1.背景

```
jieou@gxatek-fw-no:/work/jieou/gxa_gos2$ du -h output/
13G output/
```

从上面可以看到，当前车机系统一个整包大小为13G多，意思是用户如果需要OTA升级整包的话，需要下载13G多的包，一方面车机后台静默下载系统包流量和速度都是不小的考验；另一方面，一次完整系统升级需要半个多小时，如果用户有感的话，那肯定是煎熬，再则APK 的大小会影响应用加载速度、使用的内存量以及消耗的电量。从系统整体来看，主要是APK大小和各个资源文件最占磁盘。本文主要从APK瘦身角度，分析哪些工具可以检测APK可优化点，APK瘦身的方向有哪些。

2.APK结构

在讨论如何缩减应用的大小之前，有必要了解下应用 APK 的结构。APK 文件由一个 Zip 压缩文件组成，其中包含构成应用的所有文件。这些文件包括 Java 类文件、资源文件和包含已编译资源的文件。

APK 包含以下目录：

- `META-INF/`：包含 `CERT.SF` 和 `CERT.RSA` 签名文件，以及 `MANIFEST.MF` 清单文件。
- `assets/`：包含应用的资源；应用可以使用 `AssetManager` 对象检索这些资源。
- `res/`：包含未编译到 `resources.arsc` 中的资源（图片、音视频等）。
- `lib/`：包含特定于处理器软件层的已编译代码。此目录包含每种平台类型的子目录，如 `armeabi`、

  `armeabi-v7a`、`arm64-v8a`、`x86`、`x86_64` 和 `mips`。

APK 还包含以下文件。在这些文件中，只有 `AndroidManifest.xml` 是必需的.

- `resources.arsc`：包含已编译的资源。此文件包含 `res/values/` 文件夹的所有配置中的 XML 内容。打包工具会提取此 XML 内容，将其编译为二进制文件形式，并压缩内容。此内容包括语言字符串和样式，以及未直接包含在 `resources.arsc` 文件中的内容（例如布局文件和图片）的路径。
- `classes.dex`：包含以 Dalvik/ART 虚拟机可理解的 DEX 文件格式编译的类。
- `AndroidManifest.xml`：包含核心 Android 清单文件。此文件列出了应用的名称、版本、访问权限和引用的库文件。该文件使用 Android 的二进制 XML 格式。

3. Android Size Analyzer工具介绍

Android Size Analyzer 工具可让您轻松地发现和实施多种缩减应用大小的策略。它可以作为 Android Studio 插件或独立的 JAR 使用。

3.1 在 Android Studio 中使用 Android Size Analyzer

> ！！！！现在我看了好多Android Studio版本都在插件中找不到这个插件了，如果能下载到可以使用，如果下载不到，可以按照3.2使用命令行来检查。

您可以使用 Android Studio 中的插件市场下载 Android Size Analyzer 插件，如图 1 所示。要打开插件市场并安装该插件，请按以下步骤操作：

1. 依次选择 **File > Settings**（或在 Mac 上，依次选择 **Android Studio > Preferences**）。
2. 选择左侧面板中的 **Plugins** 部分。
3. 点击 **Marketplace** 标签。
4. 搜索“Android Size Analyzer”插件。
5. 点击分析器插件的 **Install** 按钮。

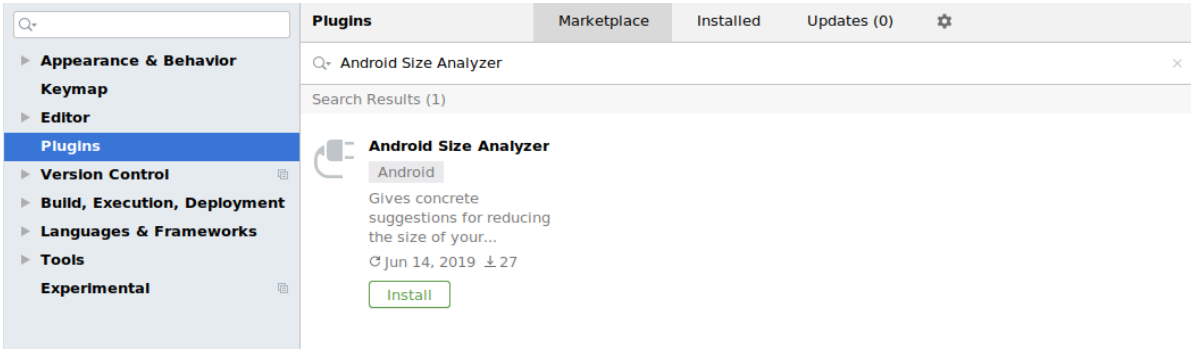


图 1. “Marketplace”标签中的“Android Size Analyzer”插件。

安装插件后，从菜单栏中依次选择 **Analyze > Analyze App Size**，对当前项目运行应用大小分析。分析了项目后，系统会显示一个工具窗口，其中包含有关如何缩减应用大小的建议，如图 2 所示。

