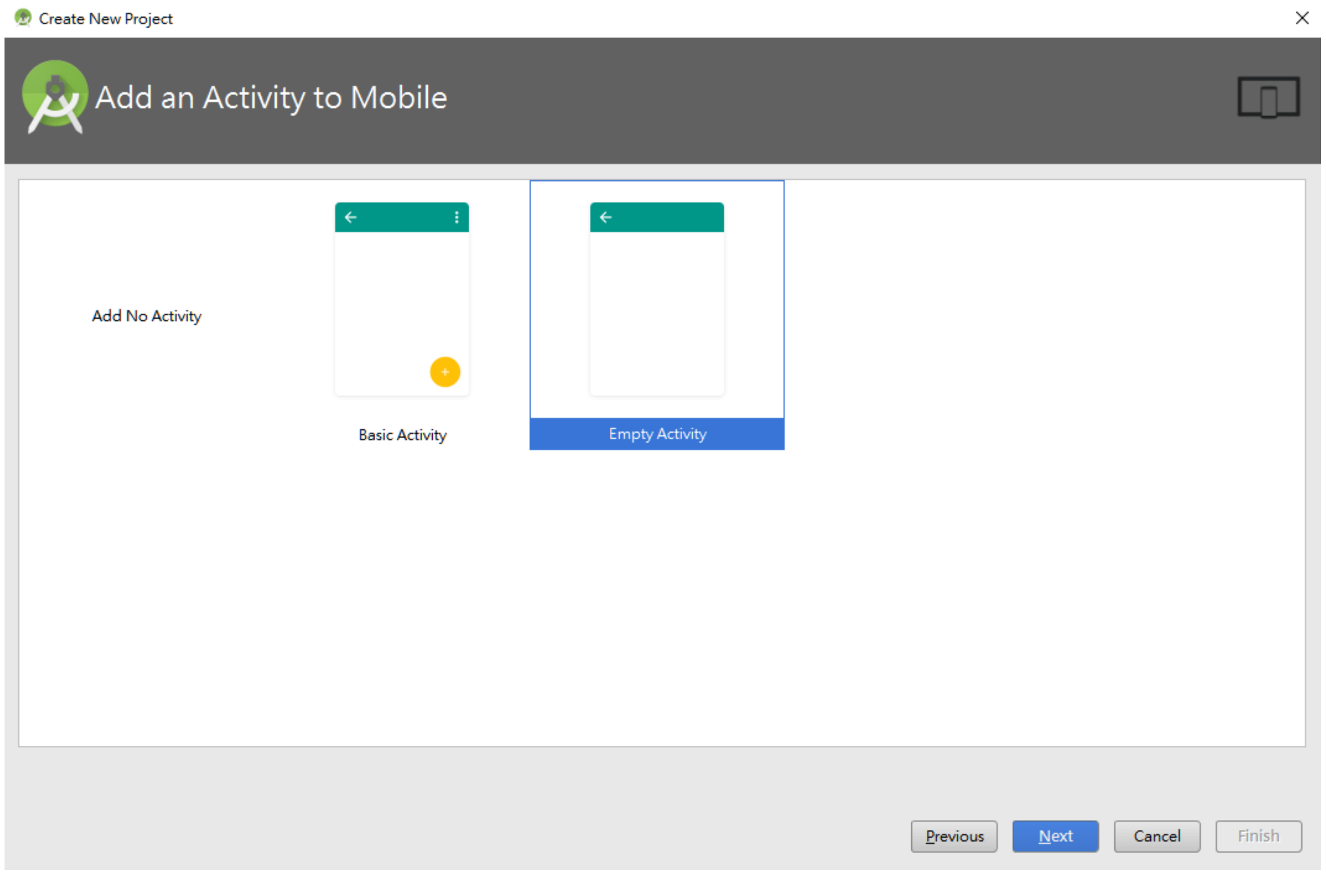
[ ] Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Android Auto

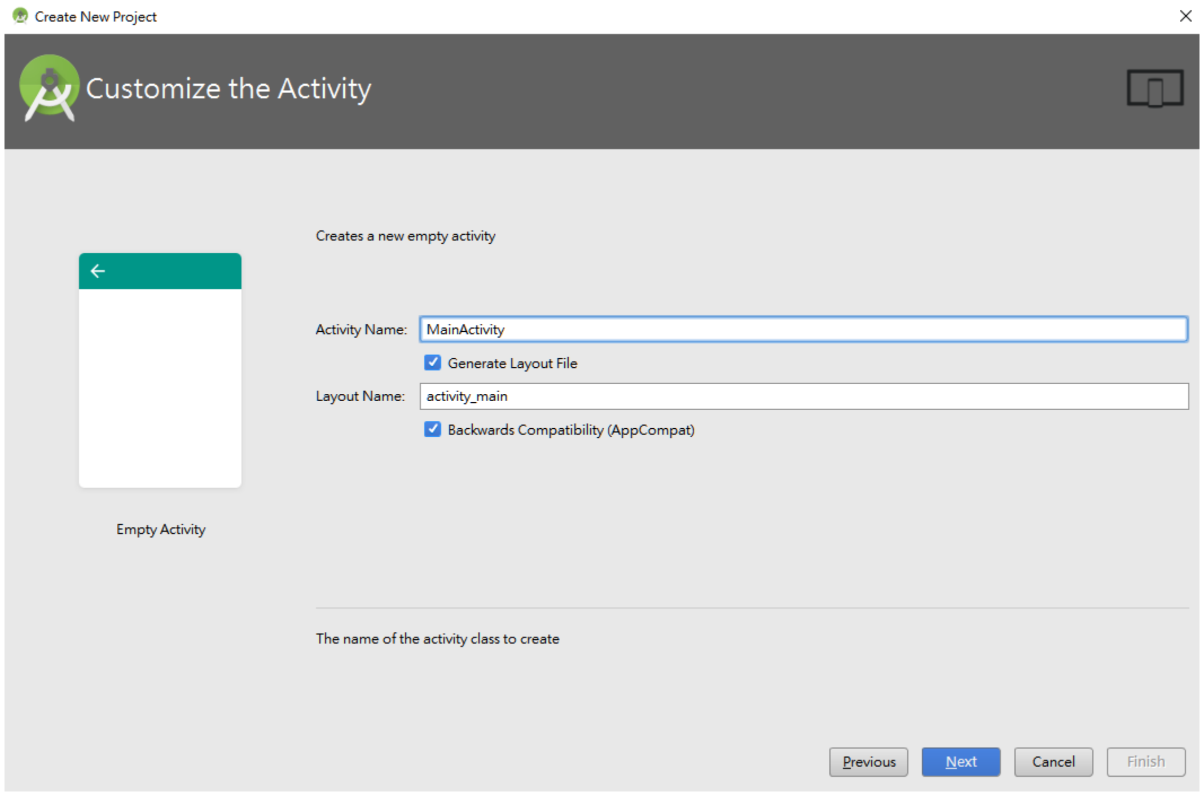
在 Android Studio 使用 NDK

7. 選擇你要的開發類型 *(注意：使用 C++ Support 會只有這兩個專案類型可選)*
8. 點選 Next 進行下一步



在 Android Studio 使用 NDK

9. 輸入剛開始的類別名稱(*Activity Name*)和佈局檔案名稱(*Layout Name*)
10. 勾選生成佈局檔案(*Generate Layout File*)和維持相容性(*Backward Compatibility*)
11. 點選 Next 進行下一步



Creates a new empty activity

Activity Name: MainActivity

☑ Generate Layout File

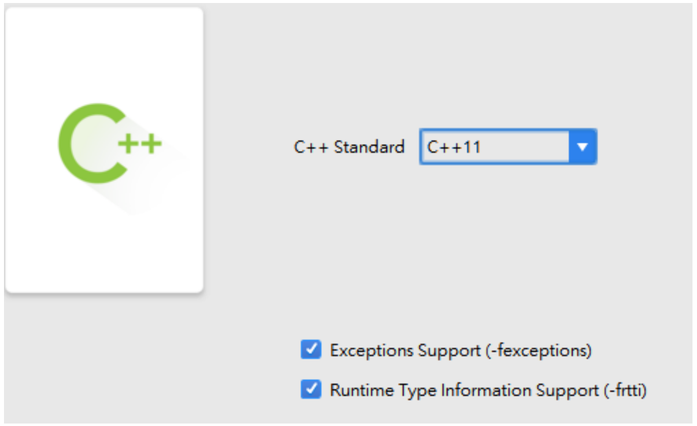
Layout Name: activity_main

☑ Backwards Compatibility (AppCompat)

The name of the activity class to create

在 Android Studio 使用 NDK

12. 在這裡，將 C++ Standard 改為 C++11，並勾選 Exceptions Support 和 Runtime Type Information Support。
13. 點選 Finish 完成



C++ Standard: C++11

✓ Exceptions Support (-fexceptions)

✓ Runtime Type Information Support (-frtti)

選項說明：

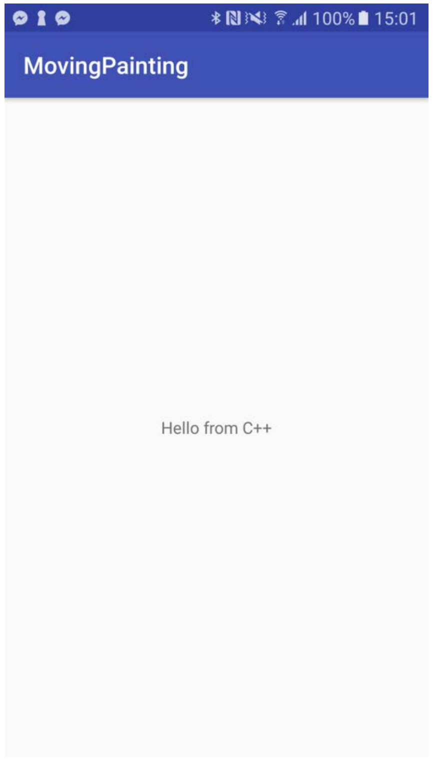
➢ C++ Standard：點擊下拉框，可以選擇標準 C++，或者選擇默認 CMake 設置的 Toolchain Default 選項。

