

Android Studio 配合 Gradle 及 Unity 交互笔记

2017-07-25

目 录

1 Gradle 安装	3
2 Gradle 编译 Android app	3
3 Android Studio 编译 APK	7
4 Android Studio 与 Unity 交互	14

1 Gradle 安装

根据文档指示，使用 Scoop 来安装 Gradle，安装 Scoop 需要在 Powershell 中执行下面的命令：

```
iex (new-object net.webclient).downloadstring('https://get.scoop.sh')
```

可能提示要修改策略，安装提示执行即可，下面是安装截图：



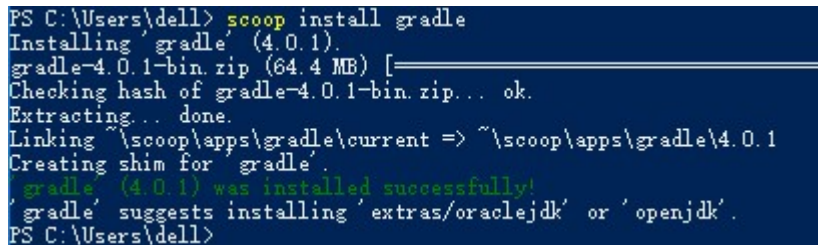
```
Windows PowerShell
版权所有 (C) 2016 Microsoft Corporation。保留所有权利。

PS C:\Users\dell> iex (new-object net.webclient).downloadstring('https://get.scoop.sh')
PowerShell requires an execution policy of 'RemoteSigned' to run Scoop.
To make this change please run:
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
PS C:\Users\dell> Set-ExecutionPolicy RemoteSigned -scope CurrentUser

执行策略更改
执行策略可以帮助你防止执行不信任的脚本。更改执行策略可能会产生安全风险，如 http://go.microsoft.com/fwlink/?LinkID=294672。
中的 about Execution Policies 帮助主题所述。是否要更改执行策略?
[Y] 是(Y) [A] 全是(A) [N] 否(N) [L] 全否(L) [S] 暂停(S) [?] 帮助 (默认值为“N”): Y
PS C:\Users\dell> iex (new-object net.webclient).downloadstring('https://get.scoop.sh')
Initializing...
Downloading...
Extracting...
Creating shim...
Adding \'scoop\' shims to your path.
Scoop was installed successfully!
Type \'scoop help\' for instructions.
PS C:\Users\dell>
```

这样只要执行下面的命令即可安装 Gradle 了：

scoop install gradle



```
PS C:\Users\dell> scoop install gradle
Installing 'gradle' (4.0.1).
gradle-4.0.1-bin.zip (64.4 MB) [=====]
Checking hash of gradle-4.0.1-bin.zip... ok.
Extracting... done.
Linking '~\scoop\apps\gradle\current => ~\scoop\apps\gradle\4.0.1
Creating shim for 'gradle'.
gradle (4.0.1) was installed successfully!
'gradle' suggests installing 'extras/oraclejdk' or 'openjdk'.
PS C:\Users\dell>
```

2 Gradle 编译 Android app

在安装完 Gradle 后可以到该地址查看如何使用 Gradle 编译安卓 App：
<https://guides.gradle.org/building-android-apps/>。文档介绍说基本只会通过支持的 IDE 也就是 Android Studio 使用 Gradle 来编译安卓应用程序。这里会创建一个 Hello World 程序，了解 Gradle 构建文件并执行常用指令。开始前需要 Android Studio 2.4 或更高版本，包括最新的 Android SDK，还需要 JDK1.7 或更高版本。

