링타이거 MiguSDK

개발 문서

（Ver 4.32）

2016년5월6일

**History**

|  |  |  |
| --- | --- | --- |
| 버전 | 날짜 | 내용 |
| 1.8 | 2015-6-16 | 각종 네이티브 SDK관련 버그 수정 |
| 1.81 | 2015-6-18 | ANYSDK 종료 관련 버그 수정  통합 SDK 리소스 재정리 및 버그 수정 |
| 1.82 | 2015-6-19 | SDK 리소스 일부 버그 수정 |
| 1.9 | 2015-6-24 | Miguyouxi 대쉬보드 연동 버그 수정  문서 내용 수정 ( 1.8에서 업그레이드법) |
| 2.01 | 2015-7-17 | ANYSDK 업데이트 및 China Telecom SDK 업데이트 적용 |
| 3.0 | 2015-9-7 | 이통사 SDK 제거 및 YiJie SDK 적용 |
| 3.42 | 2015-11-8 | 네트워크 결제 영수 검증 프로세스 추가 |
| 3.43 | 2015-11-9 | 유니티 빌드 에러 해결 및 문서 일부 수정 |
| 4.01 | 2016-2-9 | GooglePlay 결제 추가 |
| 4.32 | 2016-5-6 | GooglePlay 결제 관련 중대 버그 수정 |

# 1개요

# 1.1 소개

본 MiguSDK 는 [www.miguyouxi.com](http://www.miguyouxi.com) 플랫폼을 통해서 중국내에서 안드로이드 게임을 서비스 하기 위해서 기본적으로 사용해야 하는 결제 SDK들을 통합하여, 한국 개발사에 제공하기 위한 목적으로 만들어졌습니다. 본 SDK에 포함된 내용은 YiJieSDK결제 모듈 ( 통합 3 자 마켓용 결제 모듈 )을 모두 통합하여, 개발자가 한 번의 SDK 적용으로 손쉽게 중국 내 모든 결제 모듈을 대응 가능하도록 만든 것입니다.

# 2 Miguyouxi.com 설명

## 2.1서비스 과정

* **순서도**

1. Miguyouxi.com 개발자 계정 등록

~~6. 이통사 심사~~

~~문제 발생/재심사~~

~~심사 통과~~

2. 게임/앱 정보 등록

3. 전자계약서 다운로드 및 체결 후, 업로드

4. 테스트용 APK 업로드 ( 중문이 아니어도 됨 )

5. 빌링 항목 추가 및 중국어 번역 적용 + SDK 적용

~~7. 최종 이통사 키값 및 최종 런칭 버전 제작~~

8. 링타이거 마켓 런칭 진행

## 2.2 SDK 적용

**1）SDK 폴더 구성**

|  |
| --- |
| /MiguSDKDemoForUnity -> 유니티용 셈플 및 SDK 파일구조가 들어가 있음  /NativeMigu -> 네이티브용 셈플 및 SDK 파일구조가 들어가 있음  /SDKForNative -> 네이티브용 SDK 필요 파일만 빼놓은 내용  /SDKForUnity -> 유니티용 SDK 필요 파일만 빼놓은 내용  MiguSDK\_Verx\_xx\_2016xxxx.doc -> SDK 관련 문서 |

**2）SDK 파일 프로젝트에 적용**

A）네이티브용 SDK

SDKForNative 하위의 모든 폴더 ( assets, libs, res, runtime )내용을 게임 프로젝트의 각각의 폴더에 복사. AndroidManifest.xml 파일은 참고하여 게임 내의 내용 수정

B）유니티용 SDK

SDKForUnity 하위의 Plugins 폴더를, 유니티 프로젝트의 Assets하단의 Plugins 폴더에 복사. SDKForUnity/MiSDK 폴더에는 프로젝트에 실제 적용시 필요한 스크립트가 있으므로, 참조 또는 직접 프로젝트에 포함시켜 사용 가능. AndroidManifest.xml 파일은 참고하여 게임 프로젝트에 활용