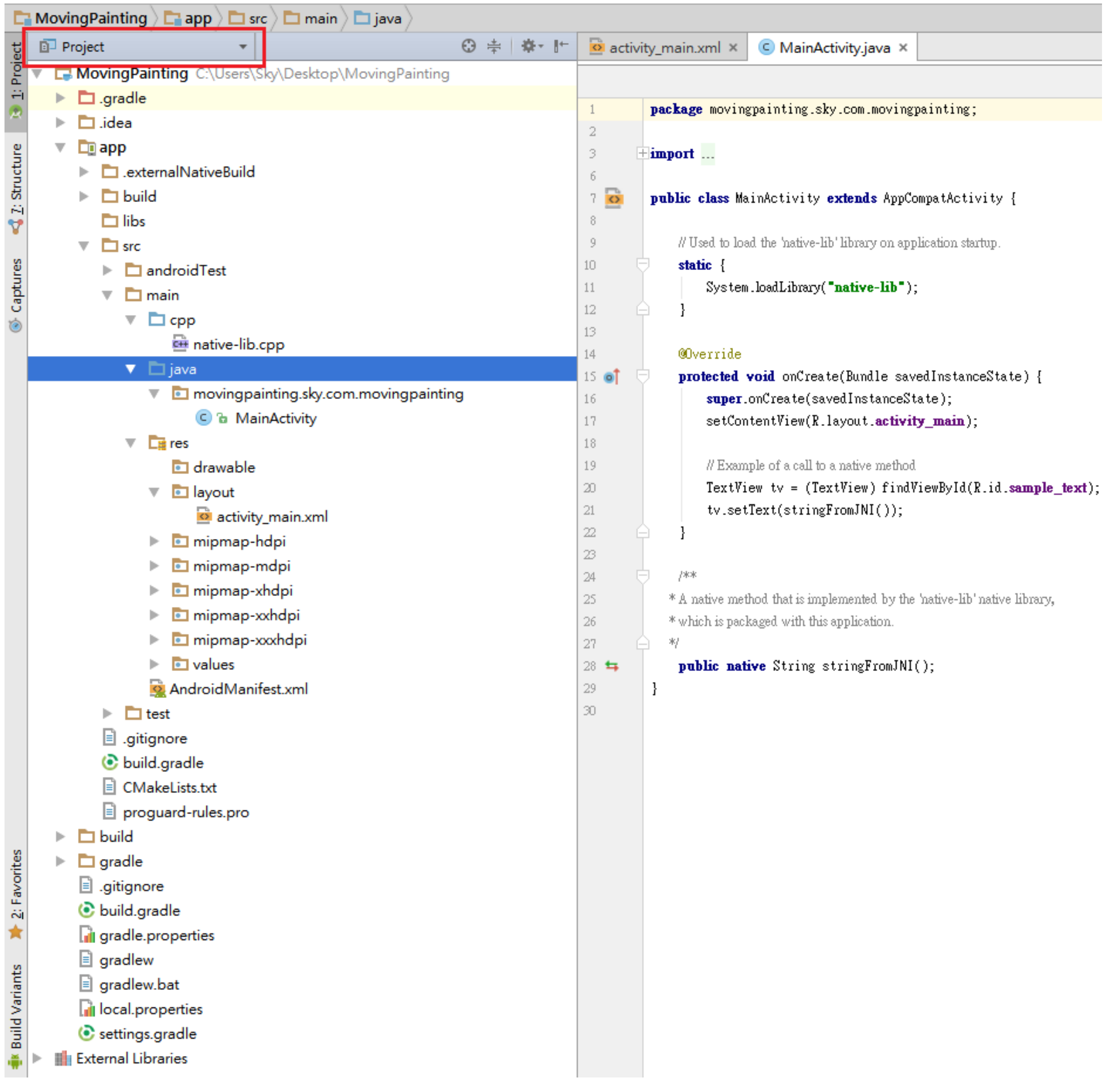
➢ Exceptions Support：如果你想使用有關 C++ 異常處理的支持，就勾選它。勾選之後，Android Studio 會在 module 層的 build.gradle 文件中的 cppFlags 中添加 -fexcetions 標誌。

➢ Runtime Type Information Support：如果你想支持 RTTI，那麼就勾選它。勾選之後，Android Studio 會在 module 層的 build.gradle 文件中的 cppFlags 中添加 -frtti 標誌。

在 Android Studio 使用 NDK

14. 在左上角切換至 Project 模式。
15. 檢查檔案結構是否完整。
16. 點擊 Run -> Run ‘app’ 進行編譯，應該會看到如下圖的執行結果。





```
MovingPainting  C:\Users\Sky\Desktop\MovingPainting
  .gradle
  .idea
  app
    .externalNativeBuild
    build
    libs
    src
      androidTest
      main
        cpp
          native-lib.cpp
        java
          movingpainting.sky.com.movingpainting
            MainActivity
        res
          drawable
          layout
            activity_main.xml
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
        AndroidManifest.xml
      test
    .gitignore
    build.gradle
    CMakeLists.txt
    proguard-rules.pro
  build
  gradle
  .gitignore
  build.gradle
  gradle.properties
  gradlew
  gradlew.bat
  local.properties
  settings.gradle
External Libraries
```

```java
package movingpainting.sky.com.movingpainting;

import ...

public class MainActivity extends AppCompatActivity {

    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary("native-lib");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.sample_text);
        tv.setText(stringFromJNI());
    }

    /**
     * A native method that is implemented by the 'native-lib' native library,
     * which is packaged with this application.
     */
    public native String stringFromJNI();
}
```

在 Android Studio 使用 NDK

17. 點擊並開啟 native-lib.cpp 檔案，位置如圖1。
18. 使用 NDK 的 C/C++ 程式碼必須引入jni.h 標準函式庫，而 NDK 本身的資料型態也跟 Java 略為不同，像是String 變為 jstring，boolean 變為 jboolean 等…。
19. 函式定義方法為：

JNIEXPORT 回傳型態 JNICALL Java_資源包名稱_類別名稱_欲定義函式名稱(JNIEnv *env, jobject) { }

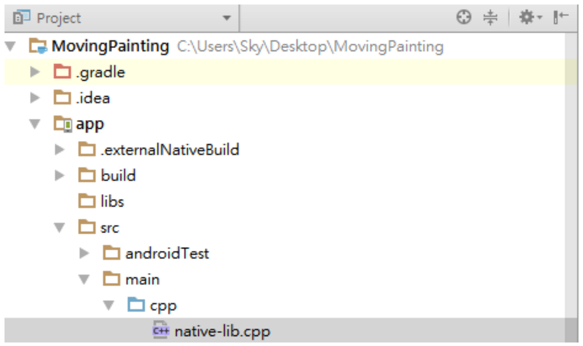
回傳型態：必須為 NDK 定義的資料型態

資源包名稱：如圖3的package，但句點改為底線

類別名稱：欲呼叫這個函式的類別名稱

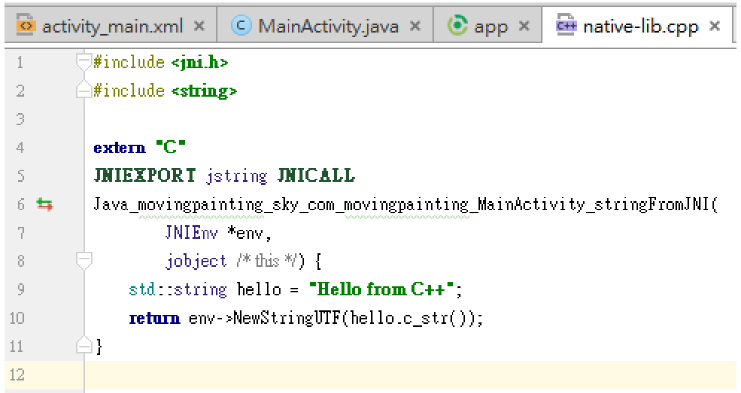
欲定義函式名稱：自行命名的函式名稱

函式參數：必須至少包含JNIEnv *env 和 jobject，若有需要則可增加參數，而呼叫端不需傳入JNIEnv *env 和 JNIEnv *env。也就是若無增加參數則直接 stringFromJNI() 即可呼叫。



```
Project
▼ MovingPainting C:\Users\Sky\Desktop\MovingPainting
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▶ .externalNativeBuild
    ▶ build
      libs
    ▼ src
      ▶ androidTest
      ▼ main
        ▼ cpp
            native-lib.cpp
```

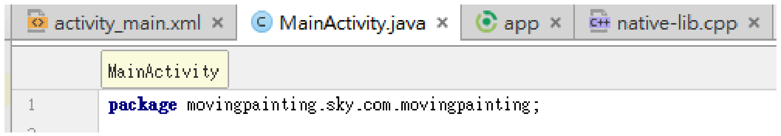
圖 1 檔案結構圖



activity_main.xml × | MainActivity.java × | app × | native-lib.cpp ×

```cpp
#include <jni.h>
#include <string>

extern "C"
JNIEXPORT jstring JNICALL
Java_movingpainting_sky_com_movingpainting_MainActivity_stringFromJNI(
        JNIEnv *env,
        jobject /* this */) {
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

圖 2 native-lib 的 C++ 程式碼



activity_main.xml × | MainActivity.java × | app × | native-lib.cpp ×

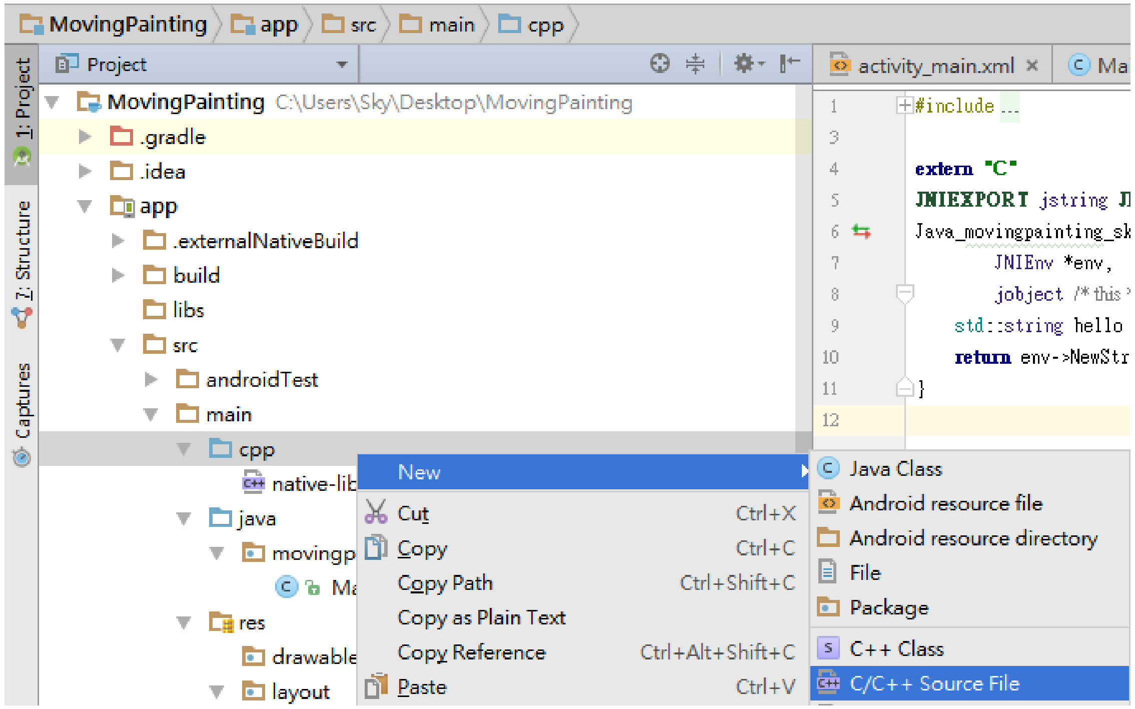
MainActivity