打开 Android Studio 后选择第一个创建新的 Android Studio 项目，名称为 HelloWorldGradle：

Configure your new project

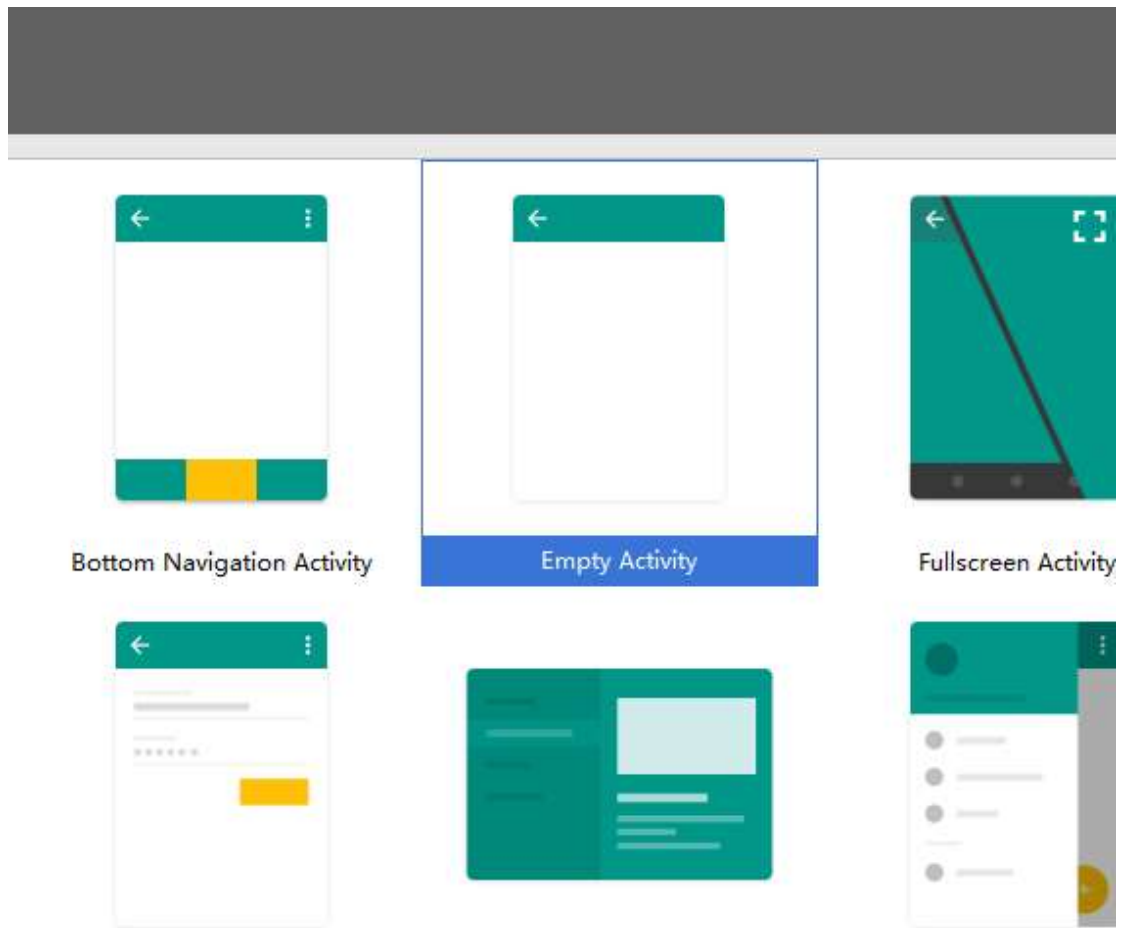
Application name: HelloWorldGradle

Company domain: gradle.org

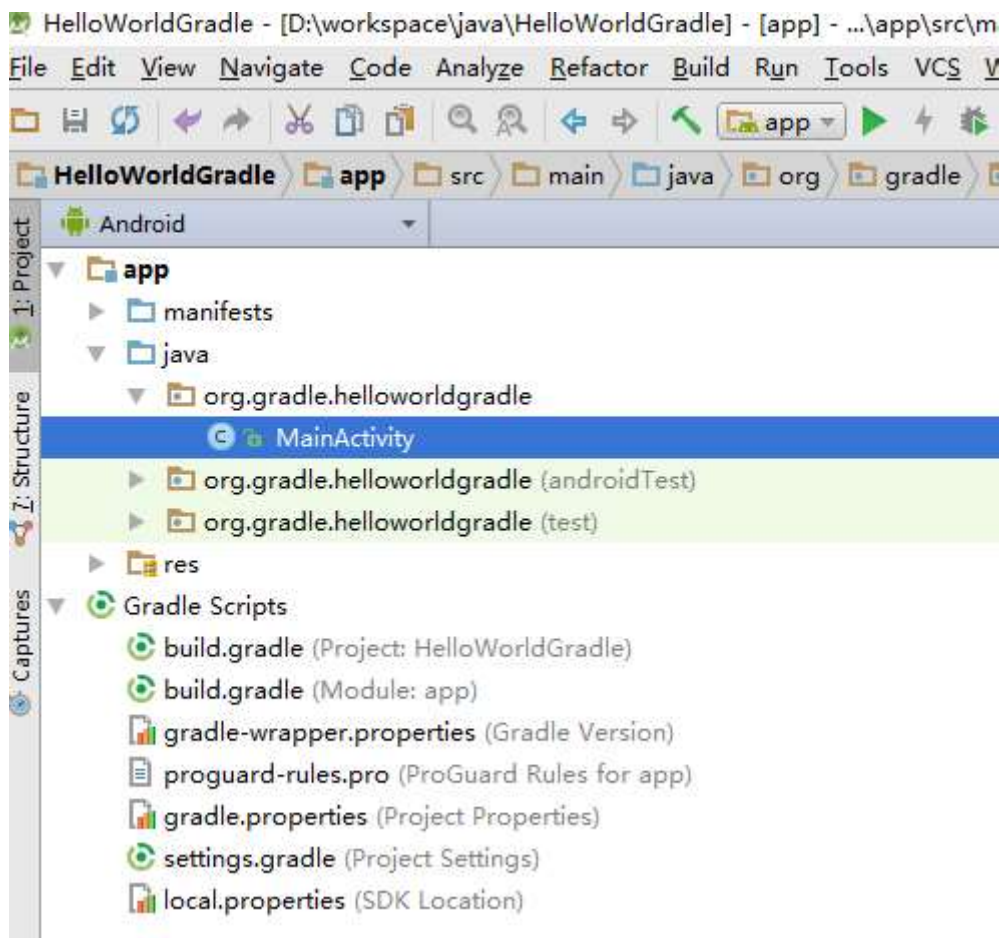
Package name: org.gradle.helloworldgradle

☐ Include C++ support

选择最低要求的 SDK 级别，可以选择 22 级也就是安卓 5.1。然后是创建空活动：



名称默认即可。生成的 Gradle 文件如下：



安卓项目是多个 Gradle 编译项目构成的，顶层是 build.gradle 文件，子目录是 app，同事也有自己的 build.gradle 文件。顶层的是 Project: HelloWorldGradle，app 层的是 Module: app。然后是 gradle.properties 文件，这个可能有两个。一个是本地项目的，另一个是可选的，只有在全局 gradle.properties 存在于用户主目录的 .gradle 子目录中才会产生。

settings.gradle 是 Gradle 多项目使用的配置文件，应该只包含一行：

```
include ':app'
```

这告诉 Gradle 子目录 app 也是一个 Gradle 项目。最后一个文件就是 gradle-wrapper.properties，也就是配置 Gradle Wrapper 的。

看下顶层 Gradle 配置文件：

```

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.3'

        // 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 rootProject.buildDir
}

```

开始的第一个 `repositories` 是下载插件的部分，`dependencies` 的部分标记安卓插件，`allprojects` 部分是顶级配置，最后是 `ad hoc` 任务。

Gradle 定义了一个特定域语言（Domain-Specific Language，DSL），`buildscript` 标记就是 DSL 的一部分。后面的清理任务会清除 `buildDir`。

然后看 `build.gradle` 文件的内容：

```

apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "26.0.0"
    defaultConfig {
        applicationId "org.gradle.helloworldgradle"
        minSdkVersion 22
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

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'
}

```

这部分更加接近安卓，简短说：

compileSdkVersion 和 buildToolsVersion 应该是最新版本，defaultConfig 标签是 app 所有变量共享的属性，minSdkVersion 是最低 Android API 要求，applicationId 是要唯一的，后面还要两种编译类型，有 debug 和 release 版本两种选择。

在 android 块有 app 所需的库的配置，这里有 compile、testCompile 和 androidTestCompile 的配置。最简单的 testCompile 依赖仅仅是 Junit4。

编译的过程看 Gradle 可以在终端中进行，但是直接点击 Android Studio 的构建运行也是可以的，我现在都怀疑是不是安装完 Android Studio 后都不用搞 Gradle 安装的。

