YRSDK 使用说明文档

Android-V2.2.2

目 录

1. 版本日志

版本号	版本发布日	变更说明	编写者	审核者
1.0.0	2018/6/5	1、文档创建 2、新增 Google 支付及跟踪。 3、增加 Facebook 登录。 4、增加接口类型及参数 5、横竖屏自适应 6、开放注册、登录、绑定 API。 7、实现自动登录流程 8、增加注销接口（清除当前角色信息）。 9、增加 Adjust 跟踪 10、兼容多语言及文字显示的适配 11、文档格式规范化。 12、修改部分接口类型及参数	曾林女	
2.0.0	2018/6/21	1、新增 AppsFlyer 跟踪 2、在 app/build.gradle 中加入 appsFlyer 的依赖：详情请看 3.2.2 3、在 Application 中进行配置 GameSDKConfig 时需要加入 AppsFlyer 的 dev key 的传值：详情请看 3.4 4、修复 FB 自动登录闪退的 bug 5、增加不同环境下的 IP 获取	曾林女	
2.1.0	2018/6/29	1、谷歌支付接口参数调整：详情请看 4.4	曾林女	
2.2.0	2018/7/9	1、将 SDKDemo 中 lib 下的 fastjson.jar 移到 YRSDK（library）下的 lib 中 2、避免签名打包后代码被混淆需要在接入项目中加入混淆代码。混淆代码文件详情请看：SDKDemo/ proguard-project.txt 3、备注将 library 的 AndroidManifest 和主程序的配置进行合并：详情请看 5.1 4、加入小钱包使用（当 APP 不能正常支付时会自动跳转到小钱包应用进行支付行为）	曾林女	

		5、onActivityResult 中代码更新小钱包支付回调结果：详情请看 3.5 6、（GameSDKConfig）初始配置数据中移除 appsFlyerDevKey 的传入：详情请看 3.3 7、调用支付接口时移除 canMakePayMent 的判断直接调用 buy 接口，详情请看 4.4.3 中的实现 8、当切换到 6 种语言外的语种时设置默认为英文语种		
2.2.1	2018/7/26	1、返回键拦截提示退出游戏对话框详情请看 4.9 2、混淆代码文件详情请看：SDKDemo/ proguard-project.txt 3、修改 fb 登录的 logInWithReadPermissions 权限参数 4、修改 FB 横屏时的登录框界面及取消屏幕触碰时取消登录逻辑 5、根据货币代码区分支付收据 6、增加初始化补单查询及登录打点	曾林女	
2.2.2	2018/7/28	1、优化悬浮球 2、修复充值掉单问题	曾林女	

2. 接入准备工作和先决条件

2.1. 接入准备工作

- 获取 library 工程 s4lib_FacebookSDK 和 YRSDK 导入 eclipse 并在项目中依赖这两个 library 工程
  注意：YRSDK 目录下中的 res 需要根据 lib（YRSDK-Vx.x.x.jar）版本来进行替换
- 获取 YRSDK V*.*.*为使用说明文档

