

Unity Export 导出到 Android Studio 再打包大致过程

deepwaterooo

November 25, 2022







Contents

1 导出的 unity 项目文件大致是这样的	1
2 Android 创建、unity 导入	2
2.1 首先新建一个 Android 项目	2
2.2 将 unity 项目以 Module 的方式导入 Android	2
2.3 选择 unityLibrary 导入。点击 Finish	3
2.4 导入之后，为 Android 添加 unityLibrary 的引用	3
2.5 配置 Android 以及 unity 的 build.gradle 文件	3
3 Android 启动运行 unity	4
3.1 在 unity 的 AndroidManifest.xml 文件	4
3.2 在 app 的 AndroidManifest.xml 文件里，在图中位置加入这两行代码：	4
3.3 在 app 的 build.gradle 里加入这行代码。	5
3.4 在 app 的 main->res->values->strings.xml 里加入这行代码	5
3.5 点击按钮启动 unity(画蛇添足)	6
3.6 在 build.gradle 中申明包裹类名称	7
4 启动运行	8
5 Android Studio 类库中重复类的修复	8
6 安卓 Android Studio 库包中有依赖的库包的解决方案 7.2.2	9
6.1 比较好一点的, 是如下: 在项目的根目录的 build.gradle 里申明类库 unityLibrary 的依赖的文件路径就可找到	9
6.2 下面的只是一种解决方案, 可能还不是很好	11
6.3 在你工程根目录下新建一个文件夹 unitylibs , 将你的 aar 文件放入, 然后在该目录下新建一个 build.gradle 文件	11
6.4 在 settings.gradle 导入该工程	11
6.5 在你需要依赖的工程里面的 build.gradle 中增加依赖	11
7 那么现在就是说: 安卓 SDK 与 unity 的交互与打包基本没有问题了	12
7.1 FATAL EXCEPTION: main	14
7.2 类库包里的错误的修复问题	14
8 安卓设备上资源包的存放位置, 以及是否本地存放有需要的资源包	15

1 导出的 unity 项目文件大致是这样的

- 大致过程记一下, 用作参考, 原理还没有吃透, 细节又比较多, 容易忘记. 作个笔记记一下, 给自己用作参考

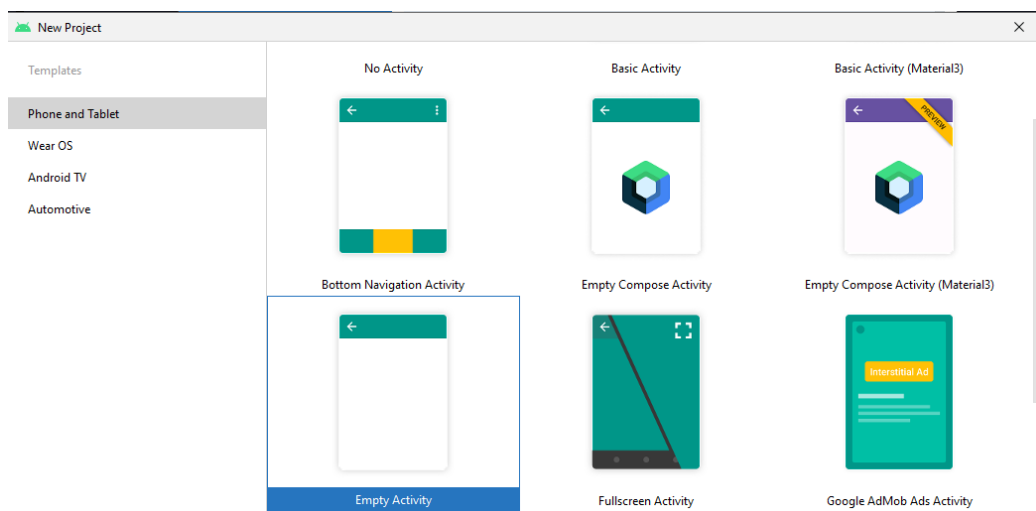
F (F:) > tmp > prevTetris19 >

Name	Date modified
 launcher	11/23/2022 1
 unityLibrary	11/23/2022 1
 build.gradle	11/23/2022 1
 gradle.properties	11/23/2022 1
 local.properties	11/23/2022 1
 settings.gradle	11/23/2022 1

- 下面是 2019 年的版本可以打出两个文件夹, 一个主工程, 一个类库的导出包, 2017 年我用的版本打不出来, 还需要想得再深一点多点儿, 到可以按照这个笔记过程打包才行
- 原始自己参考的项目是用 2017 版本的, 当时没有吃透这里的构建关系, 当时以为只能用 2017 的 unity 和 2017 的 Visual Studio 才能开发. 现在知道 2019 的版本能够导出自己可以调试的 Android Studio 项目, 而 unity 2017 版本的导出来自己还仍不知道该如何从 Android Studio 打包, 那么就暂时先用 2019 的版本, 先试图打出在安卓设备上可运行的包, 才能 move on.