3 Android Studio 编译 APK

在 build.gradle 文件中一般下面的信息：

```

android {
    compileSdkVersion 25
    buildToolsVersion "26.0.0"
    defaultConfig {
        applicationId "com.beavermagic.learningandroidui"
        minSdkVersion 22
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnit4"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

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'
}

```

前面也说过一些，这里补充如下：新建的一个项目，一般 Android Studio 会要求你填写最低的 API 登记。比如我的手机运行的是安卓 5.1，对应 API Level 22，因此选择了 22，那么自动生成的这个文件中 minSdkVersion 就是 22。compileSdkVersion 是你 SDK 的版本，buildToolsVersion 是构建工具的版本。可以用高版本的 buildTools 构建低版本的 SDK 工程，如上面的截图所示。targetSdkVersion 是 Android 提供向前兼容的主要依据，表明该 application 已经兼容从 minSdkVersion 至 targetSdkVersion 之间所有 api 的变化。在 targetSdkVersion 更新之前系统不会应用最新的行为变化。如果手机运行的正好是 targetSdkVersion 版本那么则不用使用兼容模式。appcompat-v7 这个按钮支持包是为了考虑 API level 7(即 Android 2.1)及以上版本而设计的，但是 v7 是要依赖 v4 这个包的，v7 支持了 ActionBar 以及一些 Theme 的兼容。官方有一些解释：

v7 appcompat 库

此库添加了对[操作栏](#)用户界面设计模式的支持。此库包含对 [Material Design](#) 用户界面实现的支持。

注：此库依赖于 v4 支持库。

下面是 v7 appcompat 库中包含的一些关键类：

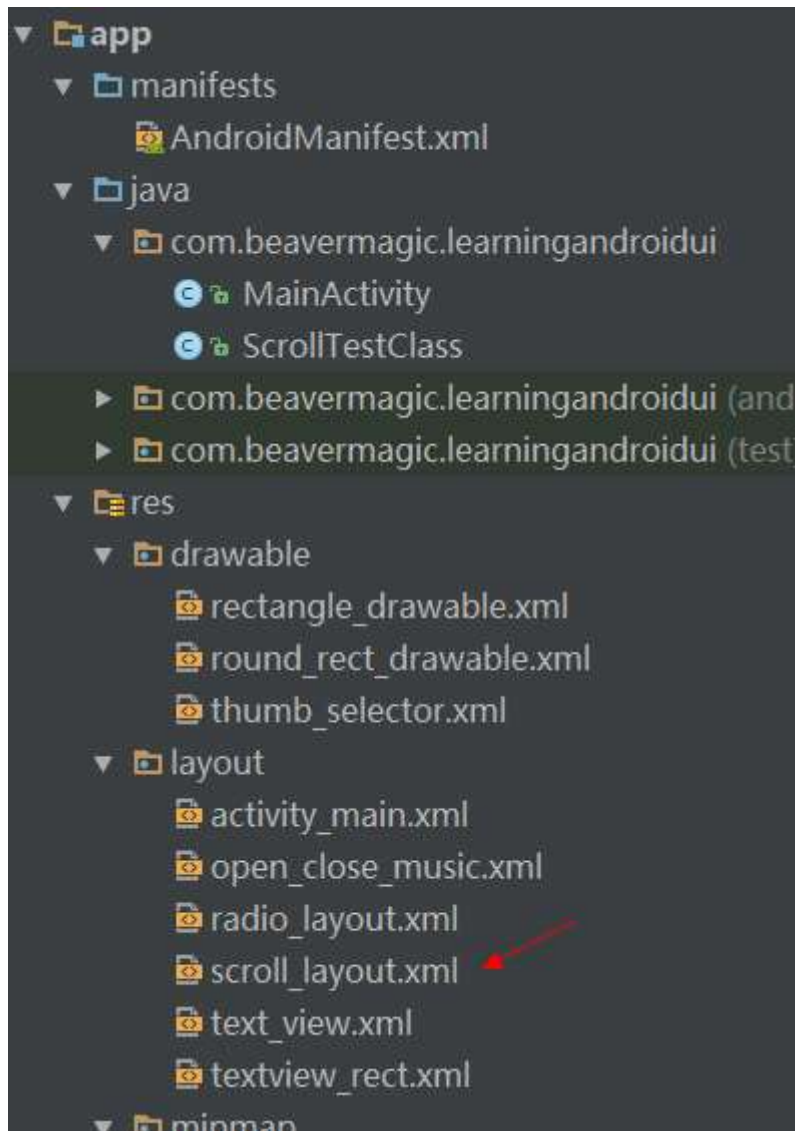
- [ActionBar](#) - 提供操作栏用户界面模式的实现。如需了解有关使用操作栏的详细信息，请参阅[操作栏](#)开发者指南。
- [AppCompatActivity](#) - 添加一个应用 Activity 类，此类可作为使用支持库操作栏实现的 Activity 的基础类。
- [AppCompatDialog](#) - 添加一个对话框类，此类可作为 AppCompat 主题对话框的基础类。
- [ShareActionProvider](#) - 包含对可以添加到操作栏中的标准化分享操作（例如电子邮件或发帖至社交应用）的支持。

此库的 Gradle 构建脚本依赖关系标识符如下所示：

```
com.android.support:appcompat-v7:24.2.0
```

在布局的时候如果使用了限制布局，则该文件会有关于该布局的引用，这个是单独的插件。

下面是一个例子，添加滑动条并设置按钮跳到最前或最后。首先在 layout 的目录下新建布局文件如 scroll_layout.xml，如下：



其内容如下：

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<ScrollView
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:id="@id/scrollView"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:scrollbars="vertical"
```

```
    android:fadingEdge="vertical">
```

```
        <LinearLayout                xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:orientation="vertical"
```

```
                                android:layout_width="fill_parent"
```

```
                                android:layout_height="fill_parent">
```

```

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ffffff"
    android:text="滑动条"/>

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Button
        android:id="@+id/btn_down"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff654321"
        android:layout_weight="1"
        android:text="滚到底部" />

    <Button
        android:id="@+id/btn_up"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#fffedcba"
        android:layout_weight="1"
        android:text="滚到顶部"/>

</LinearLayout>

<TextView
    android:id="@+id/txt_show"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ffffff"
    android:text="填充内容"/>

