**Unity Android In-App Billing 实现&测试经历**

撰写时间: 2017/07/9, 有效期未知.

上周接到一个任务, 在一个Unity实现的小游戏里面添加上Android **内购**, 是的, 这款游戏本来只打算发布到iOS上的后来又决定在**Google** **Play** 上发布了.

Android **内购** 和 iOS 的**内购**应该是一样一样的吧, I think.

好吧, 废话不多说, 赶紧上车.

本质上来说, 我是那种头脑懒惰的人, 所以我选择先搜搜看, 有没有别人已经做好了的. 然后我就收到了一大堆, 深的浅的都有.

[Unity接入谷歌支付: http://www.jianshu.com/p/31ad2e3b3023](https://www.jianshu.com/p/31ad2e3b3023)

**[Unity3d接入](https://link.jianshu.com/?t=http://www.360doc.com/content/17/0404/14/40005136_642773257.shtml" \t "_blank)**[googleplay](https://link.jianshu.com/?t=http://www.360doc.com/content/17/0404/14/40005136_642773257.shtml" \t "_blank)**[内购](https://link.jianshu.com/?t=http://www.360doc.com/content/17/0404/14/40005136_642773257.shtml" \t "_blank)**[详细说明（二) http://www.360doc.com/content/17/0404/14/40005136\_642773257.shtml](https://link.jianshu.com/?t=http://www.360doc.com/content/17/0404/14/40005136_642773257.shtml" \t "_blank)

虽然这两篇博文是我能找到的最新的, 但是依然有点不符合我的条件. 第一篇吧, 不够详细(没有代码), 第二篇呢, 首先是Eclipse的工程, 其次, 如果你照着上面的敲, 最终会发现它的逻辑是有问题的(用户只要点击了某个商品的按钮, 就会得到该商品, 更本不用付钱啊) 补充说明下: 正常的逻辑是, 用户点击了某个商品的Button , 然后转一会圈, **Google** **Play**会返回该商品的信息并询问你是不是真的要购买, 如果你确定, 就可以点击 确定 按钮进行购买. 但是最头痛的是, 似乎找不到购买成功的回调方法, 后来我才发现其实这个代码也不是完整的.

Snip20170709\_9.png

**说明:**

在In-App Billing 中, 购买行为分3种

1. 购买消耗品. 比如游戏中购买金币, 钻石. 它的特点是, 用户可以多次购买, 因为用户会在游戏中慢慢的将金币/钻石消耗掉.
2. 购买非消耗品. 比如你在游戏中购买了一款游戏角色的皮肤. 那么这个皮肤就会一直存在. 而不会被销毁.
3. 购买订阅产品. 比如订阅报纸(这个我实在没怎么看到过, 所以就举个通俗的例子吧). 你需要用户定期缴费(有点像消耗品购买), 但是呢, 在一定期限内又要让用户不能重复购买(又像非消耗品的购买).

**说正事:**

当我发现这个不行的时候, 我就去看了看文档 [https://developer.android.com/**google**/**play**/billing/billing\_overview.html](https://link.jianshu.com/?t=https://developer.android.com/google/play/billing/billing_overview.html), 但是至少粗略的看了下, 发现文档里面写的和上面两篇博文的完全不同啊, 什么 ServiceConnection IInAppBillingService的怎么都没见过啊. 难道这文档是假的?

如果你只跟着上面两篇博文就想实现购买功能, 那么我只能告诉你 不可能. 幸运的是我跟着它们就实现了. Why? 因为我是通过Unity实现的. 但这不是重点, 重点是我在跟着上面走之前使用了一个插件 OpenIAB �0�9.

但是也不是完全相同, 因为在 OpenIAB 中有些用于**内购**的抽象类的实现方法和 官网上的Demo 中的工具类的实现方法略有不同.

当然, 这些只是我的经历而已, 你不用参考着实现, 后面我会详细的说明实现方法的.

好吧, 到这里我已经踩过许多坑, 比如在 **Google** Develop Console 中.

Snip20170709\_10.png

**解释下:**

我总共发布了4个App了

第一个为什么处于 已暂停 状态?  
我在测试时购买了些东西(是真的花了钱的, 这也是个悲催的故事), 然后**Google** 现在要我完善下信息, 绑定个可用收钱的账号, 把我测试时花费的钱收了.