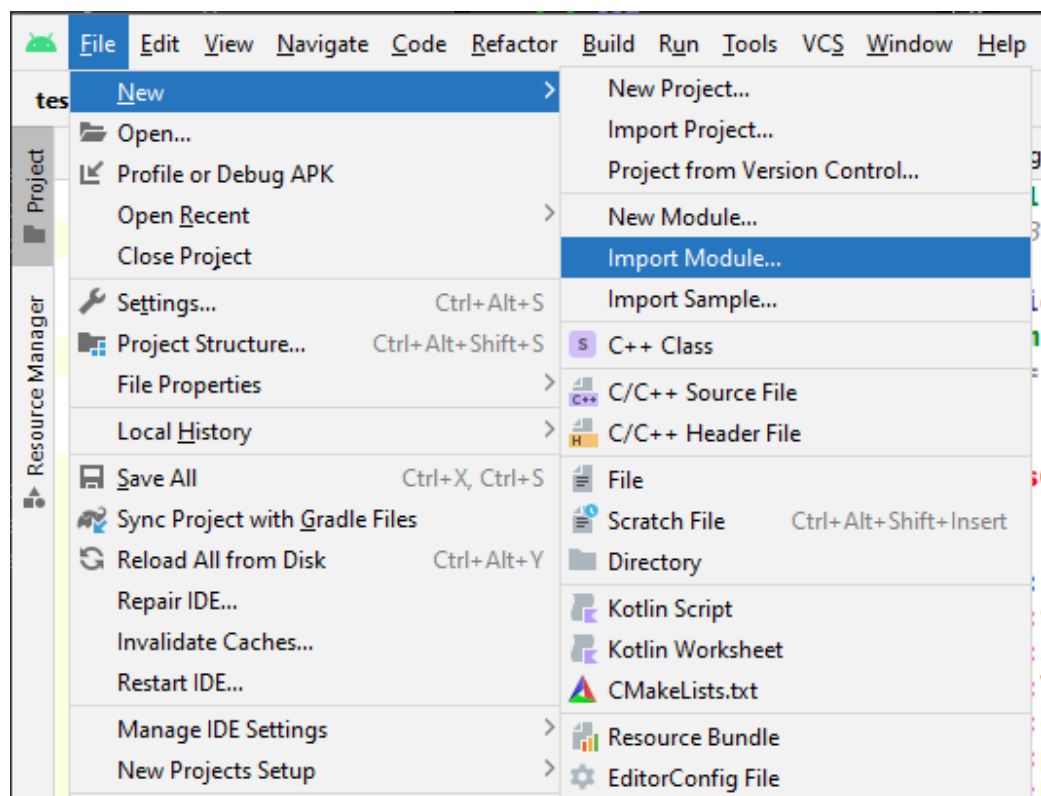
2 Android 创建、unity 导入

2.1 首先新建一个 Android 项目

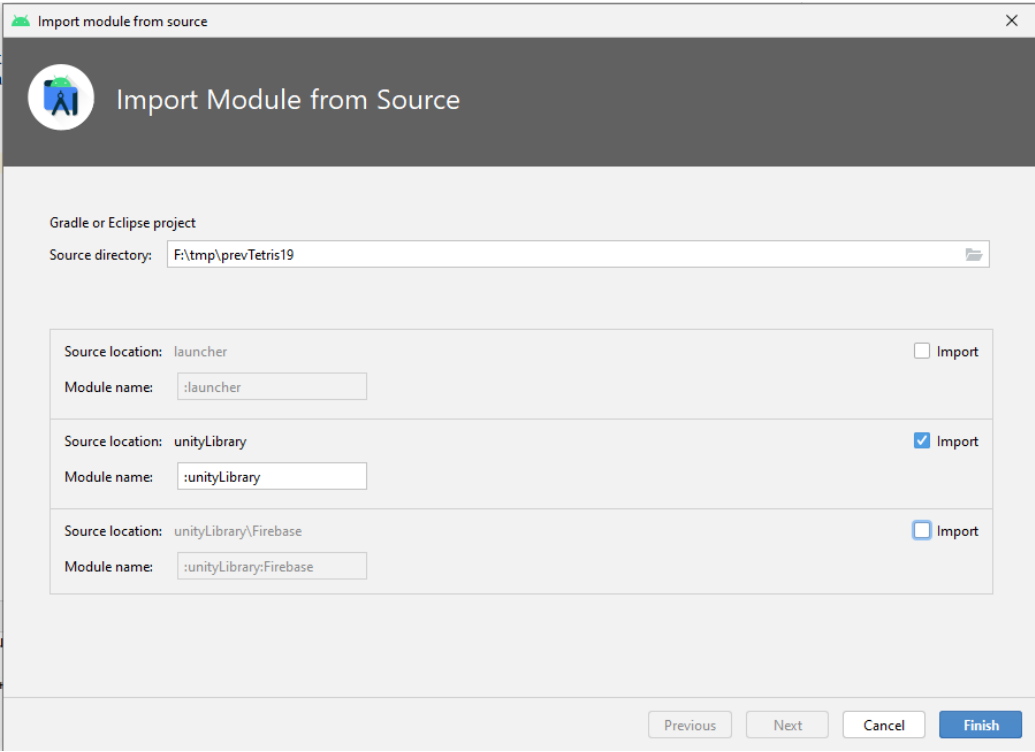


- 包名 Package name 跟 unity 的包名设置成一致, unity 包名一般是 **com.unity3d.player**。包名不一致的话, 我试过也可以实现, 但是在调用的时候要指明包, 容易混淆, 可能还有其他的一些问题, 个人也不是很清楚。推荐保持一致, 避免麻烦。Android 项目名 Name 等随意。

2.2 将 unity 项目以 Module 的方式导入 Android

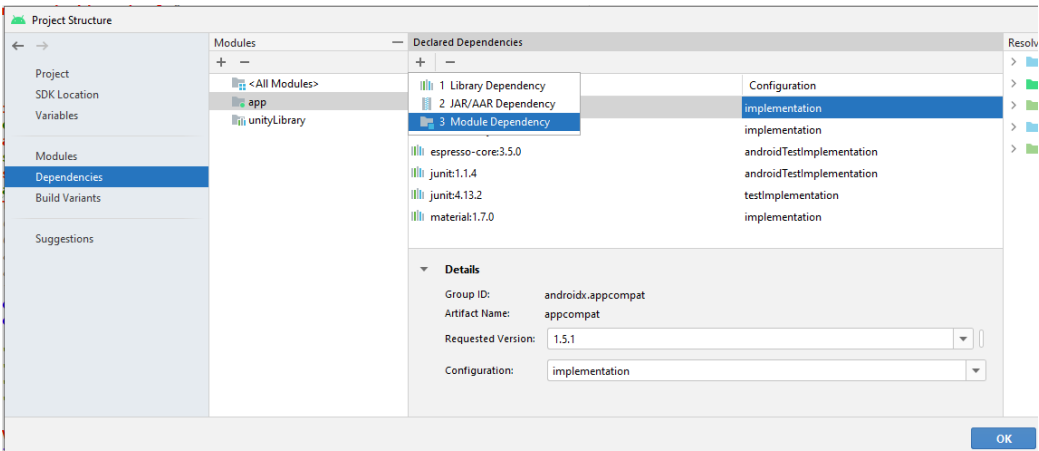


2.3 选择 unityLibrary 导入。点击 Finish



2.4 导入之后，为 Android 添加 unityLibrary 的引用

- 左上角 File——>Project Structure...
- 选择 Dependencies ——> app ，然后点击右边这个加号 + ，选择第三个 Moudule Dependency



- 勾选刚刚导入的 unity，点击 OK。再点击上图的 OK。

2.5 配置 Android 以及 unity 的 build.gradle 文件

- 将 SDK 配置成当前 Android 版本可以运行。Android 以及 unity 的 SDK 确保要一样，不然会报错，比如这个 minsdk。Build 无误就算是导入完成了！