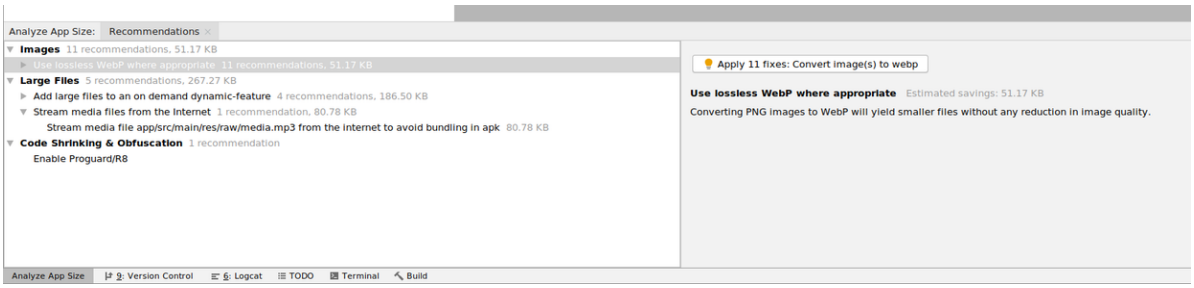


图 2. 包含建议的 Android Size Analyzer 插件工具窗口。

3.2 通过命令行使用分析器

您可以从 GitHub 以 TAR 或 ZIP 文件形式下载最新版本的 Android Size Analyzer。解压缩文件后，使用以下某个命令对 Android 项目或 Android App Bundle 运行 size-analyzer 脚本（在 Linux 或 MacOS 上）或 size-analyzer.bat 脚本（在 Windows 上）：

```
./size-analyzer check-bundle <path-to-aab>
./size-analyzer check-project <path-to-project-directory>
```

3.2.1 使用analyzer.jar

下载size-analyzer工程，使用Android Studio打开工程，然后使用gradlew来编译`analyzer.jar`.

```
./gradlew :analyzer:executableJar
```

编译完成之后，会在`analyzer\build\libs\`下生成analyzer.jar。

```
java -jar analyzer/build/libs/analyzer.jar check-bundle <path-to-aab>
java -jar analyzer/build/libs/analyzer.jar check-project <path-to-project-
directory>

# 加入 -d 参数可以直接输出优化建议
java -jar analyzer.jar check-project -d xlauncher/
```

```
优化建议如下
Converting images to webp 17.1 MiB
Convert submodule_map\common\src\main\res\drawable-mdpi\xunzhang_gq1.png to webp
with lossless encoding (saves 2.8 MiB)
Convert submodule_map\base\src\main\assets\xunzhang\fkscs.png to webp with
lossless encoding (saves 2.7 MiB)
Convert submodule_map\base\src\main\assets\xunzhang\mlbz.png to webp with
lossless encoding (saves 2.6 MiB)
Convert submodule_map\base\src\main\assets\xunzhang\qmslsj.png to webp with
lossless encoding (saves 2.3 MiB)
Convert submodule_map\mainmap\src\main\res\drawable-
mdpi\mainmap_food_theme_bg.png to webp with lossless encoding (saves 341.0 kiB)
Convert submodule_map\exploremap\src\main\res\drawable\icon_explore_map_back.png
to webp with lossless encoding (saves 290.4 kiB)
Convert app\src\main\res\drawable\ic_car.png to webp with lossless encoding
(saves 249.3 kiB)
Convert submodule_map\exploremap\src\main\res\drawable\icon_explore_map_back2.png
to webp with lossless encoding (saves 183.2 kiB)
Convert submodule_map\setting\src\main\res\drawable-
mdpi\setting_map_theme_set_bg.png to webp with lossless encoding (saves 154.5
kiB)
Convert submodule_map\setting\src\main\res\drawable-mdpi\setting_account_bg.png
to webp with lossless encoding (saves 152.7 kiB)
Convert app\src\main\res\drawable\eric.png to webp with lossless encoding (saves
135.4 kiB)
.......
# 检测到没有开启混淆和R8优化，后面拿launcher专门分析的时候会讲解到
It seems that you are not using Proguard/R8, consider enabling it in your
application. (saves )
```