**3） SDK세부 설명 ( 네이티브용 SDK )**

i）SDK 포함된 셈플NativeMigu의 temper 액티비티, RTLogoSplash액티비티를 패키지에 추가할 것

ii）SDK포함된 셈플 NativeMigu의 MainActivity에서 아래의 내용을 참고하여, 앱 상에 적용

a) SDK 초기화 관련 부분

\* 3.42 버전 이후에 영수 검증 관련 함수는 CheckBillWithMigu 함수 참조하여 게임에 적용할 것.

|  |
| --- |
| **public** **void** OnInit()  {  System.*out*.println("Migu SDK init :" + AppName + ":" + CompanyName);  MainListener = **new** BillingListener()  {  @Override  **public** **void** onPurchaseSucceed(String item, String code3rd)  {  System.*out*.println("Migu Buy Success :" + item);    // 3.4 버전에서 추가된 내용 - 영수 검증 단계 - 결제 성공이 오면, 반드시 아래의 CheckBillWithMigu 함수를 이용해서 결제 검증 후, 그 결과에 따라 아이템을 지급해야 함  **if** ( code3rd != **null** )  {  **try**  {  CheckBillWithMigu(code3rd);  }  **catch**(Exception c)  {  showDialog(MainActivity.**this**, "showDialog", "error checkbillwithmigu");  }  }  **else**  showDialog(MainActivity.**this**, "Payment", "ITEM:" + item + "\nBuy Success" );    }    @Override  **public** **void** onPurchaseInfo(String item, String msg)  {  System.*out*.println("Migu Buy Info :" + item + ":" + msg);  }    @Override  **public** **void** onPurchaseFailed(String item, String msg)  {  System.*out*.println("Migu Buy Failed :" + item + ":" + msg);  showDialog(MainActivity.**this**, "Payment", "ITEM:" + item + "\nBuy Failed!!!" );  }    @Override  **public** **void** onPurchaseCanceld(String item, String msg)  {  System.*out*.println("Migu Buy Canceled :" + item + ":" + msg);  showDialog(MainActivity.**this**, "Payment", "ITEM:" + item + "\nBuy Canceled!!!" );  }  };  postHandler.post( **new** Runnable()  {  **public** **void** run()  {  MiguSDKFactory.*init*(  MainActivity.**this**, MainListener,  MM\_appID, MM\_appkey,  CompanyName, QAPhone, AppName );  }  });  } |

b) 결제 요청 부분

|  |
| --- |
| **public** **void** OnBuy( **final** String smsPayItem, **final** String vaccode, **final** String props, **final** String money )  {  // 3.42 버전 이후 추가된 내용. 로그인이 되었는지 확인해서 로그인 진행  **if** ( MiguSDKFactory.*getInstance*().isLogined() == **false** )  {  MiguSDKFactory.*getInstance*().login();  **return**;  }    System.*out*.println("try to buy :" + smsPayItem);    postHandler.post( **new** Runnable()  {  **public** **void** run()  {  MiguSDKFactory.*getInstance*().pay(MainActivity.**this**, smsPayItem, vaccode, props, money, MainListener, **true**);  }  });  }  });  } |

c) 각 플랫폼별 기본 요구 사항 처리 부분 – NativeMigu의 MainActivity내용 사용

\* 게임에 반드시 적용해야 함

|  |
| --- |
| // 앱 종료 처리  **public** **void** exitGame()  {  **if** ( MiguSDKFactory.*getInstance*() != **null** )  postHandler.post( **new** Runnable()  {  **public** **void** run()  {  MiguSDKFactory.*getInstance*().exitGame(MainActivity.**this**, **new** ExitGameListener()  {  @Override  **public** **void** onExitGameConfirmExit()  {  System.*exit*(0);  }    @Override  **public** **void** onExitGameCancelExit()  {  }    @Override  **public** **void** onExitGameInGame()  {  }  });  }  });  } |

\* 게임에 반드시 적용해야 함

|  |
| --- |
| // 더 많은 게임 보기 기능 처리  **public** **void** viewMoreGames()  {  **if** ( MiguSDKFactory.*getInstance*() == **null** )  **return**;    MiguSDKFactory.*getInstance*().viewMoreGames(MainActivity.**this**);  } |