第二个呢?  
这是个悲催的故事, 其实它是我在**Google** **Play** 创建的第一个应用, 然后, 如果你看过上面的两个博文, 有个博文中有一个设置 签名 的环节. 是的, 我衷心的提示你, 不要再 Unity 里面设置 签名, 你最好在 Android Studio 中设置签名, 并且 保存好 Key store 文件(.jks后缀的). 如果你在跟新该项目的时候发现这个文件不见了(而且没有 启用 **Google** **Play** App Signing, 那么恭喜你, 你有两种选择, 一 不升级; 二 取消发布. 但是可悲的是我启用了 **Google** **Play** App Signing 但是不知道怎么弄, 所以就这样了.

Snip20170709\_11.png

第三个  
这个不清楚为什么,但是一直放不上去, 我是将第四个项目 改了下 applicationId 发上去的, 然后就这样了.

第四个  
这个项目就是我做好了的Demo, 是不是觉得和上面的一个很像? 除了名字. 那么我需要告诉你个秘密. 不要把 Android 放在项目的前面. 如果你取得名字是这样的 Android IAB 测试项目 那么**Google**会毙了你的, 因为你在蹭它家的名气.

也许你会问, 为什么你是先讲坑, 而不是先把实现步骤讲讲, 然后再说遇到的坑呢? 我只想告诉你, 这只是我的经历, 这些不是真正我想要讲的.

其他的坑还有测试的坑, 是的, 一个大坑.

在iOS 上, 如果你要测试一个**内购**项目, 只需要在当前的应用下的 沙箱技术测试人员 里面创建一个假的 苹果账户进行测试就好了.所以在测试Android **内购**的时候, 我也理所当然的理解成在Beta测试版 的 管理测试人员 中添加的测试人员就是可以测试 **内购**的, 所以我在测试的时候, 点击了购买然后就真的花钱了, 起初我还期盼着这些钱会在之后的什么时候会打回给我. 然而..., 后再在文档的这个位置 [https://developer.android.com/**google**/**play**/billing/billing\_testing.html](https://link.jianshu.com/?t=https://developer.android.com/google/play/billing/billing_testing.html) , 仔细看了才知道还要设置个什么 许可测试.

**言归正传**

经过差不多一周的时间, 终于,还是实现了 In-App Billing的功能, 具体实现如下:

**In-App Billing实现**

**1. 在Google Play Console 创建应用**

1. 首先 你得保证自己有一个 **Google** **Play** 的开发者账号. 这是前提, 没有的自己百度.
2. 在 所有应用 界面 选择 创建应用, 并在弹出的对话框中写好名字(与**内购**无关的设置我也不会提的)

Snip20170709\_12.png

1. 应用 创建好之后, 就会跳转到商品详情页面, 让你去填写商品详情, 图片等一些东西, 确保图中几个都打上对勾了(我现在还没填写信息, 所以都是灰色的).

注意:  
有些必须要等到你上传了 APK 之后才能填写, 这个需要注意一下, 所以你可以先随便编译一个只要不报错的APK上去(先放在Alpha/Belta版, 封闭/开放测试都可以), 在真正需要发布的时候在升级一个新的APK替换掉就好了.

Snip20170709\_13.png

1. 编译并发布APK  
   4.1. 步骤如图:

Snip20170709\_15.png

在弹出的对话框中填入 点击 Next

4.2. 接着来到了这里

Snip20170709\_16.png

如果你之前没有build 过APK, 那么你就需要点击 Create New Key Store 如果自己做着玩的就随便填一下就好了, 公司的就认真填写. 但是需要注意的是, 哪些密码要找个地方记住 我这里已经build 过了 所以就直接输入之前的密码, 然后点击Next就好.

4.3. 如图:

Snip20170709\_17.png

我暂时还不知道这两个对勾是什么意思, 但是我在 build APK 的时候就勾上了, 所以如果你们知道,这两个表示什么意思的话可以留言告诉我下. 点击 Finish 就开始 创建带签名的 APK 了. 过一会你就可以在 Android Studio 的界面中看到一个 APK打包成功的弹框.点蓝色的字体就可以进去了(上面写着什么来着?不清楚了)

什么你刚才眼睛一闭一睁, 弹框不见了, 好吧, 告诉你在这里可以找到.

Snip20170709\_19.png

OK, 这样之后你就已经拿到签名了的APK了, 现在需要返回 **Google** Develop Console 把APK放上去了.