```java
package movingpainting.sky.com.movingpainting;
```

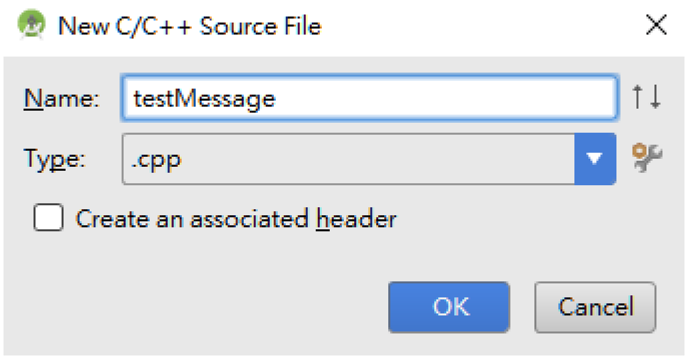
圖 3 資源包名稱示意圖

如何使用 NDK 增加程式(.cpp)關聯

1. 如下圖所示點擊 cpp資料夾 右鍵 -> New -> C/C++ Source File 創建新檔案。
2. 在 Name 輸入 testMessage。
3. 點擊 OK 完成創建。



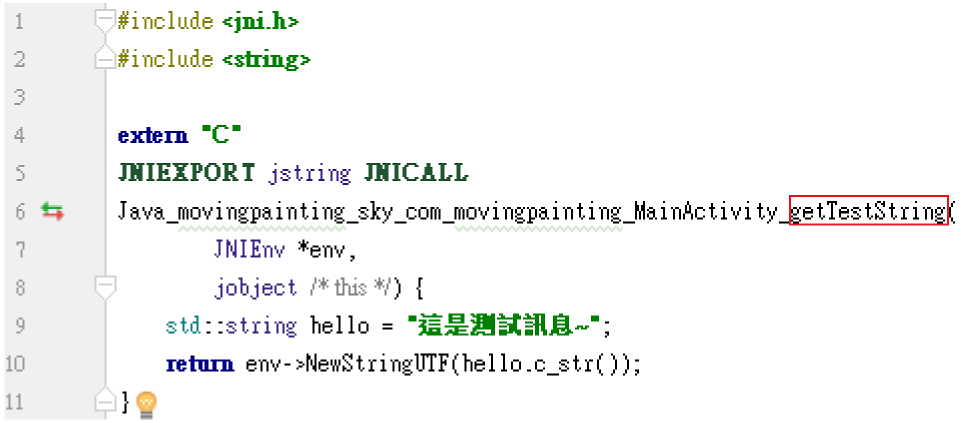
```
#include ...

extern "C"
JNIEXPORT jstring JI
Java_movingpainting_sk
        JNIEnv *env,
        jobject /* this *
    std::string hello
    return env->NewStr
}
```



Name: testMessage
Type: .cpp
☐ Create an associated header

如何使用 NDK 增加程式(.cpp)關聯

4. 開啟 native-lib.cpp 的程式碼並複製貼至 testMessage.cpp ，再將函數名稱更改為 getTestString。
5. 將 std::string hello = “Hello from C++”; 改為 std::string hello = "這是測試訊息~";
6. 點選儲存檔案(Crtl+S)



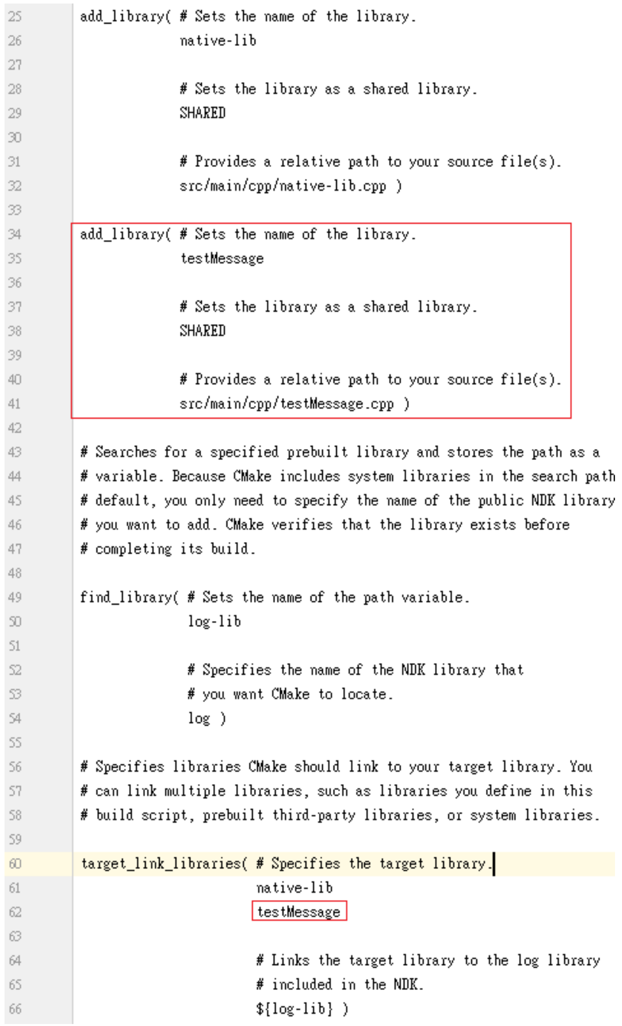
```
#include <jni.h>
#include <string>

extern "C"
JNIEXPORT jstring JNICALL
Java_movingpainting_sky_com_movingpainting_MainActivity_getTestString(
        JNIEnv *env,
        jobject /* this */) {
    std::string hello = "這是測試訊息~";
    return env->NewStringUTF(hello.c_str());
}
```

如何使用 NDK 增加程式(.cpp)關聯

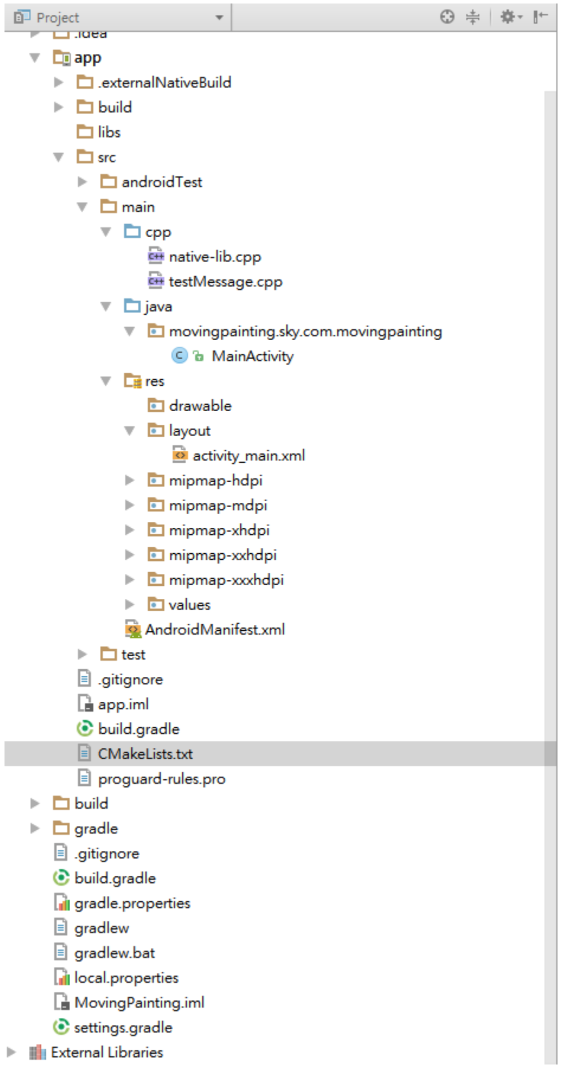
7. 開啟 CMakeLists.txt，並增加下方圖示的紅色圈選處。

※ 圖中的 testMessage 為 Add_library 所連結的關聯函式庫名稱。就是當 Java 要讀取這個 C++ 程式時所要載入的函式庫名稱 src/main/cpp/testMessage.cpp 就是對應的 C++ 程式所在路徑

※ 下方 target_link_libraries 需增加新的 library 函式庫名稱。



```
add_library( # Sets the name of the library.
             native-lib

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
             src/main/cpp/native-lib.cpp )

add_library( # Sets the name of the library.
             testMessage

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
             src/main/cpp/testMessage.cpp )

# Searches for a specified prebuilt library and stores the path as a
# variable. Because CMake includes system libraries in the search path
# default, you only need to specify the name of the public NDK library
# you want to add. CMake verifies that the library exists before
# completing its build.

find_library( # Sets the name of the path variable.
              log-lib

              # Specifies the name of the NDK library that
              # you want CMake to locate.
              log )

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

target_link_libraries( # Specifies the target library.
                       native-lib
                       testMessage

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



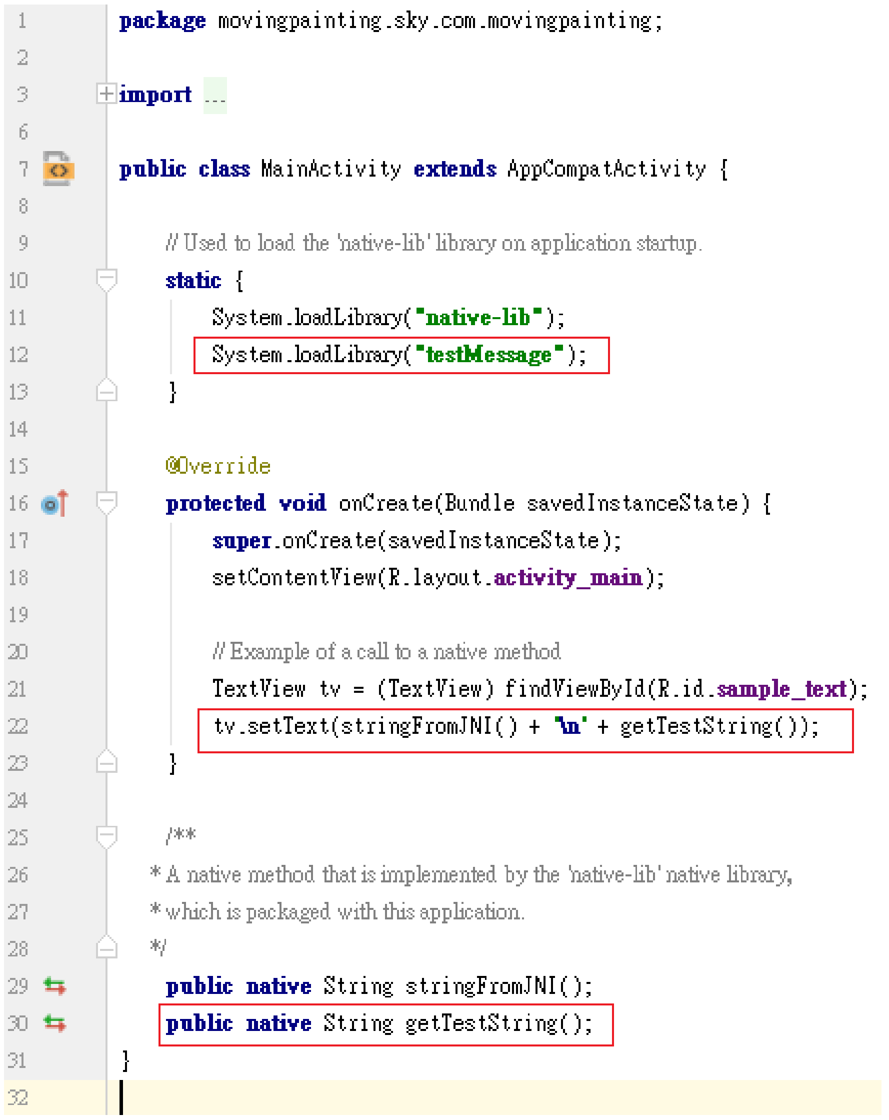
```
Project
  .idea
  app
    .externalNativeBuild
    build
    libs
    src
      androidTest
      main
        cpp
          native-lib.cpp
          testMessage.cpp
        java
          movingpainting.sky.com.movingpainting
            MainActivity
        res
          drawable
          layout
            activity_main.xml
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
        AndroidManifest.xml
      test
    .gitignore
    app.iml
    build.gradle
    CMakeLists.txt
    proguard-rules.pro
  build
  gradle
  .gitignore
  build.gradle
  gradle.properties
  gradlew
  gradlew.bat
  local.properties
  MovingPainting.iml
  settings.gradle
External Libraries
```

如何使用 NDK 增加程式(.cpp)關聯

8. 開啟 MainActivity.java 程式碼，並新增紅色圈選處的程式碼。

※ 其中 System.loadLibrary("testMessage"); 的 testMessage 就是上述 CMakeLists.txt 所使用的 Add_library 所定義之函式庫名稱。