\* ~~게임에 반드시 적용해야 함~~

|  |
| --- |
| ~~// SDK에서 사운드 켜고 크기에 따른 지원 처리~~  **~~public~~****~~boolean~~** ~~isMusicEnabled()~~  ~~{~~  **~~if~~** ~~( MiguSDKFactory.~~*~~getInstance~~*~~() ==~~ **~~null~~** ~~)~~  **~~return~~****~~true~~**~~;~~    **~~return~~** ~~MiguSDKFactory.~~*~~getInstance~~*~~().isMusicEnabled();~~  ~~}~~ |

\* 게임에 반드시 적용해야 함

|  |
| --- |
| // SDK에서 앱 스크린샷 공유 지원 처리  **public** **void** screenShotShare(**final** String filepath)  {  **if** ( MiguSDKFactory.*getInstance*() == **null** )  **return**;    System.*out*.println("Brandon : scr path" + filepath);  MiguSDKFactory.*getInstance*().doScreenShotShare(MainActivity.**this**, Uri.*fromFile*(**new** File(filepath)));  } |

\* 게임에 반드시 적용해야 함

|  |
| --- |
| // 앱에 대한 소개창 표기 처리  **public** **void** AboutUs(Context context)  {  MiguSDKFactory.*getInstance*().AboutUs(context);  } |

|  |
| --- |
| // 앱 기본 변환 처리  @Override  **protected** **void** onStop()  {  **super**.onStop();  **if** ( MiguSDKFactory.*getInstance*() != **null**)  MiguSDKFactory.*getInstance*().Stop(**this**);  }    @Override  **protected** **void** onRestart()  {  // **TODO** Auto-generated method stub  **super**.onRestart();  **if** ( MiguSDKFactory.*getInstance*() != **null**)  MiguSDKFactory.*getInstance*().Restart(**this**);  }    @Override  **protected** **void** onResume()  {  **super**.onResume();  **if** ( MiguSDKFactory.*getInstance*() != **null**)  MiguSDKFactory.*getInstance*().Resume(**this**);  }    @Override  **public** **void** onPause()  {  **super**.onPause();    **if** ( MiguSDKFactory.*getInstance*() != **null**)  MiguSDKFactory.*getInstance*().Pause(**this**);  }  @Override  **protected** **void** onDestroy() {  **super**.onDestroy();  **if** ( MiguSDKFactory.*getInstance*() != **null**)  MiguSDKFactory.*getInstance*().Destroy(**this**);  };  // 4.3 이후추가 내용  @Override  **protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  **if** ( MiguSDKFactory.*getInstance*() != **null**)  MiguSDKFactory.*getInstance*().ActivityResult(requestCode, resultCode, data);  **else**  **super**.onActivityResult(requestCode, resultCode, data);  } |

|  |
| --- |
| // 안드로이드 뒤로가기 버튼 처리  @Override  **public** **boolean** onKeyDown(**int** keyCode, KeyEvent event)  {  **if** (keyCode == KeyEvent.*KEYCODE\_BACK* && MiguSDKFactory.*getInstance*() != **null**)  {  PromptExitInGame = **false**;  MiguSDKFactory.*getInstance*().exitGame(MainActivity.**this**, **new** ExitGameListener()  {  @Override  **public** **void** onExitGameConfirmExit()  {  System.*exit*(0);  }    @Override  **public** **void** onExitGameCancelExit()  {  }    @Override  **public** **void** onExitGameInGame()  {  PromptExitInGame = **true**;  }  });    **if** ( PromptExitInGame == **true** )  **return** **super**.onKeyDown(keyCode, event);    **return** **true**;  }    **return** **super**.onKeyDown(keyCode, event);  } |

iii）AndroidManifest.xml 에서 반드시 포함되어야 할 내용

1. Application 선언에 *com.snowfish.cn.ganga.helper.SFOnlineApplication*추가

|  |
| --- |
| <application  android:allowBackup=*"true"*  android:icon=*"@drawable/ic\_launcher"*  android:label=*"@string/app\_name"*  android:theme=*"@style/AppTheme"*  android:name=*"com.snowfish.cn.ganga.helper.SFOnlineApplication"* > |