注意: 确保你的APK里面有结算权限 :

//在AndroidManifest.xml中声明权限：

<uses-permission android:name="com.android.vending.BILLING" />

1. 将APK放上去

Snip20170709\_20.png

后面的过程自己摸索摸索就能完成的, 所以我就不截图了. 但是记得设置测试方式以及测试人员  
APK放上去之后就可以继续 将一些需要先把APK放上去之后才能操作的步骤了.

1. 接着到 商店发布 下面的 应用内商品里面添加购买选项

Snip20170709\_21.png

1. 然后

Snip20170709\_22.png

注: 消耗品和非消耗品都属于受管理的商品, 还有就是 在填写好所有的信息之后, 记得在右上方,提交更新 的下面 激活一下该商品.

1. 这是最重要的一步, 曾经让我花了3$啊, 血的代价. 设置**内购**测试人员. 你是不是记得在上面上传APK的时候已经添加过一次测试人员了, 没错. 但那仅仅是有权安装测试版的App而已, 如果他们在你发布的应用中购买了东西, 那么他们是真的付了钱的. 添加**内购**测试人员的方法:  
   8.1 先跳到本文的第2张图  
   8.2 点击设置  
   8.3 如图:

Snip20170709\_23.png

多个测试账号之间用,隔开就好了.  
说明: 测试账号可以看到 免费说明

Snip20170709\_24.png

**实现过程中, 遇到的问题:**

1. 登录**Google**慢  
   这个可以翻墙, 我用的是蓝灯 [https://getlantern.org](https://link.jianshu.com/?t=https://getlantern.org) 已失效
2. 手机 谷歌服务框架的安装  
   我手机上有TapTap 然后在里面搜索 谷歌安装器 无需Root 用这个也可以安装 谷歌服务框架的.
3. 测试人员在 **Play** 商店上搜索不到 你发布的应用

Snip20170709\_25.png

把这个链接给测试人员(前提是这个测试人员的谷歌账户已经在你的测试人员上了, 并且如果要测试**内购**还要确保他在许可测试人员名单上哦,要不然是会扣钱的). 然后在链接画面中将App添加都心愿单中, 这下它就跑不掉了. 你就可以在 **Play** 商店上的 心愿单里找到它了.

1. 购买时发现 测试账号没有设置支付方式, 支付不成功  
   这也是个大坑, 真的. 我刚开始是想通过绑定VISA卡来做测试的, 但是发现在中国这个很难. 网上有方法,但是很麻烦, 会消耗好久的时间. 所以我后来是通过在付款方式 中兑换代码的方式来充钱的. 某宝上有 **Google** **Play** 兑换卡 购买的(直接在某宝上搜索 **google** **play** 礼品卡),价格也比较公道, 而且卖家会告诉你怎么设置的.

**项目中如何实现内购**

好吧现在开始讲解下如何在代码中实现**内购**

参考链接: [https://developer.android.com/training/in-app-billing/preparing-iab-app.html#AddLibrary](https://link.jianshu.com/?t=https://developer.android.com/training/in-app-billing/preparing-iab-app.html#AddLibrary)

**google** 提供的Demo链接: [https://github.com/googlesamples/android-**play**-billing](https://link.jianshu.com/?t=https://github.com/googlesamples/android-play-billing) 这个链接可以在上面的参考链接中找到.

然后用Android Studio 打开, 将下图对应的两个文件找到, 并把它们放入你自己的项目的对应位置中

记得将 util中的java文件 的包名改成你自己的包名

Snip20170709\_26.png

记得在自己项目的 AndroidManifest.xml文件中添加上**内购**许可  
<uses-permission android:name="com.android.vending.BILLING" />  
不知道在哪里加的可以参考上面的Demo里面是怎么加的

**内购的核心代码**

package com.game.tacker.iabdemo;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.IntentFilter;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.TextUtils;

import android.util.Log;

import android.view.View;

import android.widget.ImageView;

import com.game.tacker.iabdemo.util.IabBroadcastReceiver;

import com.game.tacker.iabdemo.util.IabBroadcastReceiver.IabBroadcastListener;

import com.game.tacker.iabdemo.util.IabHelper;

import com.game.tacker.iabdemo.util.IabHelper.IabAsyncInProgressException;

import com.game.tacker.iabdemo.util.Inventory;

import com.game.tacker.iabdemo.util.Purchase;

import com.game.tacker.iabdemo.util.IabResult;

import java.util.ArrayList;

import java.util.List;

public class MainActivity extends Activity implements IabBroadcastListener,

DialogInterface.OnClickListener {

// Debug tag, for logging

static final String TAG = "tackor";