</LinearLayout>

```



```

</ScrollView>

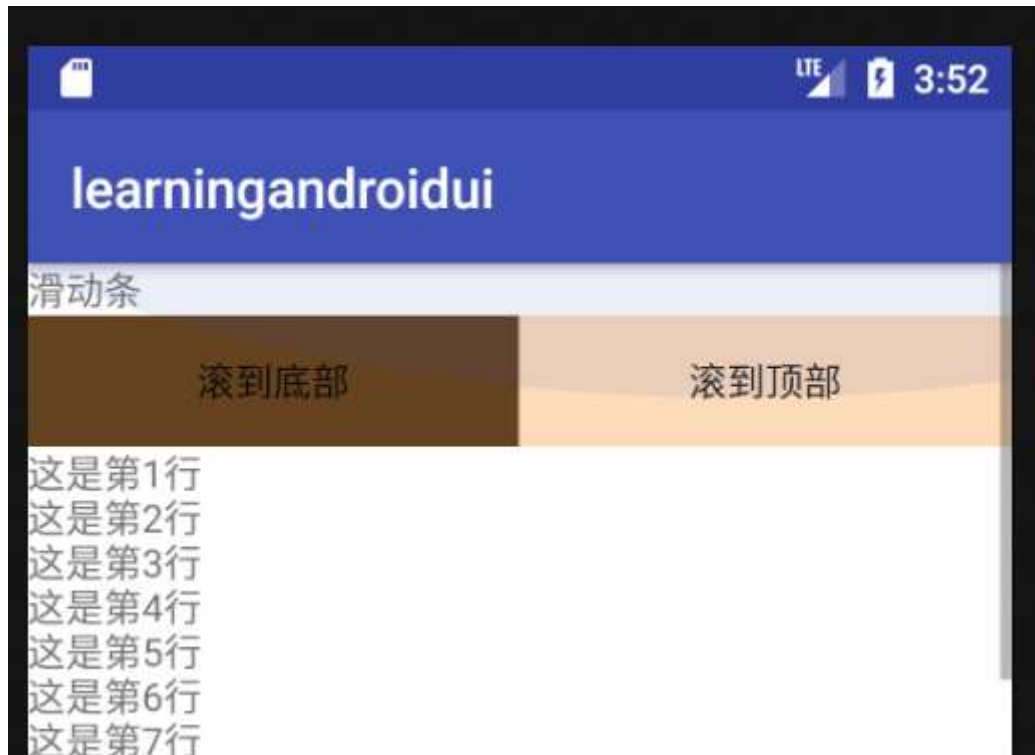
```

稍微说下，ScrollView 在最外层，内部套了线性布局，线性布局可以再嵌套线性布局或者 Button 等组件，需要获取的组件应该赋予 id 以便再 java 代码中取得该组件响应一些特定逻辑。

为了调试方便，可以建立模拟器，这里我的虚拟设备是 API 24 的 Nexus5：

Type	Name	Play Sto...	Resolution	API	Target	CPU/...
	Nexus 5 API 24		1080 × 1920: xxhdpi	24	Android 7.0 (Google Pl...	x86

运行后如下：



这里的一些填充内容和按钮的响应是代码控制的，先把代码扯出来：
package com.beavermagic.learningandroidui;

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
```

```
    //RadioGroup radGroup = (RadioGroup)findViewById(R.id.radioGroup);
```

```
    // 第一种获得单选按钮值的方法
```

```
    // 为 radioGroup 设置一个监听器 setOnCheckedChangeListener()
```

```
    /*
```

```

radGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId)
    {
        RadioButton radbtn = (RadioButton)findViewById(checkedId);
        Toast.makeText(getApplicationContext(), "按钮值发生改变, 你选了" +
radbtn.getText(), Toast.LENGTH_LONG).show();
    }
});
*/

```

```

private Button btn_down;
private Button btn_up;
private ScrollView scrollView;
private TextView txt_show;

```

```

@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.scroll_layout);
    bindViews();
}

```

```

private void bindViews()
{
    btn_down = (Button) findViewById(R.id.btn_down);
    btn_up = (Button) findViewById(R.id.btn_up);
    scrollView = (ScrollView) findViewById(R.id.scrollView);
    txt_show = (TextView) findViewById(R.id.txt_show);

    btn_down.setOnClickListener(this);
    btn_up.setOnClickListener(this);

    StringBuilder sb = new StringBuilder();

    for(int i = 1; i <= 100; i++)
    {
        sb.append("这是第" + i + "行" + "\n");
    }

    txt_show.setText(sb.toString());
}

```

```

@Override
public void onClick(View v)
{
    switch(v.getId())
    {
        case R.id.btn_down:
            scrollView.fullScroll(ScrollView.FOCUS_DOWN);
            break;

        case R.id.btn_up:
            scrollView.fullScroll(ScrollView.FOCUS_UP);
            break;
    }
}
}

```

这里是 MainActivity.java 的内容，里面 setContentView 指定加载 scroll_layout 这个布局文件，后面的 bindViews 方法向界面添加许多行以便滑动条可以有足够的内容滑动。这个类实现了点击的监听，onClick 后判断按钮是哪一个，可以分别通过 ScrollView.FOCUS_DOWN 和 FOCUS_UP 来跳转。点击 Build->Build APK 后会有如下文件生成：

名称	修改日期	类型
 app-debug.apk	2017/7/28 10:31	APK

可以安装到手机上进行真机体验。

4 Android Studio 与 Unity 交互

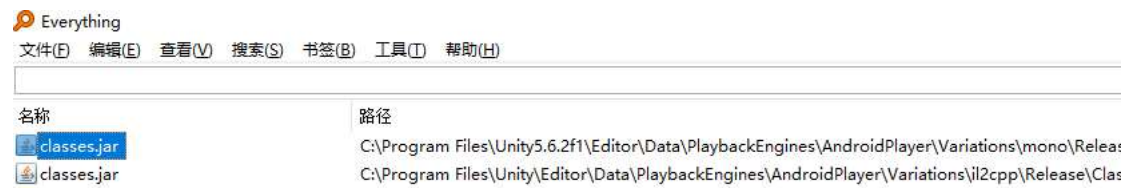
新入坑，主要也是参考他人的教程，链接如下：
<http://blog.csdn.net/yangxuan0261/article/details/52427119>。这里介绍有两种方式：