可以看到如果将列出的图片从png转化成webp格式，可以节约**17.1M**空间

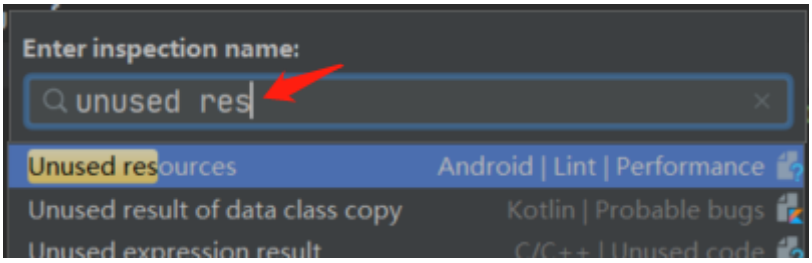
4 Gradle启动资源缩减

如果在应用的 build.gradle 文件中启用了资源缩减： `shrinkResources`，则 Gradle 在打包APK时可以自动忽略未使用资源。 资源缩减只有在与代码缩减： `minifyEnabled`配合使用时才能发挥作用。在代码缩减器移除所有不使用的代码后，资源缩减器便可确定应用仍要使用的资源 。

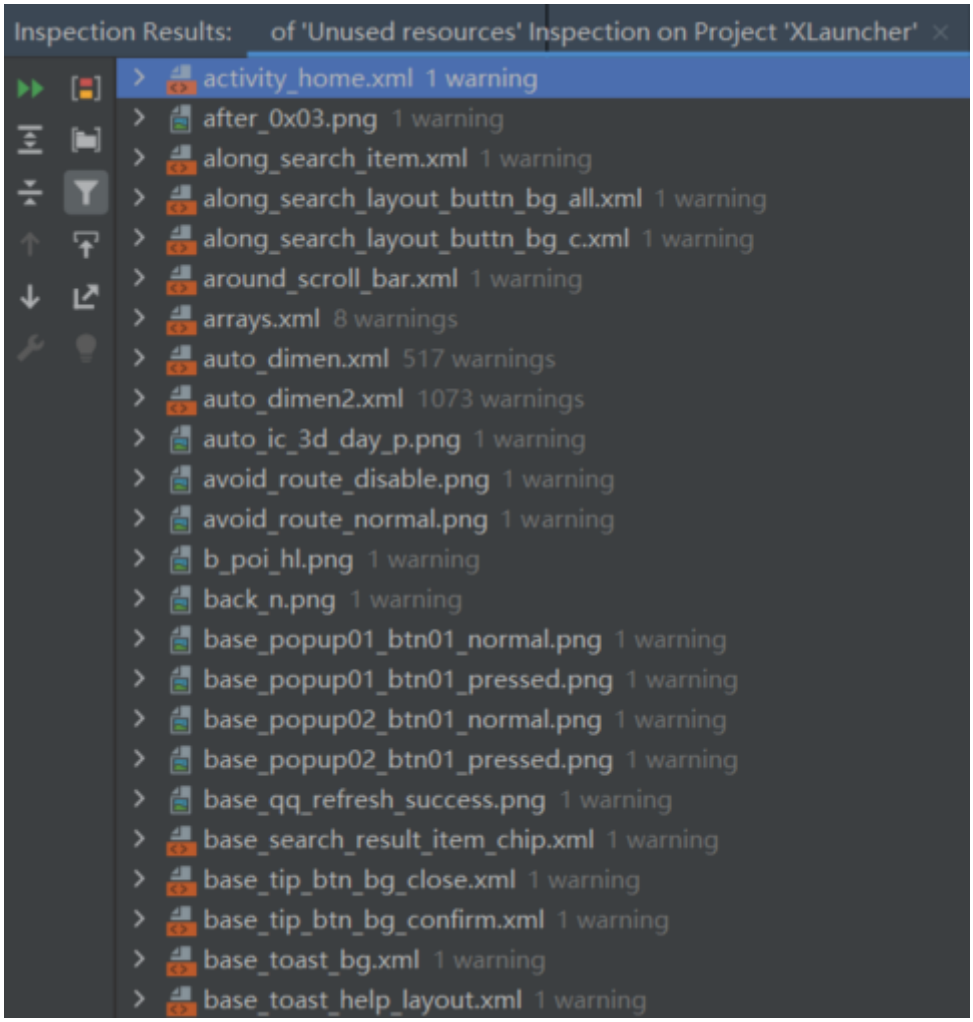
```
android {
    // Other settings
    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}
```