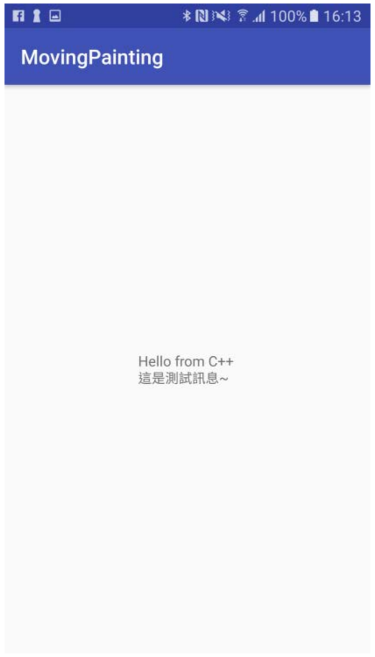
※ 宣告原型函式時，需在回傳型態前面加上 native 像是：public native String getTestString();



```java
package movingpainting.sky.com.movingpainting;

import ...

public class MainActivity extends AppCompatActivity {

    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary("native-lib");
        System.loadLibrary("testMessage");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.sample_text);
        tv.setText(stringFromJNI() + '\n' + getTestString());
    }

    /**
   * A native method that is implemented by the 'native-lib' native library,
   * which is packaged with this application.
   */
    public native String stringFromJNI();
    public native String getTestString();
}
```

如何使用 NDK 增加程式(.cpp)關聯

9. 點選 圖4 紅色圈選處的按鈕，進行同步專案建置檔案。
10. 這時右下角會出現進度條(如圖5)，表示正在同步專案建置。
11. 等待建置完成後，點擊 Run -> Run ‘app’ 進行編譯，應該會看到如右圖的執行結果。


MovingPainting

Hello from C++
這是測試訊息~

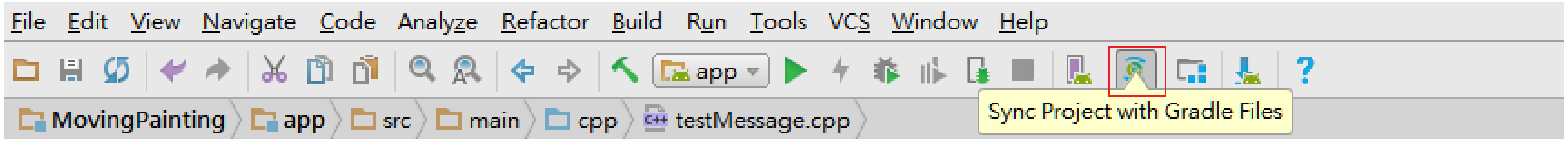


圖 4 同步專案按鈕

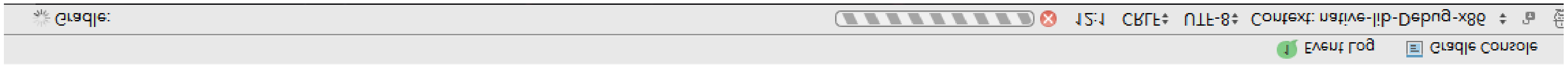
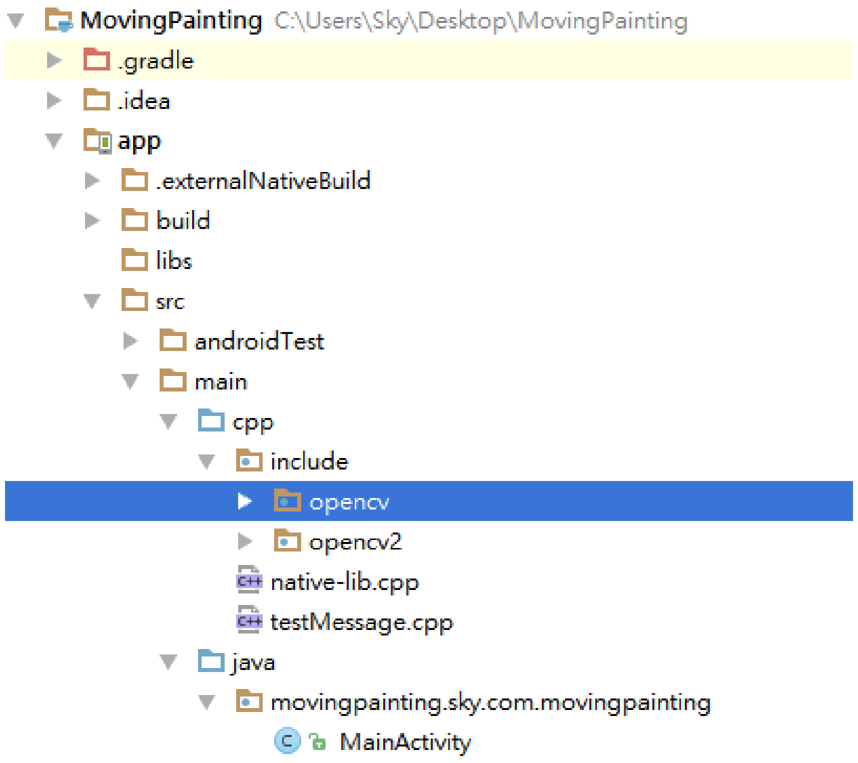


圖 5 重新建置示意圖

如何使用 NDK 導入 OpenCV

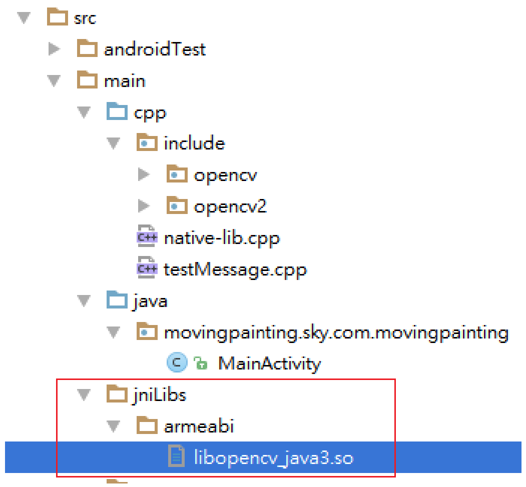
1. 前往 OpenCV 官網下載 OpenCV for Android (筆者安裝的是3.1版)
2. 安裝並解壓縮至任意位子。
3. 開啟安裝後的資料夾位子，並至 \..\OpenCV-android-sdk-3.1\sdk\native\jni\include 資料夾位子，將 include 整個資料夾複製至專案的 cpp 資料夾內，如下圖。



```
▼ MovingPainting C:\Users\Sky\Desktop\MovingPainting
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▶ .externalNativeBuild
    ▶ build
      libs
    ▼ src
      ▶ androidTest
      ▼ main
        ▼ cpp
          ▼ include
            ▶ opencv
            ▶ opencv2
            native-lib.cpp
            testMessage.cpp
        ▼ java
          ▼ movingpainting.sky.com.movingpainting
              MainActivity
```

如何使用 NDK 導入 OpenCV

4. 選擇 main 資料夾，點擊右鍵選擇 New -> Directory 創建資料夾，並命名為 jniLibs。
5. 於 jniLibs 內，再依照自己的開發版本架構創建資料夾名稱。
6. 至 \..\OpenCV-android-sdk-3.1\sdk\native\libs 位子，找到對應的開發版本架構資料夾，將 libopencv_java3.so 複製到專案。
7. 完成後的檔案結構應該如右圖所示。

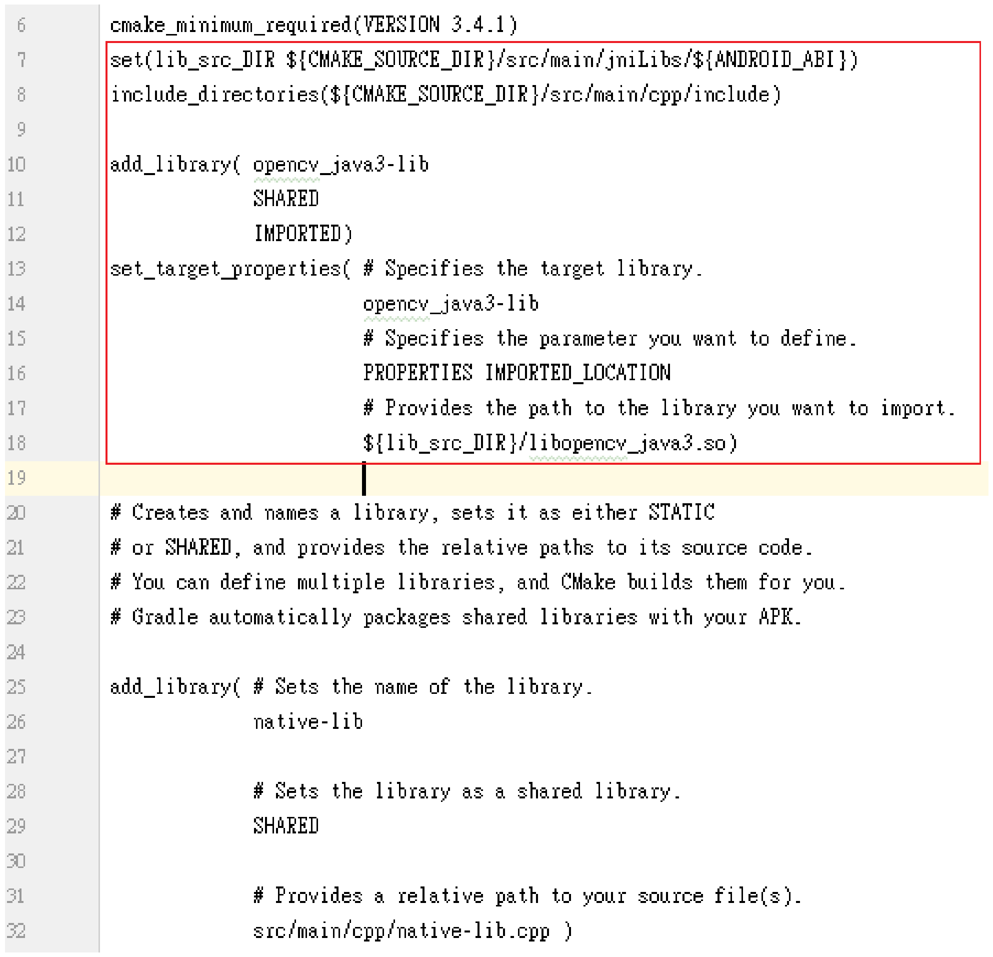


```
▼ src
  ▶ androidTest
  ▼ main
    ▼ cpp
      ▼ include
        ▶ opencv
        ▶ opencv2
        native-lib.cpp
        testMessage.cpp
    ▼ java
      ▼ movingpainting.sky.com.movingpainting
          MainActivity
    ▼ jniLibs
      ▼ armeabi
          libopencv_java3.so
```