// 是否已经购买了非消耗品

boolean mIsPremium = false;

// 是否已经订阅了无限燃油

boolean mSubscribedToInfiniteGas = false;

// 订阅是否自动续费

boolean mAutoRenewEnabled = false;

// Tracks the currently owned infinite gas SKU, and the options in the Manage dialog

String mInfiniteGasSku = "";

String mFirstChoiceSku = "";

String mSecondChoiceSku = "";

// Used to select between purchasing gas on a monthly or yearly basis

String mSelectedSubscriptionPeriod = "";

// SKUs for our products: the premium upgrade (non-consumable) and gas (consumable)

// 下面的四个 SKU 对应 **Google** Develop Console 里面你定义的商品的 ID, 如果你的是其他的, 可以在这里进行修改

static final String SKU\_PREMIUM = "premium"; // 非消耗品

static final String SKU\_GAS = "gas"; // 消耗品

// 订阅产品的 SKU (这里指 无限汽油)

static final String SKU\_INFINITE\_GAS\_MONTHLY = "infinite\_gas\_monthly"; //按月订阅产品

static final String SKU\_INFINITE\_GAS\_YEARLY = "infinite\_gas\_yearly"; //按年订阅产品

// (arbitrary) request code for the purchase flow

static final int RC\_REQUEST = 10001;

// Graphics for the gas gauge

// 表示不同油量的一组图片

static int[] TANK\_RES\_IDS = { R.drawable.gas0, R.drawable.gas1, R.drawable.gas2,

R.drawable.gas3, R.drawable.gas4 };

// 汽车的总血量

static final int TANK\_MAX = 4;

// 当前血量格数

int mTank;

// The helper object

IabHelper mHelper;

// Provides purchase notification while this app is running

IabBroadcastReceiver mBroadcastReceiver;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// load game data

loadData();

// PublicKey

String base64EncodedPublicKey = "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7BgPD8sssxklIEpM5j8iy3LfIYhLHwK4DcNJRHxb8UQTxOJ44fg/ef8omK7dPrjYDDp287fIWqTAR+WvWlHCY3BTtnlkQ0IMBlb2AT16ff4o7uYJ+VmRqFW+/OtpllGC08/uDhXrYFUJffQ2weJyHbiqJqE/NHpqSCr1MePqFBzkd9HmXfx7Dc/bcRo87Jn/zmKOOMNFZR+BCClX88zUvgX/FrGthDac3s2q/GsPWjojFaMvPbsy7z/9VCuOuqO56xzpOpOeSmnpaUKx6Pa8KxXCJm+XFxtXOjXfQAe/HHIHgWByRYRUTsl5B0uF82hT3T8hrNNfZfRxGM8GNaZ9DwIDAQAB";

// Create the helper, passing it our context and the public key to verify signatures with

Log.d(TAG, "Creating IAB helper.");

mHelper = new IabHelper(this, base64EncodedPublicKey);

// enable debug logging (for a production application, you should set this to false).

mHelper.enableDebugLogging(true);

// Start setup. This is asynchronous and the specified listener

// will be called once setup completes.

Log.d(TAG, "Starting setup.");

mHelper.startSetup(new IabHelper.OnIabSetupFinishedListener() {

public void onIabSetupFinished(IabResult result) {

Log.d(TAG, "Setup finished.");

if (!result.isSuccess()) {

// Oh noes, there was a problem.

complain("Problem setting up in-app billing: " + result);

return;

}

// Have we been disposed of in the meantime? If so, quit.

if (mHelper == null) return;

//动态创建并注册了一个广播

mBroadcastReceiver = new IabBroadcastReceiver(MainActivity.this);

IntentFilter broadcastFilter = new IntentFilter(IabBroadcastReceiver.ACTION);

registerReceiver(mBroadcastReceiver, broadcastFilter);

// IAB is fully set up. Now, let's get an inventory of stuff we own.