- 实现unity掉java里面的代码有两种方式
 1. 第一种方式，自己写个java类，jni需要的 中转站cpp、Android 烦，官网例子直接可以下，传送门
<http://docs.unity3d.com/Manual/PluginsForAndroid.html>)
 2. 第二种方式，直接使用unity分装好的类 **AndroidJavaClass** 等

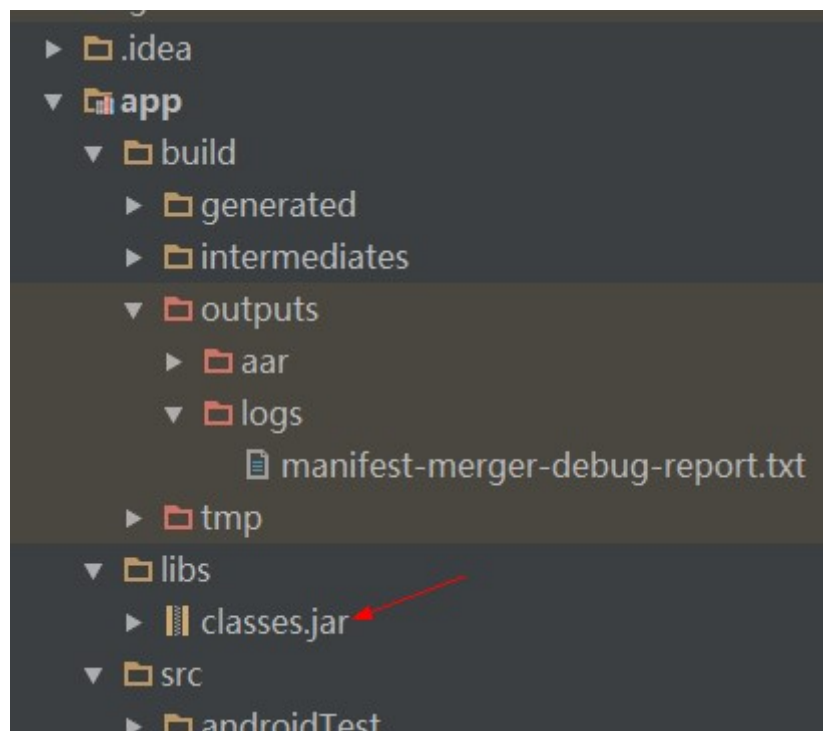
选择第二种方式会很方便。以前有这种交互需求的会使用 Eclipse 来处理，生成 jar 包。现在使用 Android Studio 来做会生成 aar 包给 Unity 使用。

先使用 AS 开始新建一个工程，设置最低 sdk 等级，我这里是 22。在 Unity 的目录下找到 classes.jar 文件，我的路径是 C:\Program

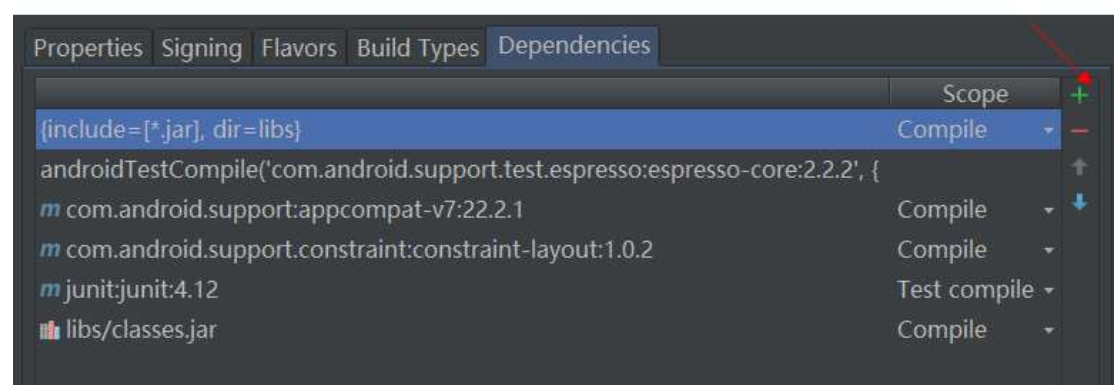
Files\Unity5.6.2f1\Editor\Data\PlaybackEngines\AndroidPlayer\Variations\mono\Release\Classes\classes.jar。如果不好找，可以使用 Everything 这个工具，搜索电脑上的东西相当方便，炫耀一下：



将该文件复制到 libs 中，直接在外边 Ctrl+C，在 libs 上 Ctrl+V 即可，好像不能使用拖进这种复制方式：



然后在项目（app）上右击选择 Open Module Settings，在里面选择添加 jar 包依赖，如下：



然后 MainActivity.java 修改如下：

```
package com.pushforward.androidstudioandunity;
```

```
import android.app.AlertDialog;
```

```

import android.content.Intent;
import android.os.Vibrator;
import android.os.Bundle;
import android.widget.Toast;

import com.unity3d.player.UnityPlayer;
import com.unity3d.player.UnityPlayerActivity;

public class MainActivity extends UnityPlayerActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        public String ShowDialog(final String _title, final String _content)
        {
            runOnUiThread(new Runnable(){
                @Override
                public void run()
                {
                    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                    builder.setTitle(_title).setMessage(_content).setPositiveButton("Down",
null);
                    builder.show();
                }
            });

            return "Java return";
        }

        public void ShowTestActivity(String str)
        {
            Intent intent = new Intent(this, TestActivity.class);
            this.startActivity(intent);
        }

        // 定义一个显示 Toast 的方法，在 Unity 中调用此方法
        public void ShowToast(final String mStr2Show)
        {
            // 同样需要在 UI 线程下执行
            runOnUiThread(new Runnable() {
                @Override