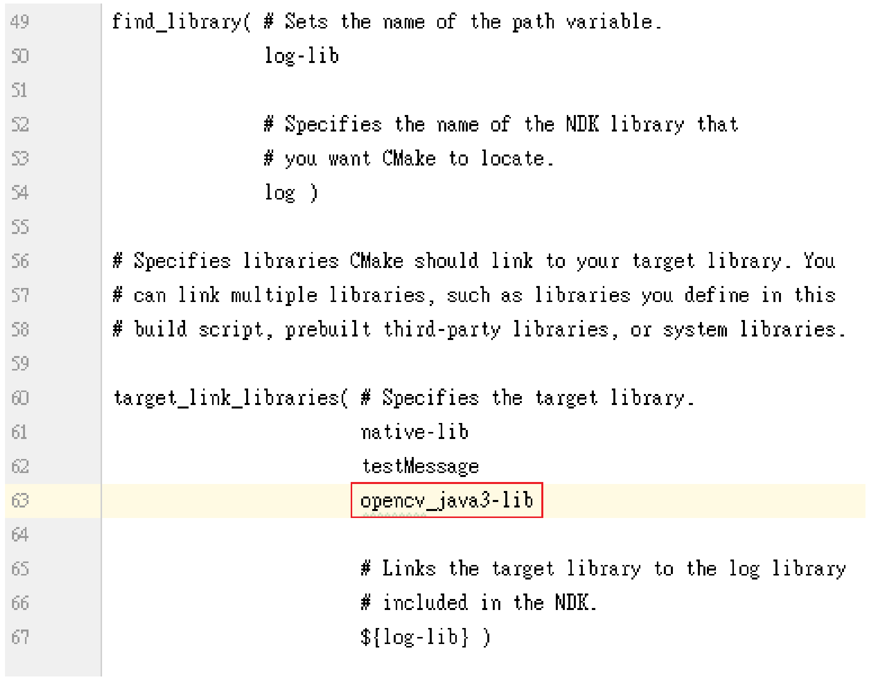
※ 開發版本架構 (armeabi, armv7a-neon, arm7a-neon-android8, mips, x86)
armeabi：ARM v5 架構 和 ARM v6 架構 於 Android API 8 以上,
armv7a-neon：NEON-optimized ARM v7 架構 於Android API 9 以上,
arm7a-neon-android8：NEON-optimized ARM v7 架構 於 Android API 8,
mips：MIPS 架構於 Android API 9 以上,
x86：Intel x86 CPUs 架構 於 Android API 9 以上.

如何使用 NDK 導入 OpenCV

8. 開啟 CMakeLists.txt，並增加下方圖示的紅色圈選處。



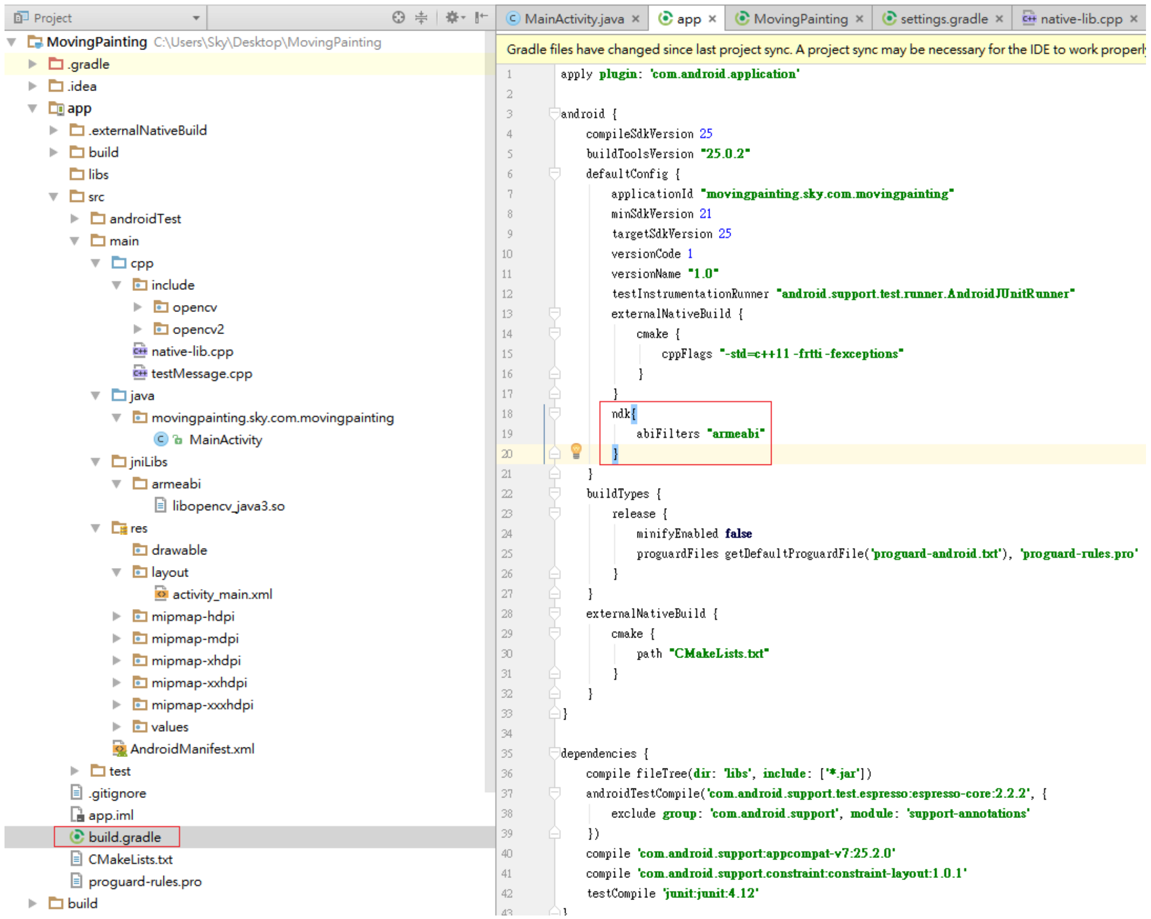
```
cmake_minimum_required(VERSION 3.4.1)
set(lib_src_DIR ${CMAKE_SOURCE_DIR}/src/main/jniLibs/${ANDROID_ABI})
include_directories(${CMAKE_SOURCE_DIR}/src/main/cpp/include)

add_library( opencv_java3-lib
             SHARED
             IMPORTED)
set_target_properties( # Specifies the target library.
                       opencv_java3-lib
                       # Specifies the parameter you want to define.
                       PROPERTIES IMPORTED_LOCATION
                       # Provides the path to the library you want to import.
                       ${lib_src_DIR}/libopencv_java3.so)

# Creates and names a library, sets it as either STATIC
# or SHARED, and provides the relative paths to its source code.
# You can define multiple libraries, and CMake builds them for you.
# Gradle automatically packages shared libraries with your APK.

add_library( # Sets the name of the library.
             native-lib

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
             src/main/cpp/native-lib.cpp )
```



```
find_library( # Sets the name of the path variable.
              log-lib

              # Specifies the name of the NDK library that
              # you want CMake to locate.
              log )

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

target_link_libraries( # Specifies the target library.
                       native-lib
                       testMessage
                       opencv_java3-lib

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

如何使用 NDK 導入 OpenCV

9. 開啟 app 內的 build.gradle，並在 defaultConfig 內增加紅色圈選處，armeabi 非固定，而是按照你的開發版本架構而定，也就是 jniLibs 內的那個資料夾。



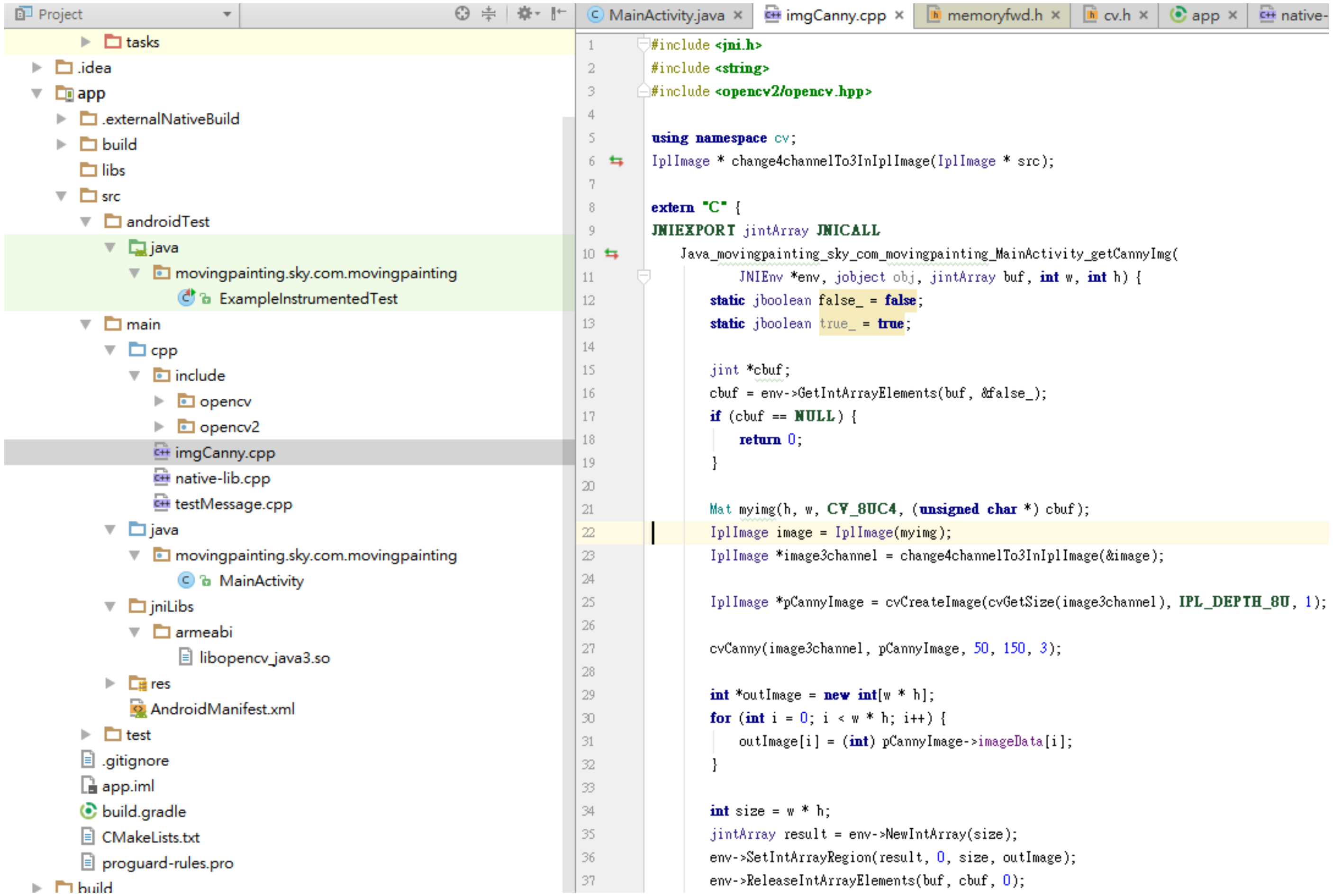
```
MovingPainting C:\Users\Sky\Desktop\MovingPainting
  .gradle
  .idea
  app
    .externalNativeBuild
    build
    libs
    src
      androidTest
      main
        cpp
          include
            opencv
            opencv2
          native-lib.cpp
          testMessage.cpp
        java
          movingpainting.sky.com.movingpainting
            MainActivity
        jniLibs
          armeabi
            libopencv_java3.so
        res
          drawable
          layout
            activity_main.xml
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
        AndroidManifest.xml
      test
    .gitignore
    app.iml
    build.gradle
    CMakeLists.txt
    proguard-rules.pro
  build
```

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properl

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

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "movingpainting.sky.com.movingpainting"
        minSdkVersion 21
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        externalNativeBuild {
            cmake {
                cppFlags "-std=c++11 -frtti -fexceptions"
            }
        }
        ndk{
            abiFilters "armeabi"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    externalNativeBuild {
        cmake {
            path "CMakeLists.txt"
        }
    }
}

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-v7:25.2.0'
    compile 'com.android.support.constraint:constraint-layout:1.0.1'
    testCompile 'junit:junit:4.12'
}
```