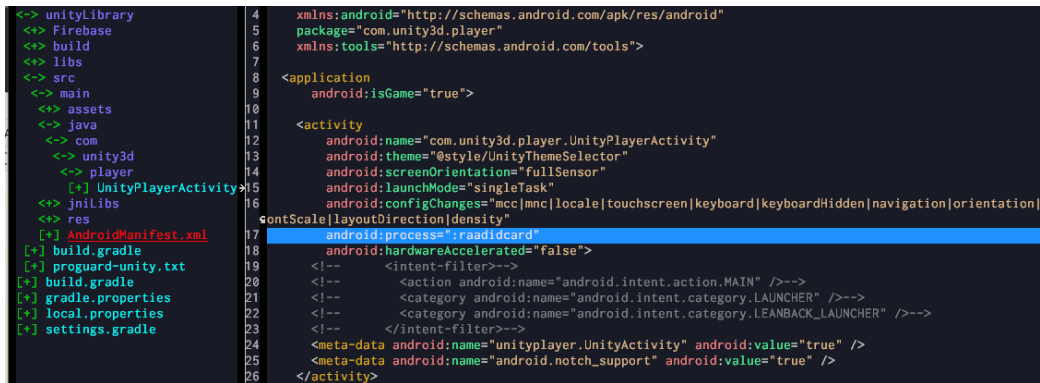
- 这里作些简单的版本修改适配自己的手机, 到项目可以构建成功为止.

3 Android 启动运行 unity

3.1 在 unity 的 AndroidManifest.xml 文件

- 把 <intent-filter>-> 删掉或者注释掉, 留着的话, 当我们把程序运行到手机或者模拟机上时会有两个图标。
- 其次是在 <activity> 里加入这行代码, 实现多线程, 避免在从 unity 返回 Android 时也将 Android 界面也结束了。

android:process=":raadidcard"



3.2 在 app 的 AndroidManifest.xml 文件里, 在图中位置加入这两行代码:

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

tools:replace="android:icon,android:theme,android:allowBackup"

- 可以成片复制的代码如下:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.unity3d.player">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        tools:replace="android:icon,android:theme,android:allowBackup"
        android:theme="@style/Theme.Test"
        tools:targetApi="31">

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
            </activity>

        </application>
    </manifest>
```

<pre> <-> app <-> build <-> libs <-> src <-> androidTest <-> main <-> java <-> res [+][AndroidManifest.xml] <-> test [+][build.gradle] <-> gradle <-> unityLibrary <-> Firebase <-> build <-> libs <-> src <-> main <-> assets <-> java <-> com <-> unity3d <-> player [+][UnityPlayerActivity] <-> jnilibs <-> res [+][AndroidManifest.xml] [+][build.gradle] [+][proguard-unity.txt] </pre>	<pre> 2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" 3 4 xmlns:tools="http://schemas.android.com/tools" 5 package="com.unity3d.player"> 6 7 <application 8 android:allowBackup="true" 9 android:dataExtractionRules="@xml/data_extraction_rules" 10 android:fullBackupContent="@xml/backup_rules" 11 android:icon="@mipmap/ic_launcher" 12 android:label="@string/app_name" 13 android:roundIcon="@mipmap/ic_launcher_round" 14 android:supportRtl="true" 15 tools:replace="android:icon,android:theme,android:allowBackup" 16 android:theme="@style/Theme.Test" 17 tools:targetApi="31"> 18 <activity 19 android:name=".MainActivity" 20 android:exported="true"> 21 <intent-filter> 22 <action android:name="android.intent.action.MAIN" /> 23 24 <category android:name="android.intent.category.LAUNCHER" /> 25 </intent-filter> 26 27 <meta-data 28 android:name="android.app.lib_name" 29 android:value="" /> 30 </activity> </pre>
---	--

3.3 在 app 的 build.gradle 里加入这行代码。