```



```

        public void run() {
            Toast.makeText(getApplicationContext(), mStr2Show,
Toast.LENGTH_LONG).show();
        }
    });
}

// 定义一个手机振动的方法，在 Unity 中调用此方法
public void SetVibrator()
{
    Vibrator mVibrator = (Vibrator)getSystemService( VIBRATOR_SERVICE);
    mVibrator.vibrate(new long[]{200, 2000, 2000, 200, 200, 200}, -1); // -1 表示不重
复，0 表示循环振动
}

// 第一个参数是 Unity 中的对象名字，记住是对象名字不是脚本类名
// 第二个参数是函数名
// 第三个参数是传给函数的参数，目前只看到一个参数，并且是 string 的，自己传进去
转吧
public void callUnityFunc(String _objName, String _funcStr, String _content)
{
    UnityPlayer.UnitySendMessage(_objName, _funcStr, "Come from:" + _content);
}
}

```

这个文件写入了很多的内容，实际上简单测试的时候只用了一个。在开始的时候 MainActivity 继承 UnityPlayerActivity，里面写了一个 ShowTestActivity 方法，该方法会启动 TestActivity，显示那几个图标组成的界面。其中 TestActivity.java 的内容为：

```
package com.pushforward.androidstudioandunity;
```

```

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

```

```

/**
 * Created by dell on 2017/7/27.
 */

```

```

public class TestActivity extends Activity implements View.OnClickListener
{
    ImageView imgView;

```

```

@Override
protected void onCreate(@Nullable Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    imageView = (ImageView)findViewById(R.id.img1);
    imageView.setOnClickListener(this);
}

public void onClick(View view)
{
    int i = view.getId();

    if (i == R.id.img1)
    {
        SetUnity();
    }
}

void SetUnity()
{
    Intent intent = new Intent(this, MainActivity.class);
    this.startActivity(intent);
}
}

```

这个类 onCreate 中写好了监听 img1 的点击事件，点击后会进入 Unity。

AndroidManifest.xml 文件内容为：

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pushforward.androidstudioandunity">

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <!-- 一定要加上这句 -->

```

```

        <meta-data android:name="unityplayer.UnityActivity" android:value="true" />
    </activity>
    <activity android:name=".TestActivity">
    </activity>
</application>

```

</manifest>

这个是要重点注意的，如果出错了可以仔细对比该文件看需要修改和删除哪些内容。

Activity_main.xml 为：

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

```

<!-- 这个是在容器中央的 -->

```

<ImageView
    android:id="@+id/img1"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_centerInParent="true"
    android:src="@drawable/and"/>

```

<!-- 在中间图片的左边 -->

```

<ImageView
    android:id="@+id/img2"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_toLeftOf="@id/img1"
    android:layout_centerVertical="true"
    android:src="@drawable/chro"/>

```

<!-- 在中间图片的右边 -->

```

<ImageView
    android:id="@+id/img3"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_toRightOf="@id/img1"
    android:layout_centerVertical="true"
    android:src="@drawable/fire"/>

```

<!-- 在中间图片的上面 -->

```

<ImageView

```

```

        android:id="@+id/img4"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_above="@id/img1"
        android:layout_centerHorizontal="true"
        android:src="@drawable/github"/>

```

<!-- 在中间图片的下面 -->

```

<ImageView
    android:id="@+id/img5"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_below="@id/img1"
    android:layout_centerHorizontal="true"
    android:src="@drawable/mic"/>

```

</RelativeLayout>

布局文件的内容就这些，没多少重点。build.gradle 文件为：

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

```
// 这样才能导出一个 aar 包
```

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

```

android {
    compileSdkVersion 22
    buildToolsVersion "26.0.0"
    defaultConfig {
        //applicationId "com.pushforward.androidstudioandunity"
        minSdkVersion 22
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
}

```

```

androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2',{
    exclude group: 'com.android.support', module: 'support-annotations'
})
compile "com.android.support:appcompat-v7:22.2.1"
compile 'com.android.support.constraint:constraint-layout:1.0.2'
testCompile 'junit:junit:4.12'
compile files('libs/classes.jar')
}

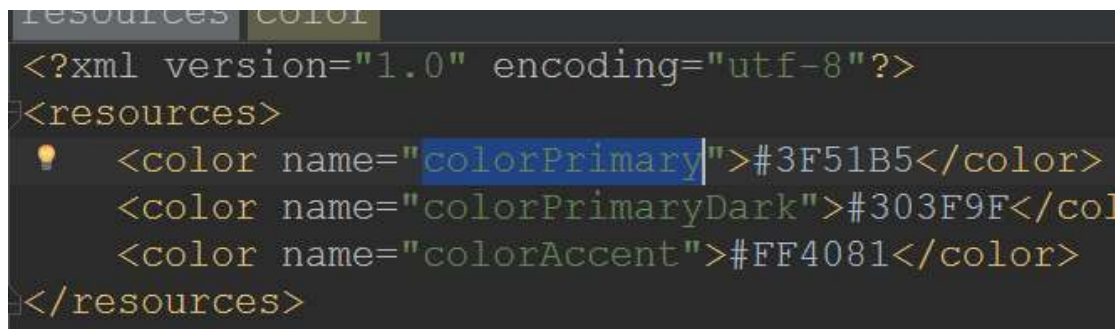
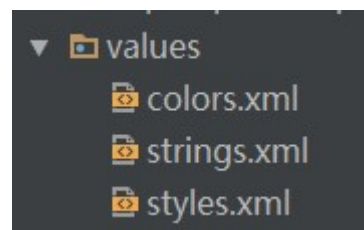
```

开始的时候修改为 library 以便导出 aar 包，使用的 SDK 的版本更具需要修改。另外需要注意的是如果使用限制布局可能无法成功在 Unity 打包，或者需要导入一些包。

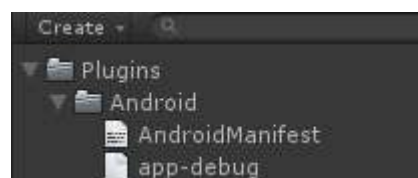
产生 aar 包，选择 Build->Build APK 会得到 app-debug.aar 包。然后用解压缩软件打开这个文件，删除 libs 下面的 classes.jar 和 res\values 目录下的文件：



删除 values 目录下的内容是因为开始的时候，values 下已经有这些东西了：



这里面的一些东西可能引入了其他的文件，如果也进入 Unity 会在打包的时候报错说找不到某些标记。然后将该文件导入到 Unity 中，同时 AndroidManifest 也一起导入：