// 获取存货清单(应该是当前用户已经购买的存货清单)

Log.d(TAG, "Setup successful. Querying inventory.");

try {

mHelper.queryInventoryAsync(mGotInventoryListener);

} catch (IabAsyncInProgressException e) {

complain("Error querying inventory. Another async operation in progress.");

}

}

});

}

// Listener that's called when we finish querying the items and subscriptions we own

// 查询用户存货清单的回调方法

IabHelper.QueryInventoryFinishedListener mGotInventoryListener = new IabHelper.QueryInventoryFinishedListener() {

public void onQueryInventoryFinished(IabResult result, Inventory inventory) {

Log.d(TAG, "Query inventory finished.");

// Have we been disposed of in the meantime? If so, quit.

if (mHelper == null) return;

// Is it a failure?

if (result.isFailure()) {

complain("Failed to query inventory: " + result);

return;

}

Log.d(TAG, "Query inventory was successful.");

/\*

\* Check for items we own. Notice that for each purchase, we check

\* the developer payload to see if it's correct! See

\* verifyDeveloperPayload().

\*/

// Do we have the premium upgrade?

// 查询我们是否已经升级了车子, 也就是说查询我们是否已经购买了 SKU 为 SKU\_PREMIUM 的非消耗品

Purchase premiumPurchase = inventory.getPurchase(SKU\_PREMIUM);

mIsPremium = (premiumPurchase != null && verifyDeveloperPayload(premiumPurchase));

Log.d(TAG, "User is " + (mIsPremium ? "PREMIUM" : "NOT PREMIUM"));

// First find out which subscription is auto renewing

Purchase gasMonthly = inventory.getPurchase(SKU\_INFINITE\_GAS\_MONTHLY);

Purchase gasYearly = inventory.getPurchase(SKU\_INFINITE\_GAS\_YEARLY);

if (gasMonthly != null && gasMonthly.isAutoRenewing()) {

mInfiniteGasSku = SKU\_INFINITE\_GAS\_MONTHLY;

mAutoRenewEnabled = true;

} else if (gasYearly != null && gasYearly.isAutoRenewing()) {

mInfiniteGasSku = SKU\_INFINITE\_GAS\_YEARLY;

mAutoRenewEnabled = true;

} else {

mInfiniteGasSku = "";

mAutoRenewEnabled = false;

}

// The user is subscribed if either subscription exists, even if neither is auto

// renewing

mSubscribedToInfiniteGas = (gasMonthly != null && verifyDeveloperPayload(gasMonthly))

|| (gasYearly != null && verifyDeveloperPayload(gasYearly));

Log.d(TAG, "User " + (mSubscribedToInfiniteGas ? "HAS" : "DOES NOT HAVE")

+ " infinite gas subscription.");

if (mSubscribedToInfiniteGas) mTank = TANK\_MAX;

// Check for gas delivery -- if we own gas, we should fill up the tank immediately

Purchase gasPurchase = inventory.getPurchase(SKU\_GAS);

if (gasPurchase != null && verifyDeveloperPayload(gasPurchase)) {

Log.d(TAG, "We have gas. Consuming it.");

try {

mHelper.consumeAsync(inventory.getPurchase(SKU\_GAS), mConsumeFinishedListener);

} catch (IabAsyncInProgressException e) {

complain("Error consuming gas. Another async operation in progress.");

}

return;

}

updateUi();

setWaitScreen(false);

Log.d(TAG, "Initial inventory query finished; enabling main UI.");

}

};

// Callback for when a purchase is finished

IabHelper.OnIabPurchaseFinishedListener mPurchaseFinishedListener = new IabHelper.OnIabPurchaseFinishedListener() {

public void onIabPurchaseFinished(IabResult result, Purchase purchase) {

Log.d(TAG, "Purchase finished: " + result + ", purchase: " + purchase);

// if we were disposed of in the meantime, quit.

if (mHelper == null) return;

if (result.isFailure()) {

complain("Error purchasing: " + result);

setWaitScreen(false);

return;

}

if (!verifyDeveloperPayload(purchase)) {

complain("Error purchasing. Authenticity verification failed.");

setWaitScreen(false);

return;

}

Log.d(TAG, "Purchase successful.");

if (purchase.getSku().equals(SKU\_GAS)) { // 如果是消耗品, 那么就立即调用下面的方法进行消耗, 以便下次进行购买

// bought 1/4 tank of gas. So consume it.

Log.d(TAG, "Purchase is gas. Starting gas consumption.");

try {

mHelper.consumeAsync(purchase, mConsumeFinishedListener);

} catch (IabAsyncInProgressException e) {

complain("Error consuming gas. Another async operation in progress.");

setWaitScreen(false);

return;

}

}

else if (purchase.getSku().equals(SKU\_PREMIUM)) { //如果是非消耗品, 根据UI把蓝色按钮隐藏掉

// bought the premium upgrade!