```

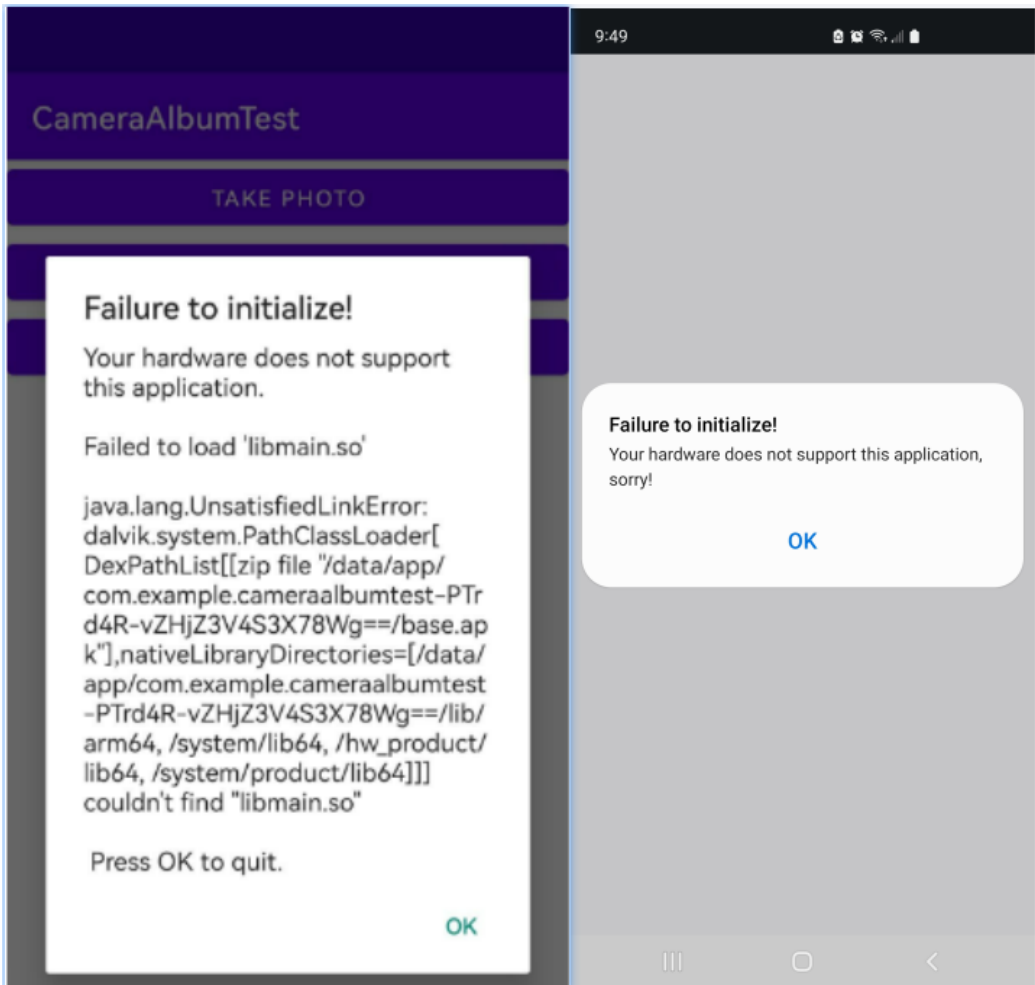
ndk {
    abiFilters 'armeabi-v7a'
}

```

<pre> <-> app <-> build <-> libs <-> src <-> androidTest <-> main <-> java <-> res [+][AndroidManifest.xml] <-> test [+][build.gradle] <-> gradle <-> unityLibrary <-> Firebase <-> build <-> libs <-> src <-> main <-> assets <-> java <-> com <-> unity3d <-> player [+][UnityPlayerActivity] <-> jnilibs <-> res [+][AndroidManifest.xml] </pre>	<pre> 2 id 'com.android.application' 3 } 4 5 android { 6 namespace 'com.unity3d.player' 7 compileSdk 32 8 9 defaultConfig { 10 applicationId "com.unity3d.player" 11 minSdk 25 12 targetSdk 31 13 versionCode 1 14 versionName "1.0" 15 16 ndk { 17 abiFilters 'armeabi-v7a' 18 } 19 20 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner" 21 } 22 23 buildTypes { 24 release { 25 minifyEnabled false 26 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro' 27 } 28 } </pre>
---	---

3.4 在 app 的 main->res->values->strings.xml 里加入这行代码

- 都还没有去想, 这句话能起到什么作用, 应该是关系不大, 或是可以跳过绕过的小细节
`<string name="game_view_content_description">Game view</string>`
- 进行这两步操作的原因是, 我在运行到手机时, 他显示硬件不支持或者闪退。加入上面两个代码后就可以正常启动 unity。
- 我个人认为真正起作用的是上一步关于手机架构的设置的 ndk 那三行, 与上面字符串无关, 应该是无关的



3.5 点击按钮启动 unity(画蛇添足)

- 感觉这个连接过程对于自己的项目就是画蛇添足. 可是如何既能避开这一步, 又能两者很好的平滑交互呢? 对于现在的自己, 是个问题和挑战
- 在主工程的 `activity_main.xml` 文件里添加一个按钮。 `MainActivity.java` 里加入启动事件, 如果在这里 `layout` 标红的话, 就把鼠标移到 `layout` 下面, 建立一个 `layout` 就行, 我分析是主工程的问题, 这个影响不大

```
<Button
    android:id="@+id/showUnityBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Show Unity"/>
```

```

f:/tmp/test/
<=> app
<=> build
<=> libs
<=> src
<=> androidTest
<=> main
<=> java
<=> res
    <=> drawable
    <=> drawable-v24
    <=> layout
    [+]activity_main.xml
    <=> mipmap-anydpi-v26
    <=> mipmap-hdpi
    <=> mipmap-mdpi
    <=> mipmap-xhdpi
    <=> mipmap-xxhdpi
    <=> mipmap-xxxhdpi
    <=> values
    [+]colors.xml
    [+]strings.xml
    [+]themes.xml
    <=> values-night
    <=> xml
    [+]AndroidManifest.xml
<=> test

1 <?xml version="1.0" encoding="utf-8"?>
2 <!-- <androidx.constraintlayout.widget.ConstraintLayout -->
3 <LinearLayout
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     xmlns:tools="http://schemas.android.com/tools"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context=".MainActivity">
10
11 <!-- <TextView -->
12 <!--     android:layout_width="wrap_content" -->
13 <!--     android:layout_height="wrap_content" -->
14 <!--     android:text="Hello World!" -->
15 <!--     app:layout_constraintBottom_toBottomOf="parent" -->
16 <!--     app:layout_constraintEnd_toEndOf="parent" -->
17 <!--     app:layout_constraintStart_toStartOf="parent" -->
18 <!--     app:layout_constraintTop_toTopOf="parent" /> -->
19
20 <Button
21     android:id="@+id/showUnityBtn"
22     android:layout_width="match_parent"
23     android:layout_height="wrap_content"
24     android:text="Show Unity"/>
25 </LinearLayout>
26 <!-- </androidx.constraintlayout.widget.ConstraintLayout -->
27

```

- MainActivity.cs 里的回调设置

```

Button btn = (Button)findViewById(R.id.showUnityBtn);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

```

```

// <----- UnityPlayerActivity <= com.unity3d.player 这里就是刚刚那个包名奇怪的地方，要不然 找不到 下面的 UnityPla
Intent intent = new Intent(MainActivity.this, UnityPlayerActivity.class); // <----- UnityPlayerAc

startActivity(intent);
}
});

```

```

f:/tmp/test/
<=> app
<=> build
<=> libs
<=> src
<=> androidTest
<=> main
<=> java
<=> com
    <=> unity3d
    <=> player
    [+]MainActivity.java
<=> res
    <=> drawable
    <=> drawable-v24
    <=> layout
    [+]activity_main.xml
    <=> mipmap-anydpi-v26
    <=> mipmap-hdpi
    <=> mipmap-mdpi
    <=> mipmap-xhdpi
    <=> mipmap-xxhdpi
    <=> mipmap-xxxhdpi
    <=> values
    [+]colors.xml
    [+]strings.xml
    [+]themes.xml

1 package com.unity3d.player; // <-----
2
3 import androidx.appcompat.app.AppCompatActivity;
4 import androidx.core.app.ActivityCompat;
5 import androidx.core.content.ContextCompat;
6 import androidx.core.os.BuildCompat;
7 import androidx.core.view.ViewCompat;
8 import androidx.core.widget.ButtonCompat;
9
10 public class MainActivity extends AppCompatActivity {
11     @Override protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         Button btn = (Button)findViewById(R.id.showUnityBtn);
16         btn.setOnClickListener(new View.OnClickListener() {
17             @Override
18             public void onClick(View view) {
19 // <----- UnityPlayerActivity <= com.unity3d.player 这里就是刚刚那个包名奇怪的地方，要不然 找不到 下面的 UnityPlayerActivity 类
20                 Intent intent = new Intent(MainActivity.this, UnityPlayerActivity.class); // <----- UnityPlayerAc
21
22                 startActivity(intent);
23             }
24         });
25     }
26 }
27

```

3.6 在 build.gradle 中申明包裹类名称

- 说是现在在 AndroidManifest.xml 里申明包裹名称已经过时了，要在配置文件里申明，于是我在这里申明的：