- 获取配置文件
  AndroidManifest.xml
  将内容合并到自己工程的 AndroidManifest.xml 中；

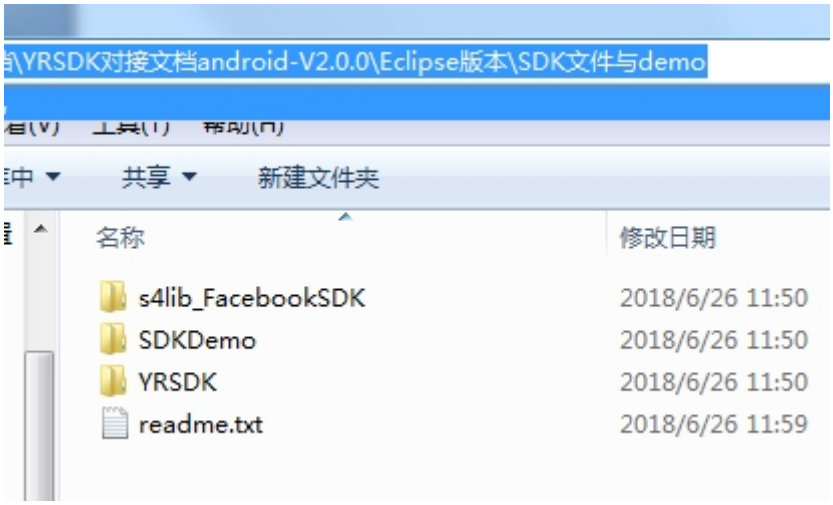
2.2. 使用条件及说明

YRSDK 里面的资源（res 包 drawle,id,string,color,layout 等）均以“yr_sdk”为开头的命名规则，所以希望用户在接入该平台时自己工程的其他资源命名不要 以
“yr_sdk”为开头，以避免资源命名冲突。
基于安卓 SDK 26.0.2(26)、Java JDK1.8 开发，支持 Android SDK4.0.3(15)及以上版本。

3. 工程配置

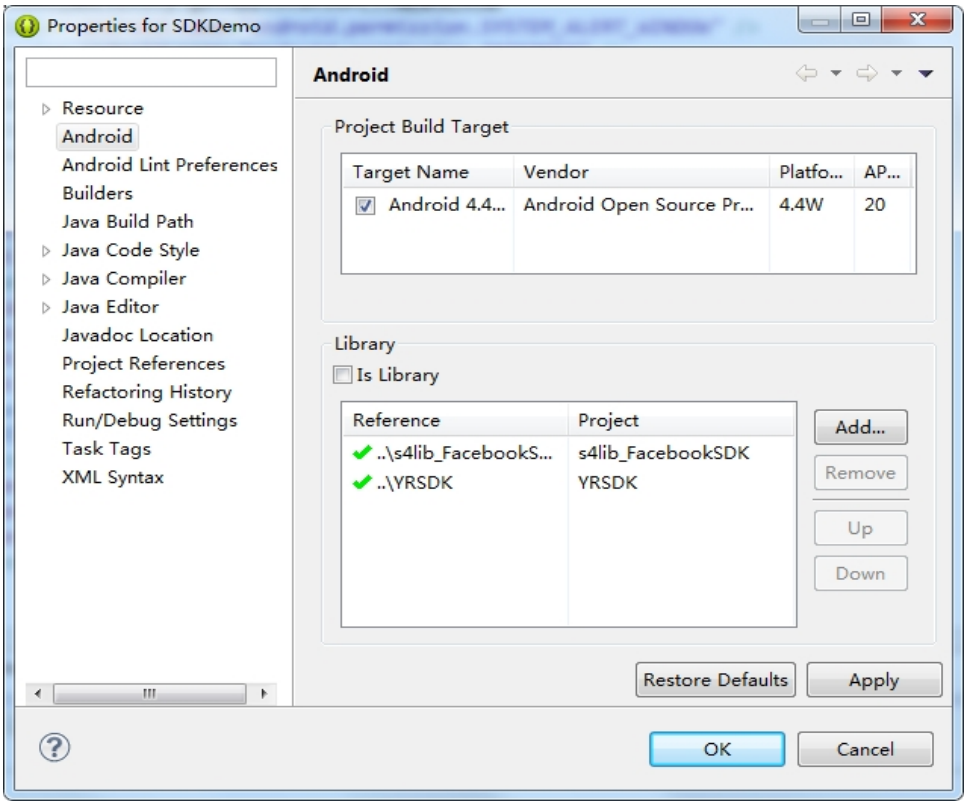
3.1. 导入 library

复制 s4lib_FacebookSDK 和 YRSDK library 工程导入 eclipse 工作空间,这两个工程请在 YRSDK 对接文档 android-Vxxx/Eclipse 版本/SDK 文件与 demo 目录下查找



3.1.1 添加 library

选择工程右击---Properties---Android--Add--将 s4lib_FacebookSDK 和 YRSDK library 进行依赖



3.2 游戏工程添加配置

3.2.1. app 项目中的 AndroidManifest.xml 配置

3.2.1.1. 加入 SDK 所需权限

```
<!--- GameSDK permission start-->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="com.android.vending.BILLING" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--- GameSDK permission end-->
```