注意其路径。新建一个脚本，内容如下：

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

```

```

using System.Runtime.InteropServices;
using UnityEngine.UI;

public class TestDLL : MonoBehaviour
{
    private Text mText;

    void Start()
    {
        mText = GameObject.Find("MsgText").GetComponent<Text>();
    }

    public void MyShowDialog()
    {
        // Android的Java接口
        AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
        AndroidJavaObject jo = jc.GetStatic<AndroidJavaObject>("currentActivity");

        // 参数
        string[] mObject = new string[2];

        mObject[0] = "Jar4Android";
        mObject[1] = "Wow, Amazing! It works!";

        // 调用方法
        string ret = jo.Call<string>("ShowDialog", mObject);
        SetMsg(ref ret);
    }

    public void MyShowToast()
    {
        AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
        AndroidJavaObject jo = jc.GetStatic<AndroidJavaObject>("currentActivity");
        jo.Call("ShowTestActivity", "");
    }

    public void MyInteraction()
    {
        AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
        AndroidJavaObject jo = jc.GetStatic<AndroidJavaObject>("currentActivity");
        jo.Call("callUnityFunc", "U2J2U", "BeCallFunc", "androidstudioandunity");
    }

    public void BeCallFunc(string content)

```

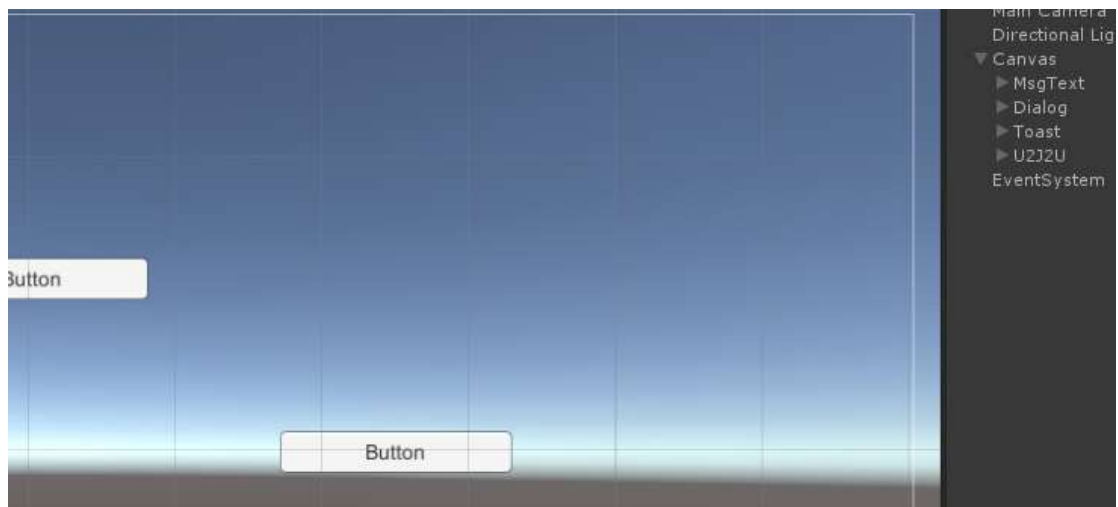
```