如何使用 NDK 導入 OpenCV

10. 依照 如何使用 NDK 增加程式(.cpp)關聯 的方式，增加 getCannyImg 函式，程式碼會附於最後幾頁，這時檔案結構應該如下圖所示，且程式碼沒有紅字錯誤。

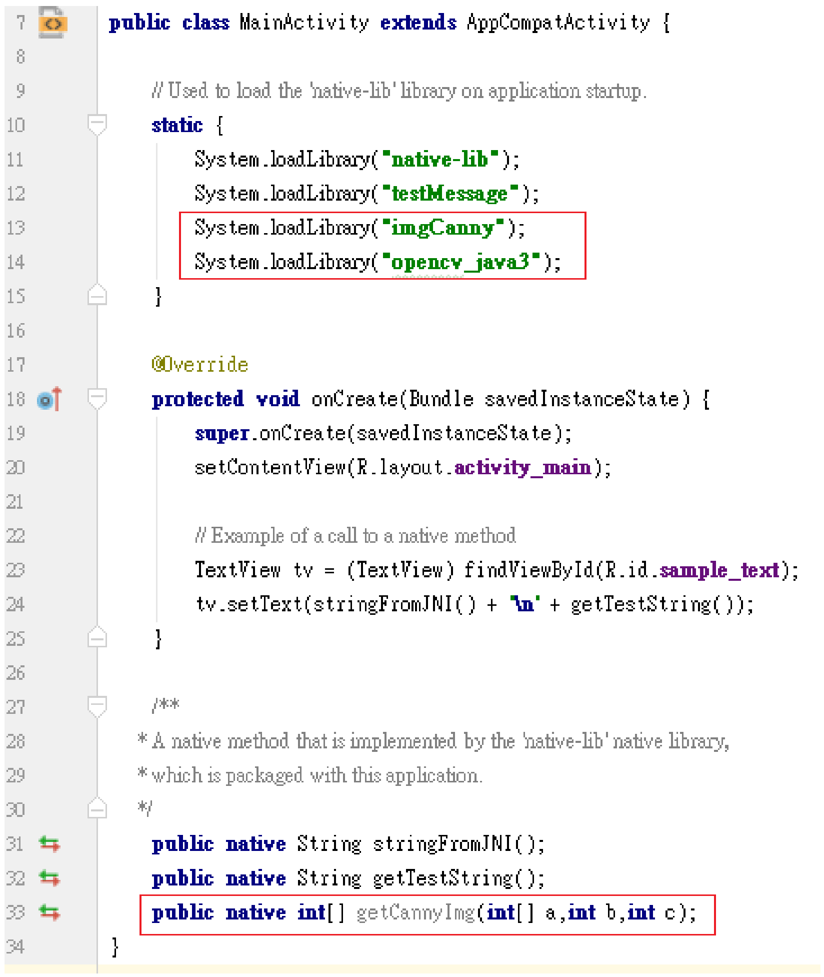


```cpp
#include <jni.h>
#include <string>
#include <opencv2/opencv.hpp>

using namespace cv;
IplImage * change4channelTo3InIplImage(IplImage * src);

extern "C" {
JNIEXPORT jintArray JNICALL
    Java_movingpainting_sky_com_movingpainting_MainActivity_getCannyImg(
        JNIEnv *env, jobject obj, jintArray buf, int w, int h) {
    static jboolean false_ = false;
    static jboolean true_ = true;

    jint *cbuf;
    cbuf = env->GetIntArrayElements(buf, &false_);
    if (cbuf == NULL) {
        return 0;
    }

    Mat myimg(h, w, CV_8UC4, (unsigned char *) cbuf);
    IplImage image = IplImage(myimg);
    IplImage *image3channel = change4channelTo3InIplImage(&image);

    IplImage *pCannyImage = cvCreateImage(cvGetSize(image3channel), IPL_DEPTH_8U, 1);

    cvCanny(image3channel, pCannyImage, 50, 150, 3);

    int *outImage = new int[w * h];
    for (int i = 0; i < w * h; i++) {
        outImage[i] = (int) pCannyImage->imageData[i];
    }

    int size = w * h;
    jintArray result = env->NewIntArray(size);
    env->SetIntArrayRegion(result, 0, size, outImage);
    env->ReleaseIntArrayElements(buf, cbuf, 0);
```

如何使用 NDK 導入 OpenCV

11. 開啟 MainActivity.java 程式碼，並新增紅色圈選處的程式碼。

System.loadLibrary("imgCanny");
System.loadLibrary("opencv_java3");
public native int[] getCannyImg(**int**[] a,**int** b,**int** c);

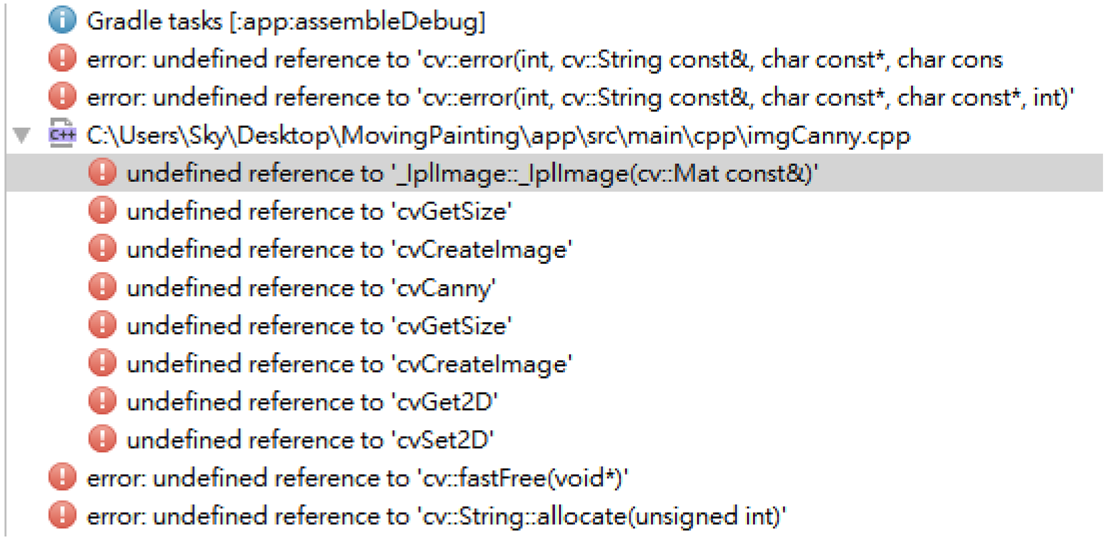


```java
public class MainActivity extends AppCompatActivity {

    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary("native-lib");
        System.loadLibrary("testMessage");
        System.loadLibrary("imgCanny");
        System.loadLibrary("opencv_java3");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.sample_text);
        tv.setText(stringFromJNI() + '\n' + getTestString());
    }

    /**
     * A native method that is implemented by the 'native-lib' native library,
     * which is packaged with this application.
     */
    public native String stringFromJNI();
    public native String getTestString();
    public native int[] getCannyImg(int[] a,int b,int c);
}
```

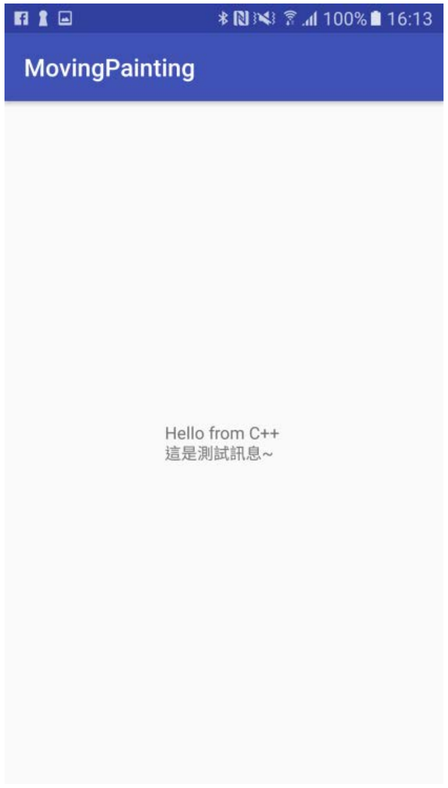
如何使用 NDK 導入 OpenCV

12. 點擊 Run -> Run ‘app’ 進行編譯，應該會看到編譯成功，且程式出現原先畫面 (如右方圖示)。

※ 若 Android Studio 的並無顯示紅字或錯誤，但執行 apk 編譯時出現如下圖所示的未定義錯誤，則代表 NDK 在編譯鏈結時期無法進行正確連結，若有此問題請參考下一頁。



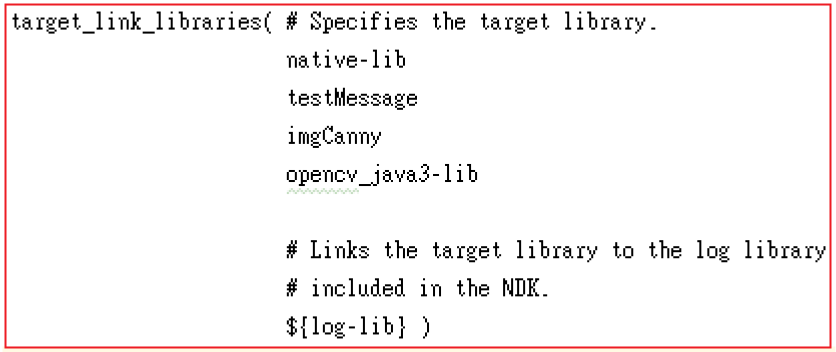
```
Gradle tasks [:app:assembleDebug]
error: undefined reference to 'cv::error(int, cv::String const&, char const*, char cons
error: undefined reference to 'cv::error(int, cv::String const&, char const*, char const*, int)'
C:\Users\Sky\Desktop\MovingPainting\app\src\main\cpp\imgCanny.cpp
    undefined reference to '_IplImage::_IplImage(cv::Mat const&)'
    undefined reference to 'cvGetSize'
    undefined reference to 'cvCreateImage'
    undefined reference to 'cvCanny'
    undefined reference to 'cvGetSize'
    undefined reference to 'cvCreateImage'
    undefined reference to 'cvGet2D'
    undefined reference to 'cvSet2D'
error: undefined reference to 'cv::fastFree(void*)'
error: undefined reference to 'cv::String::allocate(unsigned int)'
```