5 使用Lint分析器（物理删除）

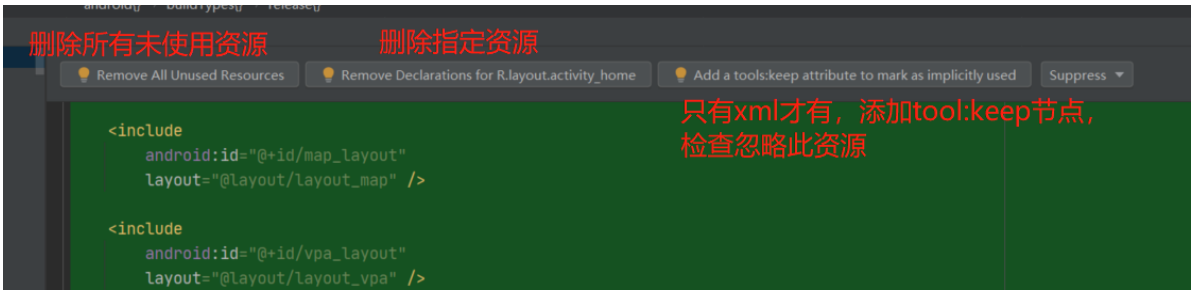
`lint` 工具是 Android Studio 中附带的静态代码分析器，可检测到`res/`文件夹中未被代码引用的资源。从菜单栏中依次选择 **Analyze > Run Inspection By Name**



lint分析完成之后，结果如下：



可以看出有大量png资源没有使用到。



lint 工具不会扫描 `assets/` 文件夹、通过反射引用的资源或已链接至应用的库文件。此外，它也不会移除资源，只会提醒您它们的存在。 **与资源缩减不同，这里点击删除，那就是把文件删了。**

反射引用资源：getResources().getIdentifier("layout_main","layout",getPackageName());

此处不建议点击直接 `删除所有未使用资源`，FO还是自己确认一下是否确实没有使用到。

5.1 一键删除无用资源

Android Studio给我们提供了一键移除所有无用的资源。从菜单栏中依次选择 **Refactor > Remove Unused Resources**，但是这种方式不建议使用，因为如果某资源仅存在动态获取资源id 的方式，那么这个资源会被认为没有使用过，从而会直接被删除。

5.2 自定义要保留的资源

如果有想要特别声明需要保留或舍弃的特定资源，在项目中创建一个包含 `<resources>` 标记的 XML 文件，并在 `tools:keep` 属性中指定每个要保留的资源，在 `tools:discard` 属性中指定每个要舍弃的资源。这两个属性都接受以逗号分隔的资源名称列表。还可以将星号字符用作通配符。

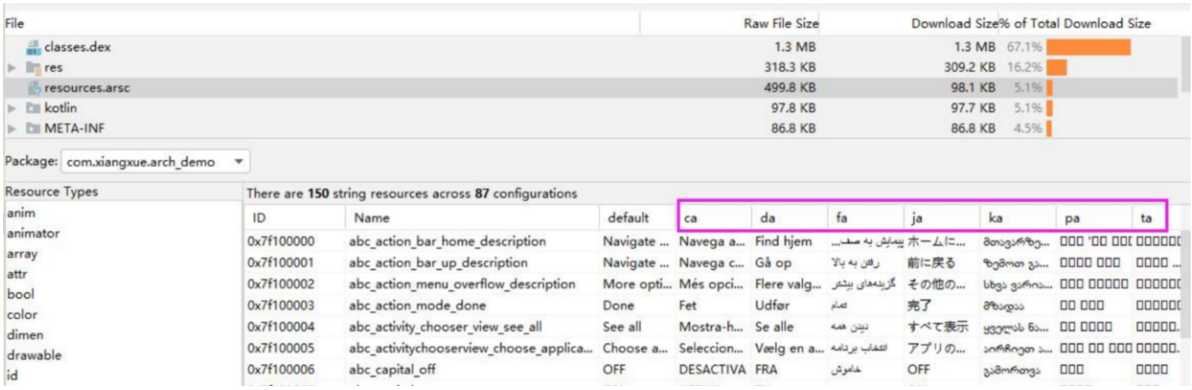
```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
    tools:keep="@layout/l_used*_c,@layout/l_used_a,@layout/l_used_b*"
    tools:discard="@layout/unused2" />
```