Log.d(TAG, "Purchase is premium upgrade. Congratulating user.");

alert("Thank you for upgrading to premium!");

mIsPremium = true;

updateUi();

setWaitScreen(false);

}

else if (purchase.getSku().equals(SKU\_INFINITE\_GAS\_MONTHLY)

|| purchase.getSku().equals(SKU\_INFINITE\_GAS\_YEARLY)) {

// bought the infinite gas subscription

Log.d(TAG, "Infinite gas subscription purchased.");

alert("Thank you for subscribing to infinite gas!");

mSubscribedToInfiniteGas = true;

mAutoRenewEnabled = purchase.isAutoRenewing();

mInfiniteGasSku = purchase.getSku();

mTank = TANK\_MAX;

updateUi();

setWaitScreen(false);

}

}

};

// Called when consumption is complete

IabHelper.OnConsumeFinishedListener mConsumeFinishedListener = new IabHelper.OnConsumeFinishedListener() {

public void onConsumeFinished(Purchase purchase, IabResult result) {

Log.d(TAG, "Consumption finished. Purchase: " + purchase + ", result: " + result);

// if we were disposed of in the meantime, quit.

if (mHelper == null) return;

// We know this is the "gas" sku because it's the only one we consume,

// so we don't check which sku was consumed. If you have more than one

// sku, you probably should check...

if (result.isSuccess()) {

// successfully consumed, so we apply the effects of the item in our

// game world's logic, which in our case means filling the gas tank a bit

Log.d(TAG, "Consumption successful. Provisioning.");

mTank = mTank == TANK\_MAX ? TANK\_MAX : mTank + 1;

saveData();

alert("You filled 1/4 tank. Your tank is now " + String.valueOf(mTank) + "/4 full!");

}

else {

complain("Error while consuming: " + result);

}

updateUi();

setWaitScreen(false);

Log.d(TAG, "End consumption flow.");

}

};

/\*\* Verifies the developer payload of a purchase. \*/

// 如果公司自己有服务器, 那么可以在该方法中实现本地数据校验, 加强安全

boolean verifyDeveloperPayload(Purchase p) {

String payload = p.getDeveloperPayload();

return true;

}

// =================== 按钮的监听方法 =========================//

// User clicked the "Buy Gas" button

// 购买消耗品(汽油, 黄色)

public void onBuyGasButtonClicked(View arg0) {

Log.d(TAG, "Buy gas button clicked.");

if (mSubscribedToInfiniteGas) {

complain("No need! You're subscribed to infinite gas. Isn't that awesome?");

return;

}

if (mTank >= TANK\_MAX) {

complain("Your tank is full. Drive around a bit!");

return;

}

// launch the gas purchase UI flow.

// We will be notified of completion via mPurchaseFinishedListener

// 购买方法, 消耗品

setWaitScreen(true);

Log.d(TAG, "Launching purchase flow for gas.");

/\* TODO: for security, generate your payload here for verification. See the comments on

\* verifyDeveloperPayload() for more info. Since this is a SAMPLE, we just use

\* an empty string, but on a production app you should carefully generate this. \*/

String payload = "";

try {

mHelper.launchPurchaseFlow(this, SKU\_GAS, RC\_REQUEST,

mPurchaseFinishedListener, payload);

} catch (IabAsyncInProgressException e) {

complain("Error launching purchase flow. Another async operation in progress.");

setWaitScreen(false);

}

}

// User clicked the "Upgrade to Premium" button.

//购买方法, 非消耗品(蓝色)

public void onUpgradeAppButtonClicked(View arg0) {

Log.d(TAG, "Upgrade button clicked; launching purchase flow for upgrade.");

setWaitScreen(true);

/\* TODO: for security, generate your payload here for verification. See the comments on

\* verifyDeveloperPayload() for more info. Since this is a SAMPLE, we just use

\* an empty string, but on a production app you should carefully generate this. \*/

String payload = "";

try {

mHelper.launchPurchaseFlow(this, SKU\_PREMIUM, RC\_REQUEST,

mPurchaseFinishedListener, payload);

} catch (IabAsyncInProgressException e) {

complain("Error launching purchase flow. Another async operation in progress.");

setWaitScreen(false);

}

}

// "Subscribe to infinite gas" button clicked. Explain to user, then start purchase

// flow for subscription.