如何使用 NDK 導入 OpenCV

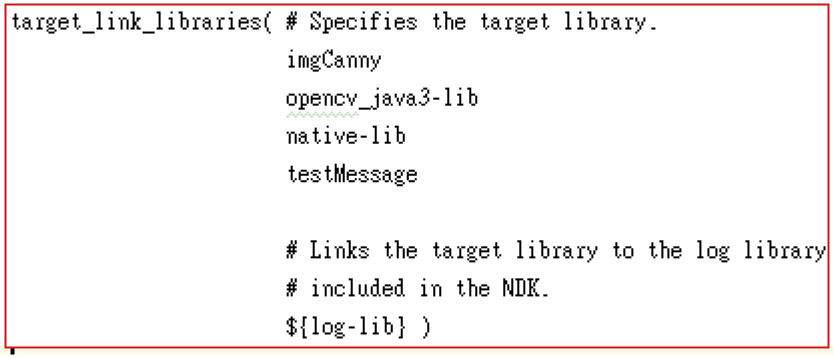
※ 鏈結錯誤的無定義參考解決方法：

1. 開啟 CMakeLists.txt，並至 target_link_libraries 處檢查。
假設 imgCanny 為主要 OpenCV 調用的 C++ 程式。
若寫法如 圖1 所示，則只有第一個 native-lib 能進行連結 log-lib 的 api。
因此才會造成 imgCanny 鏈結時期無法找到對應的定義參考函式。
2. 將鏈結函式庫的順序做更改，改變為如 圖2 所示。
3. 並重新點選同步專案按鈕 (*Sync Project with Gradle Files*)。
至於為什麼會有這種問題，筆者也不太清楚…
網路上似乎沒有太多人詳細說明這種問題。



```
target_link_libraries( # Specifies the target library.
                       native-lib
                       testMessage
                       imgCanny
                       opencv_java3-lib

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

圖 1 錯誤的鏈結順序

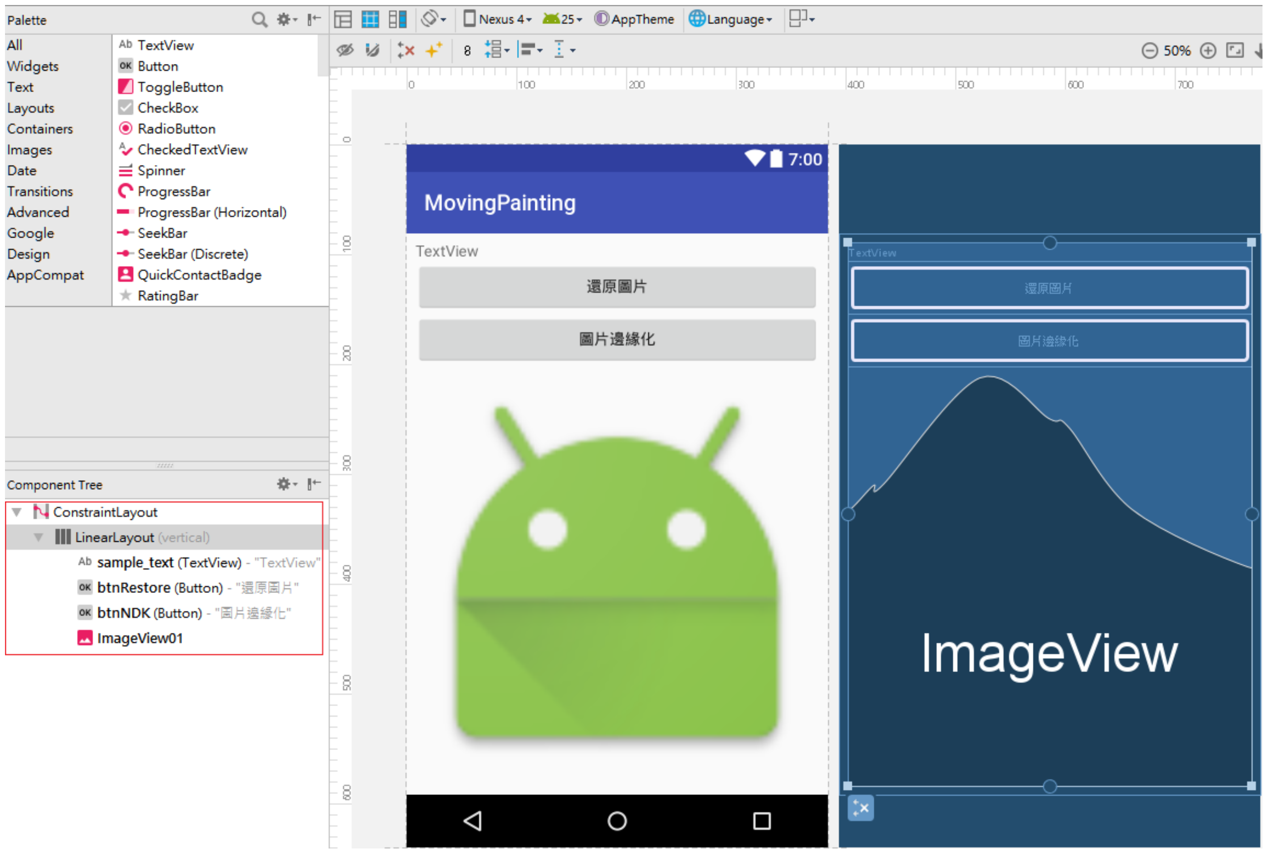


```
target_link_libraries( # Specifies the target library.
                       imgCanny
                       opencv_java3-lib
                       native-lib
                       testMessage

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

圖 2 正確的鏈結順序

如何使用 NDK 導入 OpenCV

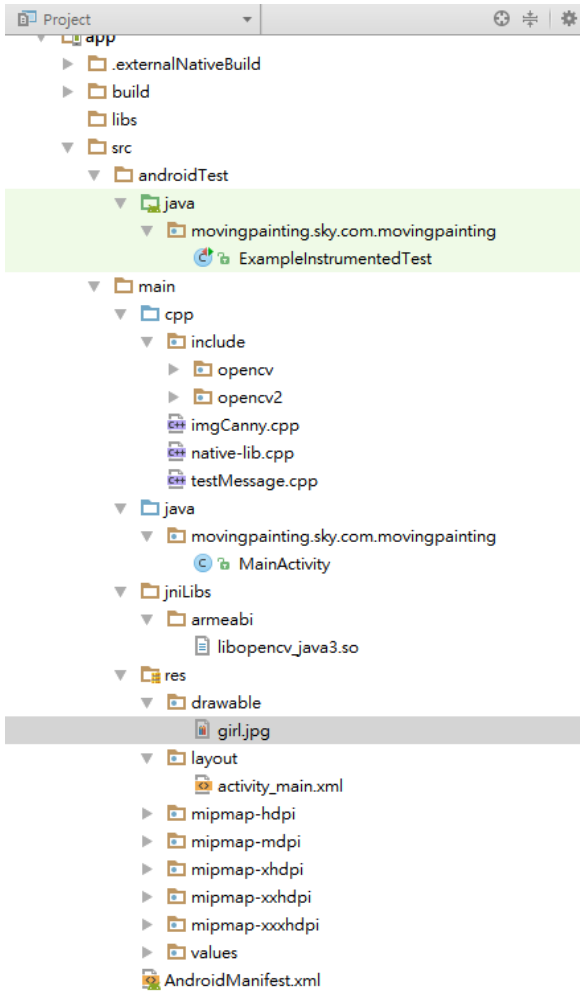
13. 開啟 src -> main -> res -> layout -> activity_main.xml 佈局檔案。
14. 新增如下圖所示的佈局，注意紅色框框內物件的配置屬性、名稱等。



```
ConstraintLayout
  LinearLayout (vertical)
    sample_text (TextView) - "TextView"
    btnRestore (Button) - "還原圖片"
    btnNDK (Button) - "圖片邊緣化"
    ImageView01
```

如何使用 NDK 導入 OpenCV

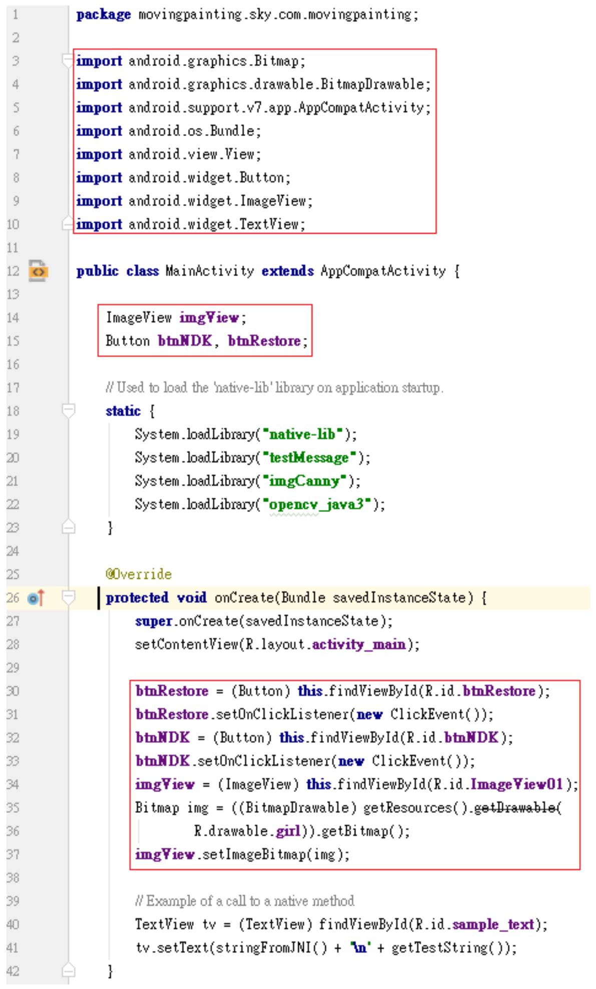
15. 將下列照片儲存命名 girl.jpg，並新增至 src -> main -> res -> drawable 資料夾內。



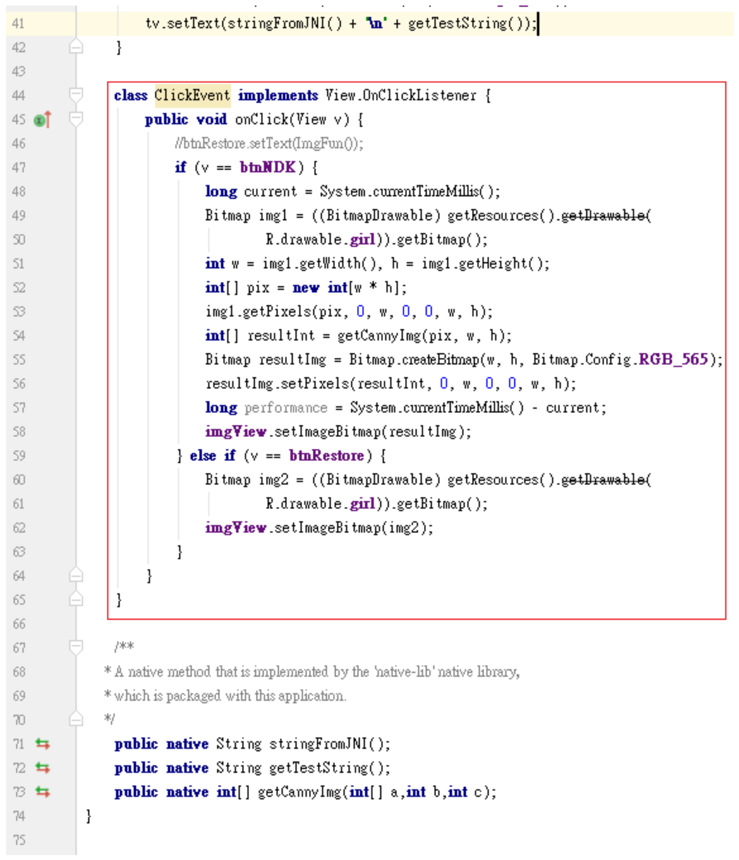
```
Project
▼ app
  ▶ .externalNativeBuild
  ▶ build
    libs
  ▼ src
    ▼ androidTest
      ▼ java
        ▼ movingpainting.sky.com.movingpainting
            ExampleInstrumentedTest
    ▼ main
      ▼ cpp
        ▼ include
          ▶ opencv
          ▶ opencv2
          imgCanny.cpp
          native-lib.cpp
          testMessage.cpp
      ▼ java
        ▼ movingpainting.sky.com.movingpainting
            MainActivity
      ▼ jniLibs
        ▼ armeabi
            libopencv_java3.so
      ▼ res
        ▼ drawable
            girl.jpg
        ▼ layout
            activity_main.xml
        ▶ mipmap-hdpi
        ▶ mipmap-mdpi
        ▶ mipmap-xhdpi
        ▶ mipmap-xxhdpi
        ▶ mipmap-xxxhdpi
        ▶ values
      AndroidManifest.xml
```