将该文件保存在项目资源中，例如，保存在 `res/raw/keep.xml` 中。构建系统不会将此文件打包到 APK 中。

5.3 移除未使用的备用资源

Gradle 资源缩减器只会移除未由应用代码引用的资源，这意味着，它不会移除用于不同设备配置的备用资源。可以使用 Android Gradle 插件的 `resConfigs` 属性移除应用不需要的备用资源文件。

例如，如果使用的是包含语言资源的库（如 AppCompat ），那么 APK 中将包含这些库中所有已翻译语言的字符串。如果只想保留应用正式支持的语言，则可以使用 `resConfig` 属性指定这些语言。系统会移除未指定语言的所有资源。




```
android {
    defaultConfig {
        ...
        resConfigs "zh-rCN"
    }
}
```

配置resConfifigs 只打包默认与简体中文资源。

6 动态库打包配置

so文件是由ndk编译出来的动态库，是 c/c++ 写的，所以不是跨平台的。ABI 是应用程序二进制接口简称（Application Binary Interface），定义了二进制文件（尤其是.so文件）如何运行在相应的系统平台上，从使用的指令集，内存对齐到可用的系统函数库。在Android 系统中，每一个CPU架构对应一个ABI，目前支持的有：`armeabi-v7a，arm64- v8a，x86，x86_64`。目前市面上手机设备基本上都是arm架构， armeabi-v7a 几乎能兼容所有设备。因此可以配置：

```
android{
    defaultConfig{
        ndk{
            abiFilters "armeabi-v7a"
        }
    }
}
```

但是！！！！但是！！！！但是！！！！针对我们当前车机，是`arm64-v8a`的，留有`arm64-v8a`就行，不要把支持`arm64-v8a`和`armeabi-v7a`的所有so库都打入到apk包中。

对于第三方服务，如百度地图、Bugly等会提供全平台的cpu架构。进行了上面的配置之后，表示只会把`armeabi-v7a`打包进入Apk。从而减少APK大小。

对于arm64架构的设备，如果使用armeabi-v7a也能够兼容，但是不如使用arm64的so性能。因此现在部分应用市场会根据设备提供不同架构的Apk安装。此时我们需要打包出针对arm64的apk与armv7a的apk，可以使用`productFlavor`。

```
flavorDimensions "default"
productFlavors{
    arm32 {
        dimension "default"
        ndk {
            abiFilters "armeabi-v7a"
        }
```

```
    }
    arm64 {
        dimension "default"
        ndk {
            abiFilters "arm64-v8a"
        }
    }
}
```

也可以使用，但不建议：

```
splits {
    abi {
        enable true
        reset()
        include 'arm64-v8a','armeabi-v7a'
        // exclude 'armeabi'
        universalApk true //是否打包一个包含所有so的apk
    }
}
```