```

android {
    namespace 'com.unity3d.player'
}

```

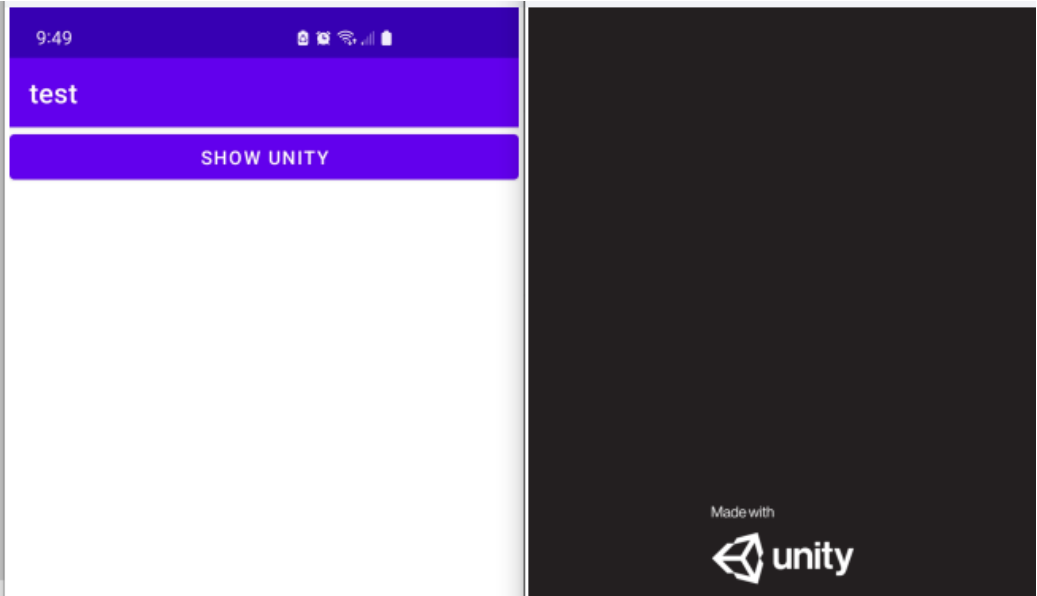


```

<-> unitylibrary
<-> Firebase
<-> libs
<-> src
<-> main
<-> assets
<-> java
<-> jnilibs
<-> res
[+] AndroidManifest.xml
[+] build.gradle
[+] proguard-unity.txt
[+] build.gradle
[+] gradle.properties
[+] local.properties
[+] settings.gradle
42 android {
43     namespace 'com.unity3d.player'
44
45     compileSdkVersion 32
46     buildToolsVersion '31.0.0'
47
48     compileOptions {
49         sourceCompatibility JavaVersion.VERSION_11
50         targetCompatibility JavaVersion.VERSION_11
51     }
52
53     defaultConfig {
54         minSdkVersion 31
55         targetSdkVersion 31
56         ndk {
57             abiFilters 'armeabi-v7a'

```

4 启动运行



5 Android Studio 类库中重复类的修复

```

Duplicate class com.airbnb.lottie.LottieAnimationView found in modules jetified-MiyataOpenUISdk-1.0.2-run
Duplicate class com.airbnb.lottie.LottieAnimationView$SavedState found in modules jetified-MiyataOpenUISdk-1.0.2-run
Duplicate class com.qq.e.ads.nativ.widget.NativeAdContainer found in modules jetified-GDTSdk.unionNormal
Go to the documentation to learn how to Fix dependency resolution errors.









```

- 如果新导入的依赖库发生了 Duplicate class android.xx.xx 这种类型的报错可能就是两个库导入了重复的类，这时候只需要把 build.gradle 中新导入的依赖做如下处理

```

implementation ('com.xxx.xxx.xx:xx:1.0.0'){
    exclude group: "com.xxxx.xxxx"
}

```
- 上面这个方法我还没有试，下面的试过了可行
- 对，就是把新导入的依赖库的后面加上大括号并把重复导入包名填入相应的位置就可以解决了，有时候可能会好几个依赖库都重复了，这就比较难判断了
- 1. 把 MiyataOpenUISdk-1.0.2.aar 改后缀成 zip，得到解压后的 MiyataOpenUISdk-1.0.2 文件夹，里面包含 classes.jar 和 res 等。

F (F:) > tmp > test > unityLibrary > libs > tmp > com.android.support.support-core-utils-26.1.0 >				
Name	^	Date modified	Type	Size
 aidl		9/13/2017 5:08 AM	File folder	
 assets		9/13/2017 5:09 AM	File folder	
 jni		9/13/2017 5:09 AM	File folder	
 libs		9/13/2017 5:09 AM	File folder	
 res		9/13/2017 5:08 AM	File folder	
 AndroidManifest.xml		9/13/2017 5:08 AM	XML File	2 KB
 classes.jar		9/13/2017 5:09 AM	Executable Jar File	93 KB
 R.txt		9/13/2017 5:08 AM	TXT File	5 KB

- 2. 同理把 classes.jar 改后缀成 zip，解压后得到 classes 文件夹，找到冲突的包，直接删除整个文件夹，如图
- 3. 使用 jar 命令重新对 classes 文件夹打包成 jar，并替换掉之前的 classes.jar。

```
jar cvf classes.jar -C classes/ .
```

- 4. 同理，使用 jar 命令重新对 MiyataOpenUISdk-1.0.2 文件夹打包成 aar，得到的 newMiyataOpenUISdk.aar 即可使用。

```
jar cvf com.android.support.support-compat-26.1.0.aar -C com.android.support.support-compat-26.1.0/ .
```

6 安卓 Android Studio 库包中有依赖的库包的解决方案 7.2.2

Direct local .aar file dependencies are not supported when building an AAR.

- 在高版本的 AndroidStudio 并且使用了版本的 gradle 出现了上述问题可以按着如下引用

6.1 比较好一点的, 是如下: 在项目的根目录的 build.gradle 里申明类库 unityLibrary 的依赖的文件路径就可找到