3.2.1.2. appliction 语句块内配置

```
<application
android:allowBackup="true"
android:name="你创建的 Application"
android:supportsRtl="true">
<activity android:name="你的主入口 LauncherActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:screenOrientation="你屏幕的方向是横屏还是竖屏？ sensorLandscape or
portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
   </activity>

<!--YRSDK start-->
<!-- ### facebook beg ### -->
<activity
    android:name="com.facebook.FacebookActivity"
```

```
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screen
Size"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    tools:replace="android:configChanges,android:theme" />
<meta-data
    android:name="com.facebook.sdk.ApplicationId"
    android:value="@string/facebook_app_id" />
<!--      将意图筛选条件添加到 manifest 文件，将 fb_login_protocol_scheme 添加到
strings.xml 文件，
      启用 Chrome 自定义选项卡。启用 Chrome 自定义选项卡时，如果未安装 Facebook 应
用，
      SDK 会在 Chrome 自定义选项卡中展示“登录”对话框，而不是网页视图中。因此，
      如果用户已在 Chrome 浏览器中登录 Facebook，则无需再次输入凭证。-->
<activity
    android:name="com.facebook.CustomTabActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <!-- // if yourbApp ID is 1234567, you should use fb1234567
         <string name="fb_login_protocol_scheme">fbAPP_ID</string>-->
        <data android:scheme="@string/fb_login_protocol_scheme" />
    </intent-filter>
</activity>
<!-- ### facebook end ### -->
<!--YRSDK end-->


</application>
```

3.3. 游戏工程 Application 配置

如果游戏没有全局的 Application，请创建一个 Application,然后指入到 AndroidManifest.xml 配置中 application 语句块中的: **android:name="你创建的 Application"** 3.3.1.2

在游戏全局的 Appliction 中进行如下顺序配置：

```
public class GlobalGameAppliction extends Application {
private static final String TAG = GlobalGameAppliction.class.getSimpleName();
```

```
        @Override
    public void onCreate() {
        super.onCreate();

         //初始化上下文配置
        ContextUtil contextUtil = ContextUtil.getInstance();

          //传入 application
        contextUtil.setApplication(this);

        //设置全局上下文
        contextUtil.setApplicationContext(this.getApplicationContext());

         //日志模式
        YRGameSDKManager.setIsDebugModel(true); //发布时请记得关闭日志

         //加载配置
        GameSDKConfig instance = GameSDKConfig.getInstance();
         /** sdkAppId   平台分配的 appId
          *   sdkAppKey  平台分配的 appKey
          *   sdkMailingAddress  平台的请求地址
          *   fbApplicationId  平台分配的 facebookAPPID
          */
        instance.init(sdkAppId,sdkAppKey,fbApplicationId);

        //在主进程中进行初始化 SDK
        boolean isMainProcess = OSUtils.isMainProcess(this);
        SDKLoggerUtil.getLogger().e(TAG, "******* Application (%s) onCreate *******",
(isMainProcess ? "main" : "mult"));
        if (isMainProcess) {
            //初始化 SDK
             YRGameSDKManager.sdkInitialize(this);
        }
          ........
    }
}
```

3.4. LauncherActivity 中的配置

```
public class YouLauncherActivity extends AppCompatActivity {

    private static final String base64EncodedPublicKey = "分配的谷歌支付公钥，为了
安全应当从业务服下发获取"
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...

//检测平台初始化的校正
YRGameSDKManager.getInstance().initCorrectionSDK(this,
 new YRGameSDKManager.InitializeCallback() {
    @Override
    public void onInitialized(boolean status) {
        if (status) {
            /**
             * 检测是否可以启动自动登录界面
                  如果不能需要在点击登录 onclick 中调用
                    YRGameSDKManager.getInstance().openLoginView();
                手动开启登录界面
             */
            boolean canAutoLoginView =
           YRGameSDKManager.getInstance().isCanAutoLoginView();
            if (canAutoLoginView == false) {
                    .....
            }
        }
    }
});
//登录及注册回调
        YRGameSDKManager.getInstance().setSdkUserBehaviorListener(new
GameSDKUserBehaviorListener() {
            @Override
            public void onLogin(LoginResult result) {
                LoginType loginType = result.getLoginType();
                String str = "";
                if (loginType == LoginType.ACCOUNTS) {
                    str = "系统用户";
                } else if (loginType == LoginType.GUEST) {
                    str = "游客用户";
                } else if (loginType == LoginType.FACEBOOK) {
                    str = "facebook 用户";
                }
                if (result.getLoginStatus() == Status.LOGIN_TYPE_SUCCEED) {
                    //登录成功跳转到游戏
                      ...
                }
            }
```

```
            @Override
            public void onRegister(RegisterResult registerResult) {
                if (registerResult.getRegisterStatus() == Status.REGISTER_TYPE_SUCCEED) {
                    //注册成功跳转到游戏
                     ...
                }

            }
        });

/**
 * 初始化 google 支付并实例化接口
 */
GooglePurchaseUtil.getInstance().initPurchaseData(MainActivity.this,
base64EncodedPublicKey, new GameSDKPayListener() {
    @Override
    public void onPay(PayResult payResult) {
        Toast.makeText(MainActivity.this, "status:" + payResult.getPayStatus(),
Toast.LENGTH_SHORT).show();
         ...
    }
});
...
}
@Override
protected void onPause() {
    super.onPause();
    //释放网络加载弹框
    YRGameSDKManager.getInstance().dismissNetLoadDialog();
}
@Override
protected void onDestroy() {
    /**
     * 释放资源
     */
    YRGameSDKManager.getInstance().release();
    super.onDestroy();
}
}
```