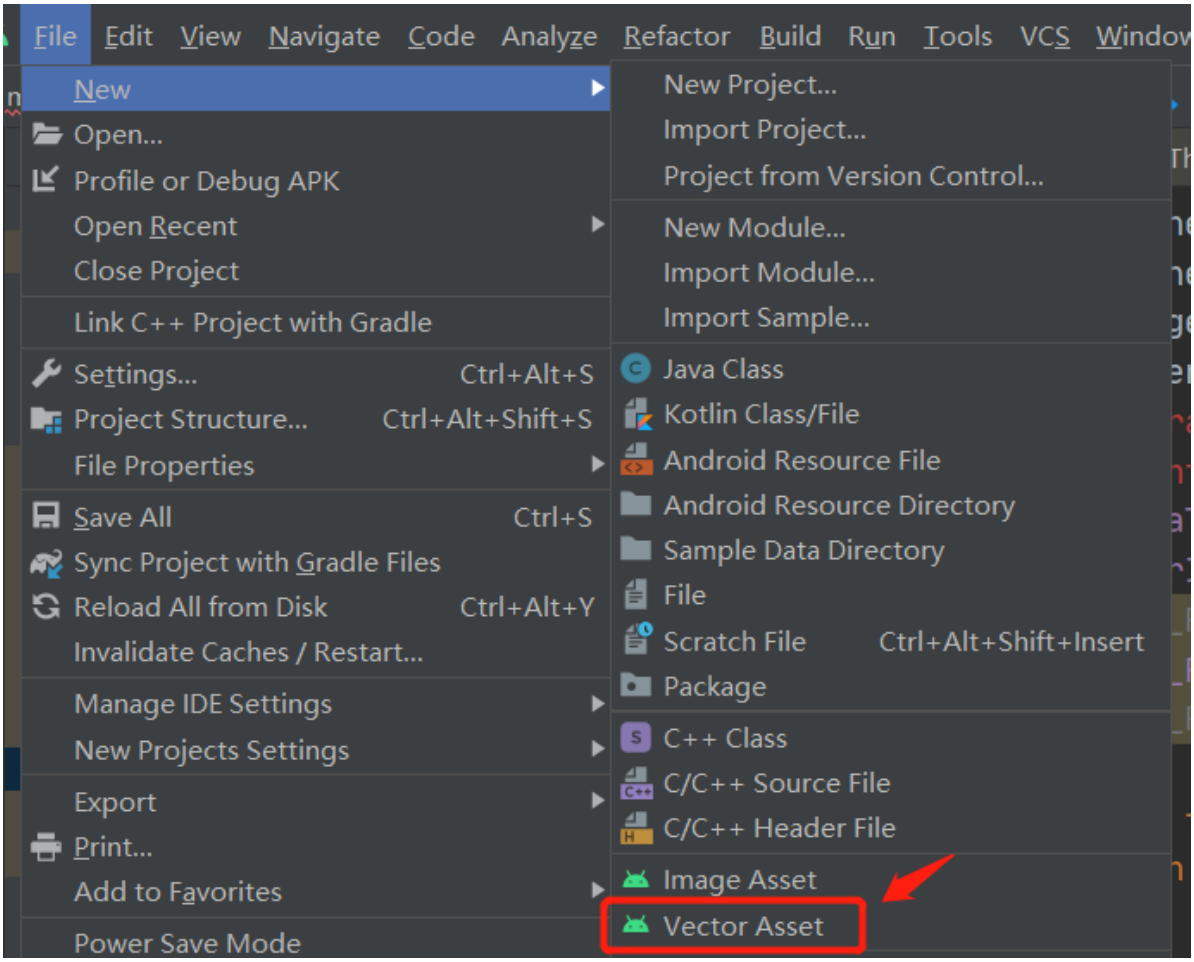
7 使用矢量图

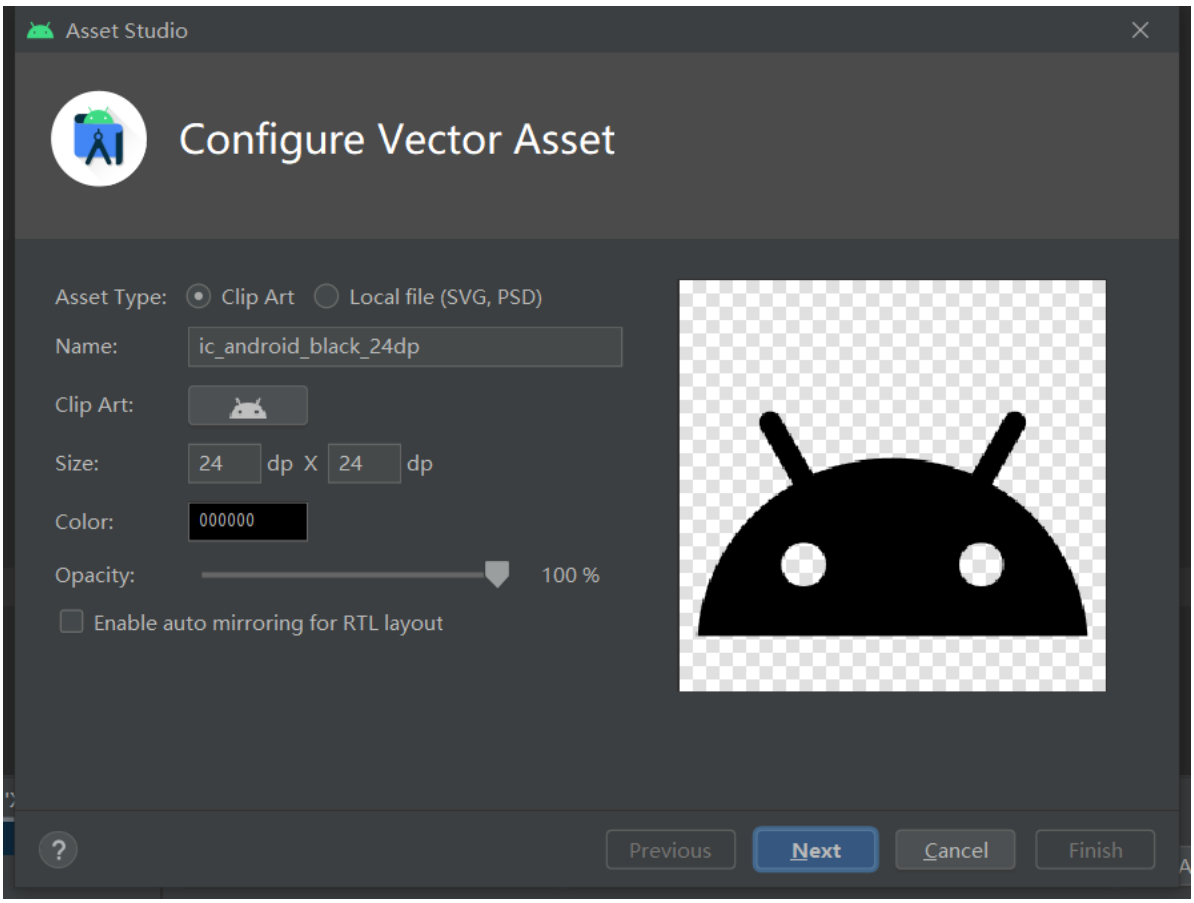
Apk中图片应该算是占用空间最多的资源。我们可以使用webp减少png、jpg图片占用空间的大小。对于小图标也可以使用矢量图。

矢量图可以创建与分辨率无关的图标和其他可伸缩媒体。使用这些图形可以极大地减少 APK 占用的空间。 矢量图片在 Android 中以 `VectorDrawable` 对象的形式表示。借助 `VectorDrawable` 对象，100 字节的文件可以生成与屏幕大小相同的清晰图片。

不过，系统渲染每个 VectorDrawable 对象需要花费大量时间，而较大的图片则需要更长的时间才能显示在屏幕上。因此，**建议仅在显示小图片时使用这些矢量图。**

新工程默认Icon就是矢量图。


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Windov
New
Open...
Profile or Debug APK
Open Recent
Close Project
Link C++ Project with Gradle
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Save All Ctrl+S
Sync Project with Gradle Files
Reload All from Disk Ctrl+Alt+Y
Invalidate Caches / Restart...
Manage IDE Settings
New Projects Settings
Export
Print...
Add to Favorites
Power Save Mode
New Project...
Import Project...
Project from Version Control...
New Module...
Import Module...
Import Sample...
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset


Asset Studio
Configure Vector Asset
Asset Type: Clip Art Local file (SVG, PSD)
Name: ic_android_black_24dp
Clip Art:
Size: 24 dp X 24 dp
Color: 000000
Opacity: 100 %
Enable auto mirroring for RTL layout
Previous
Next
Cancel
Finish

8 压缩PNG文件

`aapt` 工具可以在编译过程中通过无损压缩来优化放置在 `res/drawable/` 中的图片资源。例如，`aapt` 工具可以通过调色板将不需要超过 256 种颜色的真彩色 PNG 转换为 8 位 PNG。这样做会生成质量相同但内存占用量更小的图片。

请记住，`aapt` 具有以下限制：

- `aapt` 工具不会缩减 `asset/` 文件夹中包含的 PNG 文件。
- 图片文件需要使用 256 种或更少的颜色才可供 `aapt` 工具进行优化。
- `aapt` 工具可能会扩充已压缩的 PNG 文件。为防止出现这种情况，您可以在 Gradle 中使用 `cruncherEnabled` 标记为 PNG 文件停用此过程：

```
aaptOptions {
    cruncherEnabled = false
}
```