```
allprojects {
    buildscript {
        repositories {
            google()
            jcenter()
        }

        dependencies {
            classpath 'com.android.tools.build:gradle:7.2.2'
        }
    }

    repositories {
        google()
        jcenter()
        flatDir {
            dirs "${project(':unityLibrary').projectDir}/libs"
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

```

f:/tmp/test/
<=> app
<=> gradle
<=> unityLibrary
<=> unitylibs
[+] build.gradle
[+] gradle.properties
[+] local.properties
[+] settings.gradle

1 allprojects {
2     buildscript {
3         repositories {
4             google()
5             jcenter()
6         }
7     }
8     dependencies {
9         classpath 'com.android.tools.build:gradle:7.2.2'
10    }
11 }
12
13 repositories {
14     google()
15     jcenter()
16     flatDir {
17         dirs "${project(':unityLibrary').projectDir}/libs"
18     }
19 }
20 }
21
22 task clean(type: Delete) {
23     delete rootProject.buildDir
24 }
25
26 // Top-level build file where you can add configuration options common to all
27 // plugins {

f:/tmp/test/
<=> app
<=> build
<=> libs
<=> src
[+] build.gradle
<=> gradle
<=> unityLibrary
<=> build
<=> libs
<=> src
[+] build.gradle
[+] proguard-unity.txt
<=> unitylibs
[+] build.gradle
[+] gradle.properties
[+] local.properties
[+] settings.gradle

22 lintOptions {
23     abortOnError false
24 }
25
26
27 aaptOptions {
28     ignoreAssetsPattern = "!svn:!git:!ds_store:!*.scc:.*!CVS:!thumbs.db:!pico"
29 }
30
31 packagingOptions {
32     doNotStrip '*/armeabi-v7a/*.so'
33 }
34
35
36 dependencies {
37     implementation fileTree(dir: 'libs', include: ['*.jar'])
38 }
39

f:/tmp/test/
<=> app
<=> build
<=> libs
<=> src
[+] build.gradle
<=> gradle
<=> unityLibrary
<=> unitylibs
[+] build.gradle
[+] gradle.properties
[+] local.properties
[+] settings.gradle

17 ndk {
18     abiFilters 'armeabi-v7a'
19 }
20
21 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
22
23
24 buildTypes {
25     release {
26         minifyEnabled false
27         proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
28     }
29 }
30 compileOptions {
31     sourceCompatibility JavaVersion.VERSION_1_8
32     targetCompatibility JavaVersion.VERSION_1_8
33 }
34
35
36 dependencies {
37
38     implementation 'androidx.appcompat:appcompat:1.5.1'
39     implementation 'com.google.android.material:material:1.7.0'
40     implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
41
42     implementation project(path: ':unityLibrary')
43     testImplementation 'junit:junit:4.13.2'

```

6.2 下面的只是一种解决方案,可能还不是很好

6.3 在你工程根目录下新建一个文件夹 **unitylibs** , 将你的 **aar** 文件放入, 然后在该目录下新建一个 **build.gradle** 文件

```
f:/tmp/test/
<> app
<> gradle
<> unityLibrary
<> unitylibs
[+] build.gradle
[-] build.gradle
[+] gradle.properties
[+] local.properties
[+] settings.gradle

configurations.maybeCreate("default")
artifacts.add("default", file('android.arch.lifecycle.runtime-1.0.0.aar'))
artifacts.add("default", file('com.android.support.support-compat-26.1.0.aar'))
artifacts.add("default", file('com.android.support.support-core-ui-26.1.0.aar'))
artifacts.add("default", file('com.android.support.support-core-utils-26.1.0.aar'))
artifacts.add("default", file('com.android.support.support-fragment-26.1.0.aar'))
artifacts.add("default", file('com.android.support.support-media-compat-26.1.0.aar'))
artifacts.add("default", file('com.android.support.support-v4-26.1.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-ads-identifier-16.0.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-base-16.0.1.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-basement-16.2.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-flags-16.0.1.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-measurement-16.5.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-measurement-api-16.5.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-measurement-base-16.5.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-measurement-impl-16.5.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-measurement-sdk-16.5.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-measurement-sdk-api-16.5.0.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-stats-16.0.1.aar'))
artifacts.add("default", file('com.google.android.gms.play-services-tasks-16.0.1.aar'))
artifacts.add("default", file('com.google.firebase.firebase-analytics-16.5.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-app-unity-6.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-auth-17.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-auth-interop-17.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-auth-unity-6.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-common-17.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-database-17.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-database-collection-16.0.1.aar'))
artifacts.add("default", file('com.google.firebase.firebase-database-unity-6.0.0.aar'))
artifacts.add("default", file('com.google.firebase.firebase-iid-17.1.2.aar'))
artifacts.add("default", file('com.google.firebase.firebase-iid-interop-16.0.1.aar'))
artifacts.add("default", file('com.google.firebase.firebase-measurement-connector-17.0.1.aar'))
artifacts.add("default", file('UnityAds.aar'))
artifacts.add("default", file('UnityAdsAndroidPlugin.aar'))
```

6.4 在 **settings.gradle** 导入该工程

include ':unitylibs

```
f:/tmp/test/
<> app
<> gradle
<> unityLibrary
<> unitylibs
[+] build.gradle
[-] build.gradle
[+] gradle.properties
[+] local.properties
[+] settings.gradle

1 pluginManagement {
2     repositories {
3         gradlePluginPortal()
4         google()
5         mavenCentral()
6     }
7 }
8 dependencyResolutionManagement {
9     repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
10    repositories {
11        google()
12        mavenCentral()
13    }
14 }
15 rootProject.name = "test"
16 include ':app'
17 include ':unityLibrary'
18 include ':unitylibs'
```

6.5 在你需要依赖的工程里面的 **build.gradle** 中增加依赖

- // 这里需要注意的是, unitylibs 是你 aar 库所在文件夹

```
implementation project(path: ':unitylibs')
```

```

27 // implementation(name: 'com.google.firebase.firebase-app-unity-6.0.0', ext:'aar')
28 // implementation(name: 'com.google.firebase.firebase-auth-17.0.0', ext:'aar')
29 // implementation(name: 'com.google.firebase.firebase-auth-interop-17.0.0', ext:'aar')
30 // implementation(name: 'com.google.firebase.firebase-auth-unity-6.0.0', ext:'aar')
31 // implementation(name: 'com.google.firebase.firebase-common-17.0.0', ext:'aar')
32 // implementation(name: 'com.google.firebase.firebase-database-17.0.0', ext:'aar')
33 // implementation(name: 'com.google.firebase.firebase-database-collection-16.0.1', ext:'aar')
34 // implementation(name: 'com.google.firebase.firebase-database-unity-6.0.0', ext:'aar')
35 // implementation(name: 'com.google.firebase.firebase-iid-17.1.2', ext:'aar')
36 // implementation(name: 'com.google.firebase.firebase-iid-interop-16.0.1', ext:'aar')
37 // implementation(name: 'com.google.firebase.firebase-measurement-connector-17.0.1', ext:'aar')
38 // implementation(name: 'UnityAds', ext:'aar')
39 // implementation(name: 'UnityAdsAndroidPlugin', ext:'aar')
40
41 implementation fileTree('Firebase')
42 implementation project(path: ':unitylibs')
43