// 订阅商品(红色按钮)

public void onInfiniteGasButtonClicked(View arg0) {

if (!mHelper.subscriptionsSupported()) {

complain("Subscriptions not supported on your device yet. Sorry!");

return;

}

CharSequence[] options;

if (!mSubscribedToInfiniteGas || !mAutoRenewEnabled) {

// Both subscription options should be available

options = new CharSequence[2];

options[0] = getString(R.string.subscription\_period\_monthly);

options[1] = getString(R.string.subscription\_period\_yearly);

mFirstChoiceSku = SKU\_INFINITE\_GAS\_MONTHLY;

mSecondChoiceSku = SKU\_INFINITE\_GAS\_YEARLY;

} else {

// This is the subscription upgrade/downgrade path, so only one option is valid

options = new CharSequence[1];

if (mInfiniteGasSku.equals(SKU\_INFINITE\_GAS\_MONTHLY)) {

// Give the option to upgrade to yearly

options[0] = getString(R.string.subscription\_period\_yearly);

mFirstChoiceSku = SKU\_INFINITE\_GAS\_YEARLY;

} else {

// Give the option to downgrade to monthly

options[0] = getString(R.string.subscription\_period\_monthly);

mFirstChoiceSku = SKU\_INFINITE\_GAS\_MONTHLY;

}

mSecondChoiceSku = "";

}

int titleResId;

if (!mSubscribedToInfiniteGas) {

titleResId = R.string.subscription\_period\_prompt;

} else if (!mAutoRenewEnabled) {

titleResId = R.string.subscription\_resignup\_prompt;

} else {

titleResId = R.string.subscription\_update\_prompt;

}

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle(titleResId)

.setSingleChoiceItems(options, 0 /\* checkedItem \*/, this)

.setPositiveButton(R.string.subscription\_prompt\_continue, this)

.setNegativeButton(R.string.subscription\_prompt\_cancel, this);

AlertDialog dialog = builder.create();

dialog.show();

}

// Drive button clicked. Burn gas!

// 开车按钮(绿色的) 消耗燃油

public void onDriveButtonClicked(View arg0) {

Log.d(TAG, "Drive button clicked.");

if (!mSubscribedToInfiniteGas && mTank <= 0) alert("Oh, no! You are out of gas! Try buying some!");

else {

if (!mSubscribedToInfiniteGas) --mTank;

saveData();

alert("Vroooom, you drove a few miles.");

updateUi();

Log.d(TAG, "Vrooom. Tank is now " + mTank);

}

}

// ============== IabBroadcastListener 接口方法 ============//

@Override

public void onClick(DialogInterface dialogInterface, int id) {

if (id == 0 /\* First choice item \*/) {

mSelectedSubscriptionPeriod = mFirstChoiceSku;

} else if (id == 1 /\* Second choice item \*/) {

mSelectedSubscriptionPeriod = mSecondChoiceSku;

} else if (id == DialogInterface.BUTTON\_POSITIVE /\* continue button \*/) {

/\* TODO: for security, generate your payload here for verification. See the comments on

\* verifyDeveloperPayload() for more info. Since this is a SAMPLE, we just use

\* an empty string, but on a production app you should carefully generate

\* this. \*/

String payload = "";

if (TextUtils.isEmpty(mSelectedSubscriptionPeriod)) {

// The user has not changed from the default selection

mSelectedSubscriptionPeriod = mFirstChoiceSku;

}

List<String> oldSkus = null;

if (!TextUtils.isEmpty(mInfiniteGasSku)

&& !mInfiniteGasSku.equals(mSelectedSubscriptionPeriod)) {

// The user currently has a valid subscription, any purchase action is going to

// replace that subscription

oldSkus = new ArrayList<String>();

oldSkus.add(mInfiniteGasSku);

}

setWaitScreen(true);

Log.d(TAG, "Launching purchase flow for gas subscription.");

try {

mHelper.launchPurchaseFlow(this, mSelectedSubscriptionPeriod, IabHelper.ITEM\_TYPE\_SUBS,

oldSkus, RC\_REQUEST, mPurchaseFinishedListener, payload);

} catch (IabAsyncInProgressException e) {

complain("Error launching purchase flow. Another async operation in progress.");

setWaitScreen(false);