8.1 压缩 PNG 和 JPEG 文件工具

您可以使用 pngcrush、pngquant 或 zopflipng 等工具缩减 PNG 文件的大小，同时不损失画质。所有这些工具都可以缩减 PNG 文件的大小，同时保持肉眼感知的画质不变。

`pngcrush` 工具尤为有效：该工具会迭代 PNG 过滤器和 zlib (Deflate) 参数，使用过滤器和参数的每个组合来压缩图片。然后，它会选择可产生最小压缩输出的配置。

要压缩 JPEG 文件，您可以使用 packJPG 和 guetzli 等工具。

9 重复使用资源

现在我们有一个矢量图：

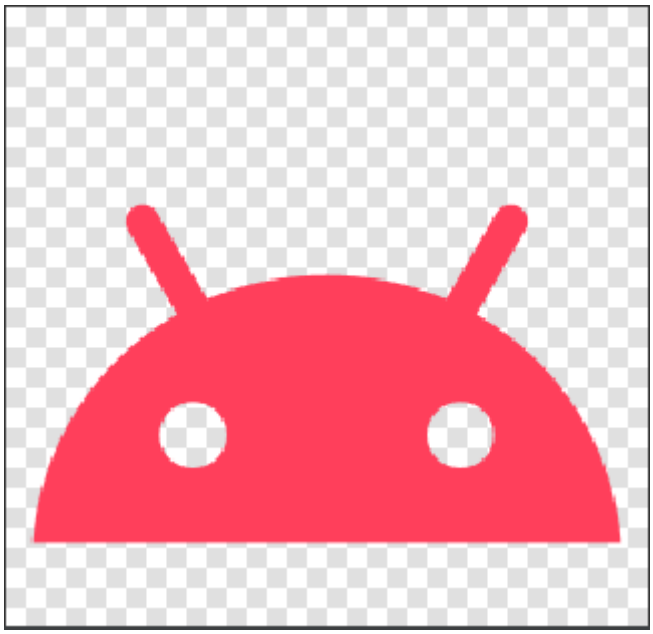
```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
    <path
    android:fillColor="@android:color/black"
    android:pathData="M10,20v-6h4v6h5v-8h3L12,3 2,12h3v8z"/>
</vector>
```

显示效果如下：

如果我们需要让矢量图显示红色怎么办？这种情况，我们不需要再去创建一个新的矢量图。可以直接给ImageView设置 `android:tint` 属性 来完成颜色的修改。

```
<ImageView
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:tint="@color/colorAccent"
    android:src="@drawable/tabbar_home_vector" />
```

效果如下：



10 选择器

如果需要让矢量图实现触摸变色。只需要创建selector，设置给tint即可。

```
<!-- tabbar_home_tint_selector -->
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/colorPrimary" android:state_pressed="true" />
    <item android:color="@color/colorAccent" />
</selector>
```

```
<ImageView
    android:clickable="true"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:src="@drawable/tabbar_home_vector"
    android:tint="@color/tabbar_home_tint_selector" />
```

阿里矢量图库

其他

- 使用精简版本的依赖：如protobuf-lite版本；对于分模块的库按需引入：如netty分模块引入；
- 主动移除无用代码（开启R8/Progurad自动移除）
- 避免使用枚举，使用 @IntDef 代替。
- 开启资源混淆：https://github.com/shwenzhang/AndResGuard
- 支付宝删除Dex debugItem https://juejin.im/post/6844903712201277448
- 对于发布Google paly的应用选择使用：AAB https://developer.android.google.cn/guide/app-bundle
- 避免解压缩原生库：在构建应用的发布版本时，您可以通过在应用清单的`<application>`元素中设置`android:extractNativeLibs="false"`，将未压缩的`.so`文件打包在 APK 中。停用此标记可防止`PackageManager`在安装过程中将`.so`文件从 APK 复制到文件系统，并具有减小应用更新的额外好处。使用 Android Gradle 插件 3.6.0 或更高版本构建应用时，插件会默认将此属性设为`"false"`。

参考文档

1.缩减应用大小 | Android 开发者 | Android Developers (google.cn)

2.开源工具android/size-analyzer: The size-analyzer is a tool to help developers find tips on how to reduce the size of their Android application. (github.com)