如何使用 NDK 導入 OpenCV

16. 開啟 MainActivity.java 程式碼，並新增紅色圈選處的程式碼。



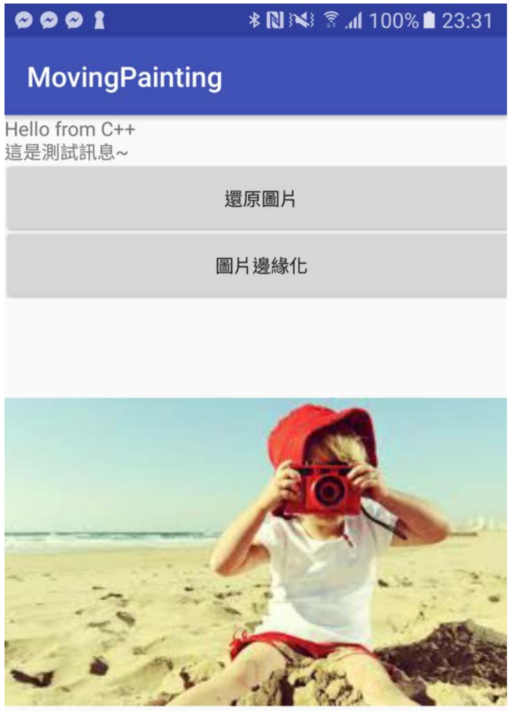
```
package movingpainting.sky.com.movingpainting;

import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    ImageView imgView;
    Button btnNDK, btnRestore;

    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary("native-lib");
        System.loadLibrary("testMessage");
        System.loadLibrary("imgCanny");
        System.loadLibrary("opencv_java3");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnRestore = (Button) this.findViewById(R.id.btnRestore);
        btnRestore.setOnClickListener(new ClickEvent());
        btnNDK = (Button) this.findViewById(R.id.btnNDK);
        btnNDK.setOnClickListener(new ClickEvent());
        imgView = (ImageView) this.findViewById(R.id.ImageView01);
        Bitmap img = ((BitmapDrawable) getResources().~~getDrawable~~(
                R.drawable.girl)).getBitmap();
        imgView.setImageBitmap(img);

        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.sample_text);
        tv.setText(stringFromJNI() + '\n' + getTestString());
    }
```



```
        tv.setText(stringFromJNI() + '\n' + getTestString());
    }

    class ClickEvent implements View.OnClickListener {
        public void onClick(View v) {
            //btnRestore.setText(ImgFun());
            if (v == btnNDK) {
                long current = System.currentTimeMillis();
                Bitmap img1 = ((BitmapDrawable) getResources().~~getDrawable~~(
                        R.drawable.girl)).getBitmap();
                int w = img1.getWidth(), h = img1.getHeight();
                int[] pix = new int[w * h];
                img1.getPixels(pix, 0, w, 0, 0, w, h);
                int[] resultInt = getCannyImg(pix, w, h);
                Bitmap resultImg = Bitmap.createBitmap(w, h, Bitmap.Config.RGB_565);
                resultImg.setPixels(resultInt, 0, w, 0, 0, w, h);
                long performance = System.currentTimeMillis() - current;
                imgView.setImageBitmap(resultImg);
            } else if (v == btnRestore) {
                Bitmap img2 = ((BitmapDrawable) getResources().~~getDrawable~~(
                        R.drawable.girl)).getBitmap();
                imgView.setImageBitmap(img2);
            }
        }
    }

    /**
     * A native method that is implemented by the 'native-lib' native library,
     * which is packaged with this application.
     */
    public native String stringFromJNI();
    public native String getTestString();
    public native int[] getCannyImg(int[] a,int b,int c);
}
```

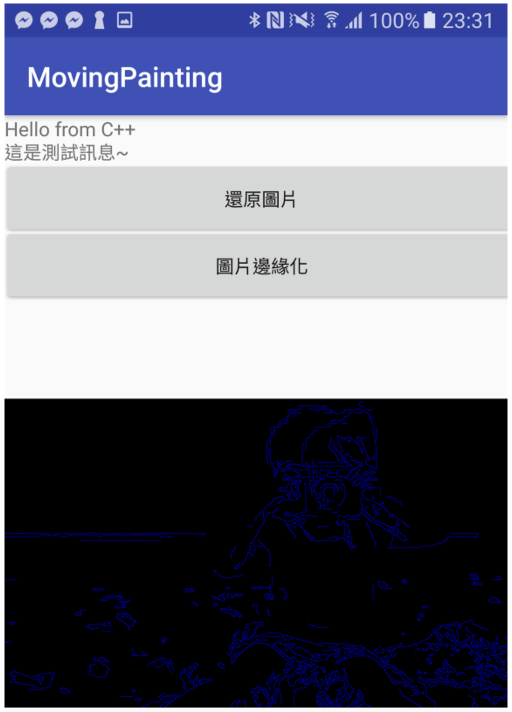
如何使用 NDK 導入 OpenCV

17. 點擊 Run -> Run ‘app’ 進行編譯，應該會看到 圖1 的執行結果。
18. 點選圖片邊緣化按鈕，出現畫面如 圖2 所示。
19. 點選還原圖片 按鈕，則還原圖片至 圖1 所示。



Hello from C++
這是測試訊息~

圖1 程式畫面 (原圖)



Hello from C++
這是測試訊息~

圖2 程式畫面 (邊緣化)

程式附錄 (imgCanny.cpp)

```
#include <jni.h>
#include <string>
#include <opencv2/opencv.hpp>

using namespace cv;
IplImage * change4channelTo3InIplImage(IplImage * src);

extern "C" {
JNIEXPORT jintArray JNICALL
    Java_movingpainting_sky_com_movingpainting_MainActivity_getCannyImg(
            JNIEnv *env, jobject obj, jintArray buf, int w, int h) {
        static jboolean false_ = false;
        static jboolean true_ = true;

        jint *cbuf;
        cbuf = env->GetIntArrayElements(buf, &false_);
        if (cbuf == NULL) {
            return 0;
        }

        Mat myimg(h, w, CV_8UC4, (unsigned char *) cbuf);
        IplImage image = IplImage(myimg);
        IplImage *image3channel = change4channelTo3InIplImage(&image);

        IplImage *pCannyImage = cvCreateImage(cvGetSize(image3channel), IPL_DEPTH_8U, 1);

        cvCanny(image3channel, pCannyImage, 50, 150, 3);

        int *outImage = new int[w * h];
        for (int i = 0; i < w * h; i++) {
            outImage[i] = (int) pCannyImage->imageData[i];
        }

        int size = w * h;
        jintArray result = env->NewIntArray(size);
        env->SetIntArrayRegion(result, 0, size, outImage);
        env->ReleaseIntArrayElements(buf, cbuf, 0);
        return result;
    }
}

IplImage * change4channelTo3InIplImage(IplImage * src) {
    if (src->nChannels != 4) {
        return NULL;
    }

    IplImage * destImg = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 3);
    for (int row = 0; row < src->height; row++) {
        for (int col = 0; col < src->width; col++) {
            CvScalar s = cvGet2D(src, row, col);
            cvSet2D(destImg, row, col, s);
        }
    }

    return destImg;
}
```

程式附錄 (MainActivity.java)

```java
package movingpainting.sky.com.movingpainting;

import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    ImageView imgView;
    Button btnNDK, btnRestore;

    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary("native-lib");
        System.loadLibrary("testMessage");
        System.loadLibrary("imgCanny");
        System.loadLibrary("opencv_java3");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnRestore = (Button) this.findViewById(R.id.btnRestore);
        btnRestore.setOnClickListener(new ClickEvent());
        btnNDK = (Button) this.findViewById(R.id.btnNDK);
        btnNDK.setOnClickListener(new ClickEvent());
        imgView = (ImageView) this.findViewById(R.id.ImageView01);
        Bitmap img = ((BitmapDrawable) getResources().getDrawable(
                R.drawable.girl)).getBitmap();
        imgView.setImageBitmap(img);

        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.sample_text);
        tv.setText(stringFromJNI() + '\n' + getTestString());
    }

    class ClickEvent implements View.OnClickListener {
        public void onClick(View v) {
            //btnRestore.setText(ImgFun());
            if (v == btnNDK) {
                long current = System.currentTimeMillis();
                Bitmap img1 = ((BitmapDrawable) getResources().getDrawable(
                        R.drawable.girl)).getBitmap();
                int w = img1.getWidth(), h = img1.getHeight();
                int[] pix = new int[w * h];
                img1.getPixels(pix, 0, w, 0, 0, w, h);
                int[] resultInt = getCannyImg(pix, w, h);
                Bitmap resultImg = Bitmap.createBitmap(w, h, Bitmap.Config.RGB_565);
                resultImg.setPixels(resultInt, 0, w, 0, 0, w, h);
                long performance = System.currentTimeMillis() - current;
                imgView.setImageBitmap(resultImg);
```

程式附錄 (activity_main.xml)

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="movingpainting.sky.com.movingpainting.MainActivity">

  <LinearLayout
    android:layout_width="368dp"
    android:layout_height="495dp"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="8dp"
    tools:layout_editor_absoluteX="8dp">

    <TextView
      android:id="@+id/sample_text"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="TextView" />

    <Button
      android:id="@+id/btnRestore"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="還原圖片" />

    <Button
      android:id="@+id/btnNDK"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="圖片邊緣化" />

    <ImageView
      android:id="@+id/ImageView01"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      app:srcCompat="@mipmap/ic_launcher" />
  </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

程式附錄 (app的build.gradle)

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

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "movingpainting.sky.com.movingpainting"
        minSdkVersion 21
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        externalNativeBuild {
            cmake {
                cppFlags "-std=c++11 -frtti -fexceptions"
            }
        }
        ndk{
            abiFilters "armeabi"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    externalNativeBuild {
        cmake {
            path "CMakeLists.txt"
        }
    }
}

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-v7:25.2.0'
    compile 'com.android.support.constraint:constraint-layout:1.0.1'
```

程式附錄 (CMakeLists.txt)

```
# For more information about using CMake with Android Studio, read the
# documentation: https://d.android.com/studio/projects/add-native-code.html

# Sets the minimum version of CMake required to build the native library.

cmake_minimum_required(VERSION 3.4.1)
set(lib_src_DIR ${CMAKE_SOURCE_DIR}/src/main/jniLibs/${ANDROID_ABI})
include_directories(${CMAKE_SOURCE_DIR}/src/main/cpp/include)

add_library( opencv_java3-lib
        SHARED
        IMPORTED)
set_target_properties( # Specifies the target library.
            opencv_java3-lib
            # Specifies the parameter you want to define.
            PROPERTIES IMPORTED_LOCATION
            # Provides the path to the library you want to import.
            ${lib_src_DIR}/libopencv_java3.so)

# Creates and names a library, sets it as either STATIC
# or SHARED, and provides the relative paths to its source code.
# You can define multiple libraries, and CMake builds them for you.
# Gradle automatically packages shared libraries with your APK.

add_library( # Sets the name of the library.
        imgCanny

        # Sets the library as a shared library.
        SHARED

        # Provides a relative path to your source file(s).
        src/main/cpp/imgCanny.cpp )

add_library( # Sets the name of the library.
        native-lib

        # Sets the library as a shared library.
        SHARED

        # Provides a relative path to your source file(s).
        src/main/cpp/native-lib.cpp )

add_library( # Sets the name of the library.
        testMessage

        # Sets the library as a shared library.
        SHARED
```