3.5. 响应结果

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
//必须要在 super 前，加上下面这句，用于数据回调
  YRGameSDKManager.getInstance().onActivityResult(requestCode, resultCode,
data);

  super.onActivityResult(requestCode, resultCode, data);

  //小钱包支付回调
  if (requestCode == GameSDKConfig.TOP_UP_HELP_REQUEST_CODE && resultCode ==
GameSDKConfig.TOP_UP_HELP_RESULT_CODE) {
    if (data != null) {
        String payResultJson =
data.getStringExtra(GameSDKConfig.TOP_UP_HELP_RESULT_DATA_KEY);

        if (!Utility.isNullOrEmpty(payResultJson)) {
            PayResult payResult = (PayResult)
JSONUtil.jsonToObject(payResultJson, PayResult.class);
            if (payResult.getPayStatus() ==
GameSDKPayListener.PayStatus.SUCCEED) {

                //支付成功
                Toast.makeText(MainActivity.this, payResult.toString(),
Toast.LENGTH_LONG).show();

            }
        }

    }
}
  }
```

3.6. 注销用户

只需要调用 logout 方法即可

```
YRGameSDKManager.getInstance().logout();
```

4. SDK 客户端接口

[必须] 表示该接口必须被调用或使用
[按需求] 表示该接口按需求方的具体需求调用或使用
[按功能] 表示该接口可根据实际需要调用或使用

4.1. 用户登录[[必需]]

4.1.1. 接口说明

验证用户登录信息。目前支持：游客账号、YRSDK 平台 账号、Facebook。

4.1.2. 前置条件

SDK 初始化完成并已设置
YRGameSDKManager.getInstance().setSdkUserBehaviorListener(
GameSDKUserBehaviorListener)回调接口

4.1.3. 接口名称

YRGameSDKManager.getInstance().openLoginView()

4.1.4. 接口参数

无

4.1.5. 响应结果

登录成功后，通过 GameSDKUserBehaviorListener. onLogin(LoginResult)将用户信息回调给游戏

4.1.6. 特殊说明

无

4.2. 获取当前用户信息[[按功能]]

4.2.1. 接口说明

获取当前已登录玩家的信息。

4.2.2. 前置条件

登录成功或注册成功 。

4.2.3. 接口名称

YRGameSDKManager.getInstance().getUserInfo();

4.2.4. 接口参数

无

4.2.5. 响应结果

返回 UserInfo。包括：sid、name 等信息

4.2.6. 特殊说明

无

4.3. 欢迎回来[[按功能]]

4.3.1. 接口说明

一进入 app 时获取当前已登录玩家的信息进行判断是否有进行登录过。如果有则触发自动登录

4.3.2. 前置条件

登录成功或注册成功并已设置
YRGameSDKManager.getInstance().setSdkUserBehaviorListener(
GameSDKUserBehaviorListener)回调接口。

4.3.3. 接口名称

YRGameSDKManager.getInstance().isCanAutoLoginView();

4.3.4. 接口参数

无

4.3.5. 响应结果

登录成功后，通过 GameSDKUserBehaviorListener. onLogin(LoginResult)将用户信息回调给游戏

4.3.6. 特殊说明

该接口要在检测平台初始化的校正接口完成后调用:

```
//检测平台初始化的校正
YRGameSDKManager.getInstance().initCorrectionSDK(this,
 new YRGameSDKManager.InitializeCallback() {
     @Override
     public void onInitialized(boolean status) {
         if (status) {
             /**
              *检测是否可以启动自动登录界面
                    如果不能需要在点击登录 onclick 中调用
                      YRGameSDKManager.getInstance().openLoginView();
```

```
                手动开启登录界面
             */
            boolean canAutoLoginView =
YRGameSDKManager.getInstance().isCanAutoLoginView();
            if (canAutoLoginView == false) {
                ...
            }
        }
    }
});
```

4.4. Google 支付[[必需]]

4.4.1. 接口说明

在游戏商城中，玩家选择某商品并通过 Google 进行支付。

4.4.2. 前置条件

登录成功或切换账号成功
已设置当前用户的游戏信息
已实现 GameSDKPayListener，并复写 onPay(PayResult payResult)

4.4.3. 接口名称

初始化 google 支付：
GooglePurchaseUtil.getInstance().initPurchaseData(this,
base64EncodedPublicKey,GameSDKPayListener()

初始化成功后发起支付：

```
/**
 * * @param pid 就是后台对应的 productID
 * @param payloadCpOrderID 业务分配的订单号
 * @param _extral 额外参数
 */
    GooglePurchaseUtil.getInstance().buy(productID, payloadCpOrderID, extral);
```

4.4.4. 接口

4.4.4.1. initPurchaseData 接口

参数	类型	是否必填	说明
this	activity	是	当前 Activity
base64EncodedPublicKey	String	是	分配的谷歌支付公钥，为了安全应当从业务服下发获取
listener	GameSDKPayListener	是	支付结果回调

4.4.4.2. buy 接口

参数	类型	是否必填	说明
productID	String	是	谷歌后台中对应的商品 ID
payloadCpOrderID	String	是	游戏业务服生成的订单 ID 用于区分处理
extral	String	否	额外参数

4.4.5. 响应结果

结果将通过 GameSDKPayListener 返回给游戏

4.4.6. 特殊说明

初始化成功后发起支付时需要先判断是否可以正常开启支付

4.5. 悬浮按钮[[必需]]

4.5.1. 接口说明

用于显示悬浮按钮。

4.5.2. 前置条件

初始化及登录成功，并 CP 已经跟 SDK 后台校正成功 、只能在进入游戏后才能调用本接口

4.5.3. 接口名称

```
//显示
/**
 * @param roleId     角色 ID
 * @param serverId  区服 ID
 */
YRGameSDKManager.getInstance().openFloatView(String roleId, String serverId);
```

4.5.4. 接口参数

参数	类型	是否必填	说明
roleId	String	是	角色 ID
serverId	String	是	区服 ID

4.5.5. 响应结果

无

4.5.6. 特殊说明

无

4.6. 设置 SDK 显示的语种[[必需]]

4.6.1. 接口说明

可设置 SDK 显示的语种，目的是 SDK 与游戏统一语种。在游戏启动时、游戏切换语种时调用。默认与系统语言一致。

当游戏是单包单语种时，必须设置为游戏的语种；否则 SDK 将与系统语言一致。

目前支持语种：英语、中文、繁体中文（台湾、香港）、越南语、泰语。

4.6.2. 前置条件

建议：在 application 中进行设置语种

最好在 application 中进行设置语种（则不用重启 activity 就可以加载到对应的语种）

注：需要在//初始化 ContextUtil 配置完后才可以调用本接口

例如：

```
//初始化上下文配置
ContextUtil contextUtil = ContextUtil.getInstance();
//传入 application
contextUtil.setApplication(this);
//设置全局上下文
contextUtil.setApplicationContext(this.getApplicationContext());
//语言初始化
LanguageUtil.getInstance().setLocaleLanguage(Language.zh_rCN);
```

如果在 application 以外类中调用本接口后，需要重启 activity

```
private void restartApplication() {
    //切换语言信息，需要重启 Activity 才能实现
    Intent intent = new Intent(this, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);
}
```

例如：

```
LanguageUtil.getInstance().setLocaleLanguage(language);
restartApplication();
```

4.6.3. 接口名称