{
    SetMsg(ref content);
}

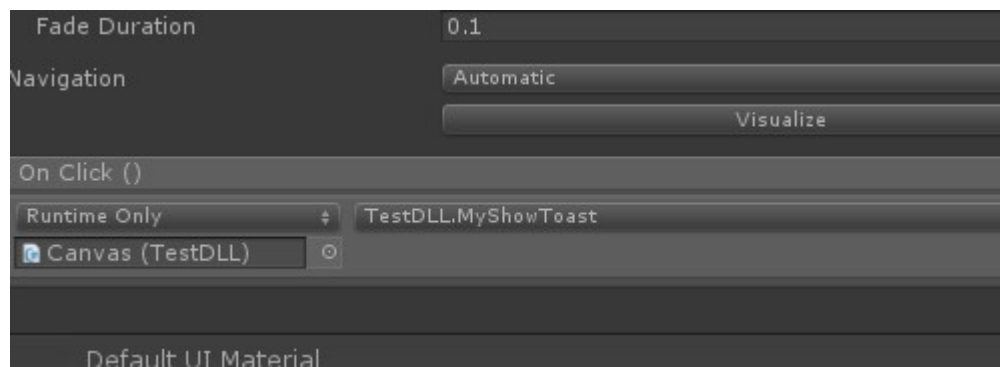
private void SetMsg(ref string str)
{
    mText.text = str;
}
}

```

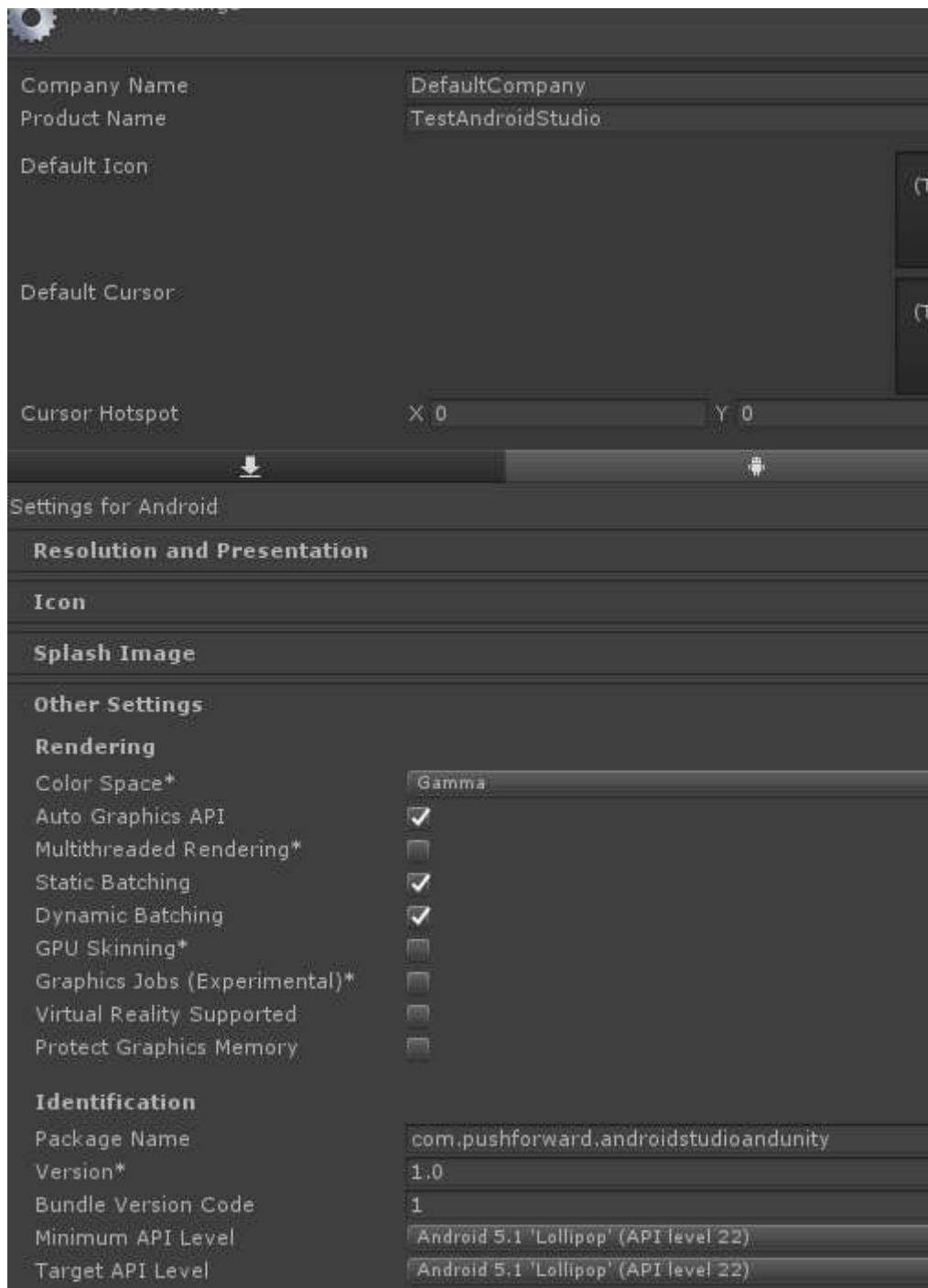
然后新添加几个按钮用于 Android 与 Unity 交互触发：



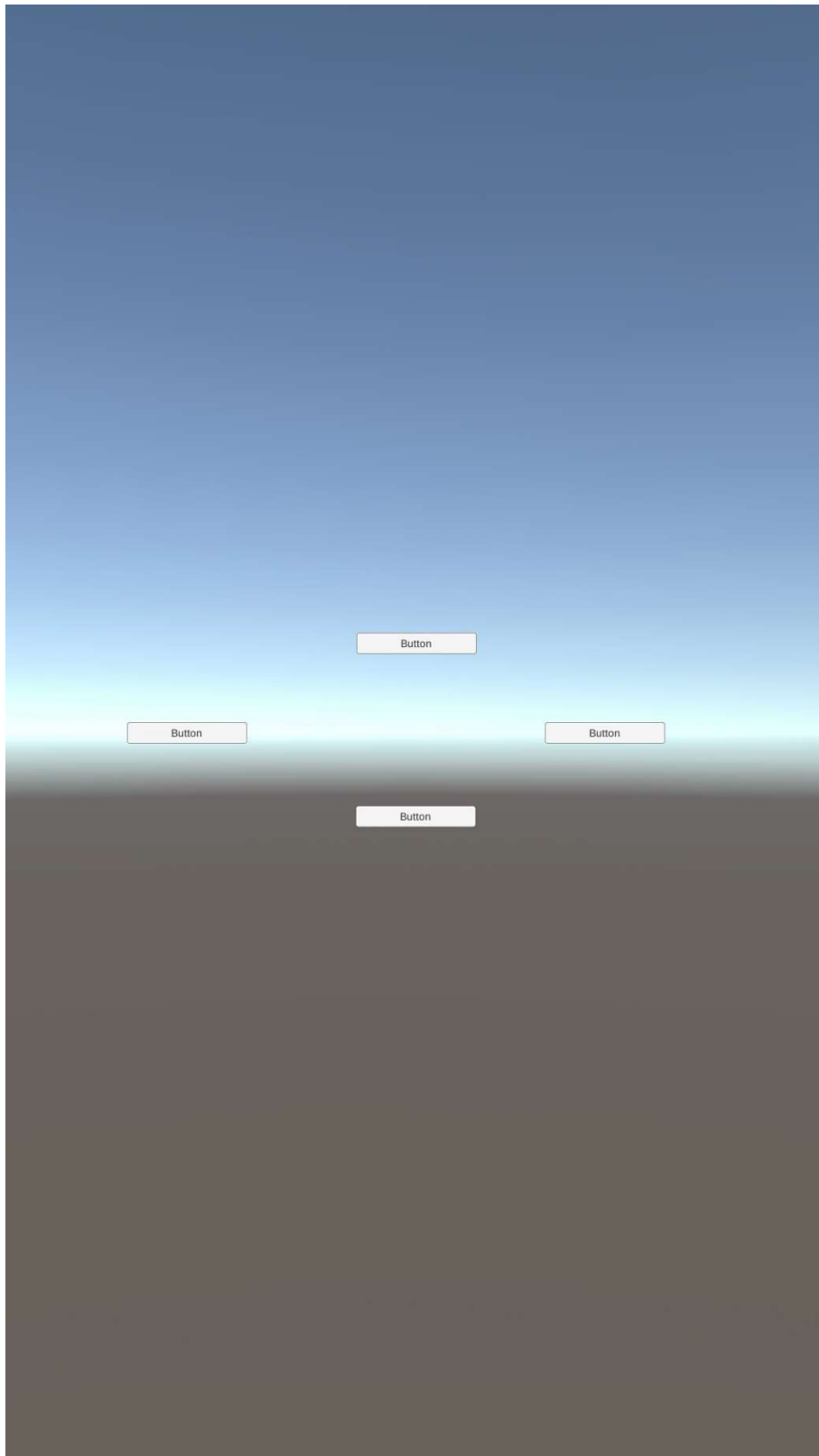
事件如下：



准备编译 APK，设置一下包名，等级是 22：



运行后先出现 Unity 设计的界面：



点击按钮，触发切换到 Android 界面：

14:04

0 K/s 4G 35

TestAndroidStudio



这里点击设定的一个图后会跳到 Unity 的界面，至此二者的交互测试就完成了。