b) temper 액티비티 추가에 따른, 메인 액티비티로 지정 및 시작 액티비티로 지정

|  |
| --- |
| <activity android:name=*"게임패키지명.temper"*  android:theme=*"@android:style/Theme.NoTitleBar.Fullscreen"*  android:screenOrientation=*"portrait"*>  <intent-filter>  <action android:name=*"android.intent.action.MAIN"* />  <category android:name=*"android.intent.category.LAUNCHER"* />  </intent-filter>  </activity> |

c) 원래 메인 게임 액티비티에서 LAUNCHER 카테고리 제외

|  |
| --- |
| <activity android:name=*"게임패키지명.메인액티비티"*>  <intent-filter>  <action android:name=*"android.intent.action.MAIN"* />  </intent-filter>  </activity> |

d) RTLogoSplash 액티비티 추가

|  |
| --- |
| <activity  android:name=*"게임패키지명.RTLogoSplash"*  android:theme=*"@android:style/Theme.NoTitleBar.Fullscreen"*  android:screenOrientation=*"portrait"* /> |

e) 통합 결제 SDK용 액티비티들 추가

|  |
| --- |
| <service  android:name=*"com.snowfish.a.a.s.ABGSvc"*  android:enabled=*"true"*  android:process=*"com.snowfish.a.a.bg"*>  <intent-filter>  <action android:name=*"com.snowfish.a.a.s.ABGSvc"* />  <category android:name=*"android.intent.category.DEFAULT"* />  </intent-filter>  </service>    <service android:name=*"com.qy.pay.a.e"* />  <activity android:name=*"com.qy.pay.a.b"* android:configChanges=*"keyboard|keyboardHidden|navigation|orientation"* /> |

f) metadata 추가

|  |
| --- |
| <!--SP의 경우, 아래에 "001"~ 시작하는 아이템 항목에 SP를 붙인 이름 및 관련 결제키값 등록 -->  <meta-data android:value=*"\05144796"* android:name=*"SP001"*/>    <meta-data android:value=*"\064"* android:name=*"BTypeValue"*/>  <meta-data  android:name=*"com.snowfish.appid"*  android:value=*"{20C36611-FDCA7470}"* >  </meta-data>  <meta-data  android:name=*"com.snowfish.channelid"*  android:value=*"{4ff036a1-3254eafe}"*>  </meta-data>  <meta-data  android:name=*"com.snowfish.customer"*  android:value=*"snowfish"* >  </meta-data>  <meta-data  android:name=*"com.snowfish.channel"*  android:value=*"snowfish"* >  </meta-data>  <meta-data  android:name=*"com.snowfish.sdk.version"*  android:value=*"2"* >  </meta-data> |

g) permission 추가

|  |
| --- |
| <uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"*/>  <uses-permission android:name=*"android.permission.ACCESS\_WIFI\_STATE"*/>  <uses-permission android:name=*"android.permission.SEND\_SMS"*/>  <uses-permission android:name=*"android.permission.INTERNET"*/>  <uses-permission android:name=*"android.permission.READ\_PHONE\_STATE"*/>  <uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"*/>  <uses-permission android:name=*"android.permission.READ\_EXTERNAL\_STORAGE"*/>  <uses-permission android:name=*"android.permission.SYSTEM\_ALERT\_WINDOW"* />  <uses-permission android:name=*"android.permission.GET\_TASKS"* />  <uses-permission android:name=*"android.permission.WAKE\_LOCK"* />  <uses-permission android:name=*"android.permission.CHANGE\_NETWORK\_STATE"* />  <uses-permission android:name=*"android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"*/> |

**h) 화면 방향에 따른 각각의 액티비티에 portrait/landscape 통일**

|  |
| --- |
| <activity android:name=*" "*  android:theme=*"@android:style/Theme.NoTitleBar.Fullscreen"*  android:screenOrientation=*"portrait"*> |

iv）기타 추가 작업

**a) res/values 폴더의 strings.xml 파일 내의 temper 액티비티를 실제 게임에 포함시킨 temper액티비티로 변경**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <!-- Class name of game entry -->  <string name=*"sf\_class\_name"*> *게임패키지명.temper* </string></resources> |