```
/**
 * @param language 语种
Language 枚举：
en[英语]、zh_rCN[中文简体]、zh_rHK[中文繁体]、zh_rTW[中文繁体]、vi[越南文]、th[泰文]

 */
LanguageUtil.getInstance().setLocaleLanguage(Language.zh_rCN);
```

4.6.4. 接口参数

参数	类型	是否必填	说明
language	Language	是	语种

4.6.5. 响应结果

无

4.6.6. 特殊说明

游戏启动时，未设置语种，将默认显示系统语言。

4.7. 设置 Facebook 的密钥散列[[必需]]

4.7.1. 接口说明

用于获取 facebook 登录的开发及发布密钥散列

4.7.2. 前置条件

SDK 初始化完成并已设置上下文：
contextUtil.setApplicationContext(this.getApplicationContext());

module(app)build.gradle 中含有签名配置

4.7.3. 接口名称

```
/**
 * 获取 facebook 的密钥散列
 * @return  keyHash {String}
 */
YRGameSDKManager.getInstance().getKeyHash()
```

4.7.4. 接口参数

无

4.7.5. 响应结果

返回一串密钥散列

4.7.6. 特殊说明

获取该密钥散列后，需配合将完整密钥复制给 SDK 运营，或者上报给 SDK 业务后台。

4.8. 设置渠道[[按功能]]

4.8.1. 接口说明

获取应用的渠道 ID，用于分析市场来源

4.8.2. 前置条件

1、在 AndroidManifest.xml 中的 application 块内加入下列语句：

```
<!-- 渠道统计 主要值在 build.gradle 中 -->
                <meta-data
                android:name="UMENG_CHANNEL"
                android:value="${UMENG_CHANNEL_VALUE}    />
                "
```

2、在 app 下的 build.gradle 中打包发布前配置 productFlavors 例如：

```
android{
...
buildTypes {
        release {
            minifyEnabled true
            signingConfig signingConfigs.release
            proguardFiles getDefaultProguardFile('proguard-android.txt'),'proguard-rules.pro'
            applicationVariants.all { variant ->
                variant.outputs.each { output ->
                    def outputFile = output.outputFile
                    if (outputFile != null && outputFile.name.endsWith('.apk')) {
                        def fileName1 =
"                                $                {archivesBaseName}
_v${defaultConfig.versionName}_${variant.productFlavors[0].name}.apk"
                        output.outputFile = new File(outputFile.parent, fileName1)
                    }
                }
            }
        }
    ...
}
productFlavors {
        google {}
        productFlavors.all { flavor ->
            flavor.manifestPlaceholders = [UMENG_CHANNEL_VALUE: name]
        }
```

```
    }
.....}
```

4.8.3. 接口名称

无

4.8.4. 接口参数

无

4.8.5. 响应结果

无

4.8.6. 特殊说明

无

4.9. 退出游戏弹框提示[[按功能]]

4.9.1. 接口说明

点击一下手机的返回键就弹出退出游戏的对话框

4.9.2. 前置条件

游戏本身的 launcher activity 中没有退出游戏提示的逻辑处理，且 onBackPressed
及 onKeyDown 的方法内没有对 if (keyCode == KeyEvent.*KEYCODE_BACK*) 进行多余的退出提示

4.9.3. 接口名称

launcher activity 中继承 QuitBaseActivity

注：如果游戏本身有继承了别的 BaseActivity 无需替换继承 QuitBaseActivity 可直接在 onKeyDown 或者 onKeyUp 中调用 QuitBaseActivity.createQuitDialog(Activity);
方法的用例：

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (KeyEvent.KEYCODE_BACK == keyCode) {
        QuitBaseActivity.createQuitDialog( activity: this);
    }
    return super.onKeyDown(keyCode, event);
}
```

4.9.4. 接口参数

无

4.9.5. 响应结果

无

4.9.6. 特殊说明

无

5. FAQ

5.1. 将 library 的 AndroidManifest 和主程序的配置进行合并

方法：在 project.properties 定义 manifestmerger.enable=true，这样会自动将 library 的 androidManifest 配置和主程序进行自动合并。
注意：library 的 ADT 版本、minSdkVersion 和 targetSdkVersion 必须好主程序的相同，否则会在编译时出错。