```

- 当然如果你有很多 aar 库，那么你需要在根目录创建一个 LocalRepo 目录，然后将你不同的 aar 库放在不同文件夹下。在 setting.gradle 分别导入
- 下面它是这么说的, 可是我都把它们放在同一个类库里, 看不行的话再移. 为什么每个包都需要一个单独的类库呢? 解偶多个不同包之间的依赖性? 加载时的内存性能影响等?

7 那么现在就是说: 安卓 SDK 与 unity 的交互与打包基本没有问题了

- This PC's S10+storage.defaultcompany.trunk



SCORE LEVEL LINE
0 1 0



SAVE GAME

IN

RESUME GAME

TUTORIAL

BACK TO MAIN
MENU

CREDIT

- 但对自己更大的挑战是: 为什么 unity 里一个空物件挂载到热更新的过程, 我打包之后在安卓手机上运行不出来, 仍需要时间 debug 这个过程 (呵呵, 前面昨天还是前天已经想到问题的原因, 不到因为探讨其它的想法, 直到今天傍晚刚才整个过程才理清. 不过目前仍是用 unity 直接到包, 还有许多其它的细节小问题需要解决)
- 过程中遇到过, 还会遇到很多不懂的问题, 比如同样的某些 android studio 里加 android:exported="true" 各种标签等, 如果只用 unity 打包, 该如何实现呢? 两套不同的打包机制都得弄明白. 但都是这么一个学习的过程, 不会被轻易挫败.
- 相比之下, 安卓 SDK 的实现极其简单, 可以放在后面

7.1 FATAL EXCEPTION: main

```

8943 8943 E AndroidRuntime: FATAL EXCEPTION: main
8943 8943 E AndroidRuntime: Process: com.defaultcompany.trunk, PID: 8943
8943 8943 E AndroidRuntime: java.lang.RuntimeException: Unable to destroy activity [com.defaultcompany.trunk/com.defaultcompany.trunk.UnityPlayerActivity]: java.lang.NullPointerException: Attempt to invoke virtual method 'boolean android.os.Handler.sendMessage(android.os.Message)' on a null object reference
8943 8943 E AndroidRuntime: at android.app.ActivityThread.performDestroyActivity(ActivityThread.java:5950)
8943 8943 E AndroidRuntime: at android.app.ActivityThread.handleDestroyActivity(ActivityThread.java:5995)
8943 8943 E AndroidRuntime: at android.app.servertransaction.DestroyActivityItem.execute(DestroyActivityItem.java:47)
8943 8943 E AndroidRuntime: at android.app.servertransaction.ActivityTransactionItem.execute(ActivityTransactionItem.java:45)
8943 8943 E AndroidRuntime: at android.app.servertransaction.TransactionExecutor.executeLifecycleState(TransactionExecutor.java:176)
8943 8943 E AndroidRuntime: at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:97)
8943 8943 E AndroidRuntime: at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2438)
8943 8943 E AndroidRuntime: at android.os.Handler.dispatchMessage(Handler.java:106)
8943 8943 E AndroidRuntime: at android.os.Looper.loopOnce(Looper.java:226)
8943 8943 E AndroidRuntime: at android.os.Looper.loop(Looper.java:313)
8943 8943 E AndroidRuntime: at android.app.ActivityThread.main(ActivityThread.java:8663)
8943 8943 E AndroidRuntime: at java.lang.reflect.Method.invoke(Native Method)
8943 8943 E AndroidRuntime: at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:567)
8943 8943 E AndroidRuntime: at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1135)
8943 8943 E AndroidRuntime: Caused by: java.lang.NullPointerException: Attempt to invoke virtual method 'boolean android.os.Handler.sendMessage(android.os.Message)' on a null object reference
8943 8943 E AndroidRuntime: at android.os.Message.sendToTarget(Message.java:468)
8943 8943 E AndroidRuntime: at com.unity3d.player.UnityPlayer$a(Unknown Source:8)
8943 8943 E AndroidRuntime: at com.unity3d.player.UnityPlayer$a(Unknown Source:2)
8943 8943 E AndroidRuntime: at com.unity3d.player.UnityPlayer.quit(Unknown Source:28)
8943 8943 E AndroidRuntime: at com.defaultcompany.trunk.UnityPlayerActivity.onDestroy(UnityPlayerActivity.java:44)
8943 8943 E AndroidRuntime: at android.app.Activity.performDestroy(Activity.java:8571)
8943 8943 E AndroidRuntime: at android.app.Instrumentation.callActivityOnDestroy(Instrumentation.java:1364)
8943 8943 E AndroidRuntime: at android.app.ActivityThread.performDestroyActivity(ActivityThread.java:5937)
8943 8943 E AndroidRuntime: ... 5 more

```

- 这个没有再出现了, 根据这里改的:<https://forum.unity.com/threads/android-crashes-after-1126979/>
- 但是游戏的界面仍然是渲染不出来, 还在找原因

```

@Override protected void onDestroy () {
    Log.d(TAG, "onDestroy() ");
    // mUnityPlayer.destroy();
    mUnityPlayer.removeAllViews();
    mUnityPlayer.quit();
    super.onDestroy();
}

```

7.2 类库包里的错误的修复问题

- 现在还不是很懂, 或是还没有经历很好地锻炼怎么改类库包里的错误, 晚点儿再理会这些

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_enabled="false"
        android:color="@color/common_google_signin_btn_text_dark_disabled" />
    <item
        android:state_pressed="true"
        android:color="@color/common_google_signin_btn_text_dark_pressed" />
    <item
        android:state_focused="true"
        android:color="@color/common_google_signin_btn_text_dark_focused" />
    <item
        android:color="@color/common_google_signin_btn_text_dark_default" />
</selector>

```

ERROR: F:\tmp\test\unitylibs\build\transforms\ac63488beaabb860e19f1e3c851bb20a\transformed\com.google.android.gms.play-services-base-16.0.1\

- 先只把这些有错误的类库包不连上

8 安卓设备上资源包的存放位置, 以及是否本地存放有需要的资源包

Android:

```
Application.dataPath          /data/app/package name-1/base.apk
Application.streamingAssetsPath jar:file:///data/app/package name-1/base.apk!/assets
Application.tmporaryCachePath /storage/emulated/0/Android/data/package name/cache
Application.persistentDataPath /storage/emulated/0/Android/data/package name/files
```

看Android上的路径, 跟iOS有点类似, 简单说一下。Android的几个目录是apk程序包、内存存储(InternalStorage)和外部存储(ExternalStorage)目录。

apk程序包目录: apk的安装路径, /data/app/package name-n/base.apk, dataPath就是返回此目录。

内部存储目录: /data/data/package name-n/, 用户自己或其它app都不能访问该目录。打开会发现里面有4个目录 (需要root)

cache 缓存目录, 类似于iOS的Cache目录

databases 数据库文件目录

files 类似于iOS的Documents目录

shared_prefs 类似于iOS的Preferences目录, 用于存放常用设置, 比如Unity3D的PlayerPrefs就存放于此

外部存储目录: 在内置或外插的sd上, 用户或其它app都可以访问, 外部存储目录又分私有和公有目录。

公有目录是像DCIM、Music、Movies、Download这样系统创建的公共目录, 当然你也可以像微信那样直接在sd卡根目录创建一个文件夹。好处嘛, 就是卸载app数据依旧存在。

私有目录在/storage/emulated/n/Android/data/package name/, 打开可以看到里面有两个文件夹cache和files。为什么跟内部存储目录重复了? 这是为了更大的存储空间, 以防内存存储空间较小。推荐把不需要隐私的、较大的数据存在这里, 而需要隐私的或较小的数据存在内部存储空间。

下面是各路径对应的Java访问方法:

apk包内: AssetManager.open(String filename)

内部存储: context.getFilesDir().getPath() or context.getCacheDir().getPath()

外部存储: context.getExternalFilesDir(null).getPath() or context.getExternalCacheDir().getPath()

理解了Android存储的原理, 最后来说说开头提到的bug, Application.tmporaryCachePath/persistentDataPath返回空字符串。这其实因为权限的原因, app没有声明访问外部存储空间的权限, 但是Application.tmporaryCachePath/ Application.persistentDataPath却想返回外部存储的路径。这是Unity3D的bug, 没有权限本应该抛出异常或者错误, 让开发者知道原因。

经反复测试发现, 有【外置SD卡】的设备上, 如果声明读写外部存储设备的权限, 会返回外部存储路径, 不声明则会返回内部存储路径, 这样不会有问題。而在【无外置SD卡】的设备上, 不管是否声明读写外部存储设备的权限, Application.tmporaryCachePath/persistentDataPath都返回外部存储路径, 但是又没有权限, 就可能会导致返回null了, 之所以说可能是因为这个bug不是必现, 如果出现了设备重启之后就好了, 怀疑是linux设备mount问題。但是出了问题, 我们不能跟用户说你重启一下手机就好了。

```
1 | <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

```
Application.dataPath          /data/app/package name-1/base.apk
Application.streamingAssetsPath jar:file:///data/app/package name-1/base.apk!/assets
Application.tmporaryCachePath /storage/emulated/0/Android/data/package name/cache
Application.persistentDataPath/storage/emulated/0/Android/data/package name/files
```

- 看 Android 上的路径, 跟 iOS 有点类似, 简单说一下。Android 的几个目录是 apk 程序包、内存存储 (InternalStorage) 和外部存储 (ExternalStorage) 目录。
- apk 程序包目录:** apk 的安装路径, /data/app/package name-n/base.apk, dataPath 就是返回此目录。
- 内部存储目录:** /data/data/package name-n/, 用户自己或其它 app 都不能访问该目录。打开会发现里面有 4 个目录 (需要 root)
- cache 缓存目录, 类似于 iOS 的 Cache 目录
- databases 数据库文件目录
- files 类似于 iOS 的 Documents 目录

- `shared_prefs` 类似于 iOS 的 `Preferences` 目录，用于存放常用设置，比如 Unity3D 的 `PlayerPrefs` 就存放于此
- 外部存储目录: 在内置或外插的 `sd` 上，用户或其它 `app` 都可以访问，外部存储目录又分私有和公有目录。
- 公有目录是像 `DCIM`、`Music`、`Movies`、`Download` 这样系统创建的公共目录，当然你也可以像微信那样直接在 `sd` 卡根目录创建一个文件夹。好处嘛，就是卸载 `app` 数据依旧存在。
- 私有目录在 `/storage/emulated/n/Android/data/package name/`，打开可以看到里面有两个文件夹 `cache` 和 `files`。为什么跟内部存储目录重复了？这是为了更大的存储空间，以防内存存储空间较小。推荐把不需要隐私的、较大的数据存在这里，而需要隐私的或较小的数据存在内部存储空间。
- 下面是各路径对应的 Java 访问方法：
 - apk 包内: `AssetManager.open(String filename)`
 - 内部存储: `context.getFilesDir().getPath()` or `context.getCacheDir().getPath()`
 - 外部存储: `context.getExternalFilesDir(null).getPath()` or `context.getExternalCacheDir().getPath()`

理解了 Android 存储的原理,最后来说说开头提到的 bug, `Application temporaryCachePath/persistentDataPath` 返回空字符串。这其实因为权限的原因, `app` 没有声明访问外部存储空间的权限,但是 `Application temporaryCachePath/ ApplicationpersistentDataPath` 却想返回外部存储的路径。这是 Unity3D 的 bug,没有权限本应该抛出一个异常或者错误,让开发者知道原因。

- 经反复测试发现,有【外置 SD 卡】的设备上,如果声明读/写外部存储设备的权限,会返回外部存储路径,不声明则会返回内部存储路径,这样不会有问题。而在【无外置 SD 卡】的设备上,不管是否声明读/写外部存储设备的权限, `Application temporaryCachePath/persistentDataPath` 都返回外部存储路径,但是又没有权限,就可能会导致返回 `null` 了,之所以说可能是因为这个 bug 不是必现,如果出现了设备重启之后就好了,怀疑是 linux 设备 `mount` 问题。但是出了问题,我们不能跟用户说你重启一下手机就好了。

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```