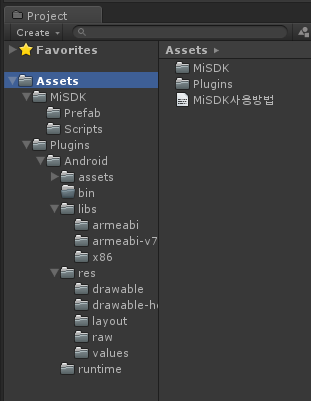
**b) res/values 폴더의 strings.xml 파일 내의 app\_name에 정확히 등록한 게임명으로 변경 ( 2.0 이후 버전 추가 내용 )**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <!-- Class name of game entry -->  <string name=*"app\_name"*>게임명을 정확히 입력</string>  </resources> |

**4） SDK세부 설명 ( Unity용 SDK )**

i）SDK 포함된 셈플 MiguSDKDemoForUnity를 참고하여, 프로젝트에 적용할 것 – 폴더 구조

ii）Plugins 하단의 내용을 게임 프로젝트에 복사



iii) MiSDKProcessor.cs 코드의 내용을 참고하여 게임상에 적용

a) SDK 에 사용될 초기값 셋팅

|  |
| --- |
| public String MM\_appID; // MiSDK\_dev\_appID 값과 동일한 값, 대쉬보드에서 확인한 값으로 인스펙터 창에서 정상값 입력할것  public String MM\_appKey; // MiSDK\_dev\_appkey 값과 동일한 값, 대쉬보드에서 확인한 값으로 인스펙터 창에서 정상값 입력할것  public String Company\_name = "杭州指老虎网络科技有限公司"; // 기본 데이터 사용  public String QAPhone\_number = "57182877709"; // 기본 데이터 사용  public String APP\_name; // 게임명 지정 |

b) SDK 초기화

|  |
| --- |
| MiguSDKBilling.instance.InitializeApp(  MM\_appID, MM\_appKey,  Company\_name, QAPhone\_number, APP\_name ); |

b) 결제 요청

|  |
| --- |
| String encrypt = MD5.Md5Sum(MiSDK\_dev\_appKey + "com.miguyouxi");  String extkey = MiSDK\_dev\_appID + "|" + encrypt;  // DoBilling 함수 - 인자 순서별 ( 아이템 ID - 001 부터 시작, MiSDK 키, 아이템 이름, 가격( 1 은 0.01위안 ), 결과 받을 게임 오브젝트명, 결과 받을 함수명 )  MiguSDKBilling.instance.DoBilling(itemidx.ToString("D03"), extkey, "아이템 명”, "금액(위안)", “결제 결과 전달받을 GameObject 고유이름”, "결제 결과 처리할 함수명 ");  } |

c) 결제 결과 처리

|  |
| --- |
| void 결제 결과 처리 함수명 (String result)  {  String[] results = result.Split('|');  switch( int.Parse(results[0]) ) // 첫번째는 성공 여부  {  case 1: // 성공  // 3.42 버전부터 추가된 내용. 영수 검증 단계를 위해, 성공 후 얻어온 각각의 값을 전달하여 실제 영수 검증 진행  MiSDKCheck(results[1], results[3], results[4], results[5], results[6], results[7] );  Debug.Log("Brandon Unity : Confirm real pay =" + results[1] + "- item, " + results[3] + " ordered, " + results[4] + " app, " + results[5] + " key, " + results[6] + " userid, " + results[7] + " packageName " );  break;  case 2: // 취소에 대한 처리  break;  default: // 결제 실패에 대한 처리  break;  } |

iv）영수 검증 함수 \*\*\* ( 3.42 버전 이후 추가됨. 반드시 이 과정을 완료한 후에 최종 성공시 아이템 등을 지급해야 함 )