}

// Reset the dialog options

mSelectedSubscriptionPeriod = "";

mFirstChoiceSku = "";

mSecondChoiceSku = "";

} else if (id != DialogInterface.BUTTON\_NEGATIVE) {

// There are only four buttons, this should not happen

Log.e(TAG, "Unknown button clicked in subscription dialog: " + id);

}

}

@Override

public void receivedBroadcast() {

// Received a broadcast notification that the inventory of items has changed

Log.d(TAG, "Received broadcast notification. Querying inventory.");

try {

mHelper.queryInventoryAsync(mGotInventoryListener);

} catch (IabAsyncInProgressException e) {

complain("Error querying inventory. Another async operation in progress.");

}

}

// ================== 系统方法 ====================//

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

Log.d(TAG, "onActivityResult(" + requestCode + "," + resultCode + "," + data);

if (mHelper == null) return;

// Pass on the activity result to the helper for handling

if (!mHelper.handleActivityResult(requestCode, resultCode, data)) {

// not handled, so handle it ourselves (here's where you'd

// perform any handling of activity results not related to in-app

// billing...

super.onActivityResult(requestCode, resultCode, data);

}

else {

Log.d(TAG, "onActivityResult handled by IABUtil.");

}

}

// We're being destroyed. It's important to dispose of the helper here!

@Override

public void onDestroy() {

super.onDestroy();

// very important:

if (mBroadcastReceiver != null) {

unregisterReceiver(mBroadcastReceiver);

}

// very important:

Log.d(TAG, "Destroying helper.");

if (mHelper != null) {

mHelper.disposeWhenFinished();

mHelper = null;

}

}

// =============== 游戏相关方法 ==================//

// updates UI to reflect model

public void updateUi() {

// update the car color to reflect premium status or lack thereof

((ImageView)findViewById(R.id.free\_or\_premium)).setImageResource(mIsPremium ? R.drawable.premium : R.drawable.free);

// "Upgrade" button is only visible if the user is not premium

//如果用户已经购买了非消耗品, 那么就隐藏该按钮

findViewById(R.id.upgrade\_button).setVisibility(mIsPremium ? View.GONE : View.VISIBLE);

ImageView infiniteGasButton = (ImageView) findViewById(R.id.infinite\_gas\_button);

if (mSubscribedToInfiniteGas) {

// If subscription is active, show "Manage Infinite Gas"

infiniteGasButton.setImageResource(R.drawable.manage\_infinite\_gas);

} else {

// The user does not have infinite gas, show "Get Infinite Gas"

infiniteGasButton.setImageResource(R.drawable.get\_infinite\_gas);

}

// update gas gauge to reflect tank status

if (mSubscribedToInfiniteGas) {

((ImageView)findViewById(R.id.gas\_gauge)).setImageResource(R.drawable.gas\_inf);

}

else {

int index = mTank >= TANK\_RES\_IDS.length ? TANK\_RES\_IDS.length - 1 : mTank;

((ImageView)findViewById(R.id.gas\_gauge)).setImageResource(TANK\_RES\_IDS[index]);

}

}

// Enables or disables the "please wait" screen.

void setWaitScreen(boolean set) {

findViewById(R.id.screen\_main).setVisibility(set ? View.GONE : View.VISIBLE);

findViewById(R.id.screen\_wait).setVisibility(set ? View.VISIBLE : View.GONE);

}

void complain(String message) {

Log.e(TAG, "\*\*\*\* TrivialDrive Error: " + message);

alert("Error: " + message);

}

void alert(String message) {

AlertDialog.Builder bld = new AlertDialog.Builder(this);

bld.setMessage(message);

bld.setNeutralButton("OK", null);

Log.d(TAG, "Showing alert dialog: " + message);

bld.create().show();

}

void loadData() {

SharedPreferences sp = getPreferences(MODE\_PRIVATE);

mTank = sp.getInt("tank", 2);

Log.d(TAG, "Loaded data: tank = " + String.valueOf(mTank));

}

void saveData() {

SharedPreferences.Editor spe = getPreferences(MODE\_PRIVATE).edit();

spe.putInt("tank", mTank);

spe.apply();

Log.d(TAG, "Saved data: tank = " + String.valueOf(mTank));

}

}

如果觉得代码不够详细, 你可以自己下载 **Google**提供的Demo(上面有提供下载链接)自己实现一下. 如果发现我有什么地方理解错误的可以提出来一起探讨哦.