|  |
| --- |
| /// <summary>  /// MiSDK 서버와 실제 결제가 이루어졌는지 체크 시도  /// </summary>  /// <param name="orderID"></param>  void MiSDKCheck(String itemidx, String orderID, String appid, String appkey, String userid, String packageName)  {  WebSlaveParam param = new WebSlaveParam();  // 접속할 URL  param.url = "http://open.miguyouxi.com/index.php?m=open&c=yisdk&a=queryPay&";  // 리턴 받을 객체 - 현재 객체  param.return\_object = gameObject;  // 리턴 받을 함수명 지정  param.return\_function = "ResultConfirmFromMiSDK";  // POST 형식인지 GET 형식인지에 따라 처리 - 이건 POST  param.form = new WWWForm();  param.form.AddField("packageName", packageName );  param.form.AddField("mi\_appid", appid );  param.form.AddField("mi\_appkey", appkey );  param.form.AddField("orderID", orderID);  param.form.AddField("userID", userid);  String randstr = MD5.Md5Sum(System.DateTime.Now.Ticks.ToString());  randstr = randstr.Substring(0, 10);  param.form.AddField("Rand", randstr );  String original = "packageName=" + packageName + "&mi\_appid=" + appid + "&mi\_appkey=" + appkey + "&orderID=" + orderID + "&userID=" + userid + "&rand=" + randstr;  String key = MD5.Md5Sum(original + "D8936149A201D1B0");  param.form.AddField("sign", key);  // CallWeb 함수 호출  WebSlave.instance.SendMessage("CallWeb", param);  Debug.Log("Brandon Unity : Start checking : " + param.url);  }  /// <summary>  /// 결제 MiSDK 서버로 체크 요청 후 결과 받은 콜백 함수  /// </summary>  public void ResultConfirmFromMiSDK(WebSlaveParam result)  {  Debug.Log("Brandon Unity : MiSDK Check result : " + result.return\_value);  if (result.return\_error != null) // 여기서 실패는 웹 통신 자체의 문제로 실패이므로, 3회 바로 재시도  {  Check\_success\_counter++;  if (Check\_success\_counter < 3)  {  WebSlave.instance.SendMessage("CallWeb", result);  return;  }  else  {  // 결제 실패 관련 함수 처리  Debug.Log("Brandon Unity : MiSDK Check Perfect failed" );  return;  }  }  JSONObject datas = new JSONObject(result.return\_value);  String status = datas.GetString(datas["status"]);  String message = datas.GetString(datas["message"]);  if ( status.CompareTo("success") == 0 ) // 성공  {  //  // 여기에 성공 관련된 처리 및 관련 함수 호출  //  Debug.Log("Brandon Unity : MiSDK Confirm Success !!");  }  else  {  // 결제 실패 관련 함수 처리  Debug.Log("Brandon Unity : MiSDK Confirm Perfect failed");  return;  }  } |

iv）AndroidManifest.xml 에서 반드시 포함되어야 할 내용

1. MiguSDKDemoForUnity 에 첨부된 AndroidManifest.xml 을 전혀 수정할 필요 없이 그대로 사용. 즉, 기존에 사용하는 기타 플러그인 등이 있다면, 이 xml 파일에 추가하여 사용할 것
2. **아래의 메타 데이터만 대쉬보드에서 확인하여 교체**

|  |
| --- |
| 1. <!--com.snowfish.appid 를 대쉬보드에서 확인한 값으로 사용 --> 2. <meta-data 3. android:name="com.snowfish.appid" 4. android:value="{EDC39B9A-CF2E836E}" > 5. </meta-data> |

1. **화면 방향에 따른 각각의 액티비티에 portrait/landscape 통일**

|  |
| --- |
| <activity android:name=*" "*  android:theme=*"@android:style/Theme.NoTitleBar.Fullscreen"*  android:screenOrientation=*"portrait"*> |

**f) res/values 폴더의 strings.xml 파일 내의 app\_name에 정확히 등록한 게임명으로 변경 ( 2.0 이후 버전 추가 내용 )**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <!-- Class name of game entry -->  <string name=*"app\_name"*>게임명을 정확히 입력</string>  </resources> |

## 2.3 각각의 버전 빌드 방법

**1）최초 테스트용 제출 버전**

A）한국어 또는 중국어 버전 APK

기본적인 중국내 서비스 가능한 게임인지에 대한 여부를 판단 할 수 있는 내용이면 됨

**2） 통합용 최종 빌드**

A） AndroidManifest.xml 에서 아래의 내용 수정

|  |
| --- |
| <meta-data android:value="\01" android:name="BTypeValue"/>   * <meta-data android:value="\064" android:name="BTypeValue"/> |

B）빌드 후, APK 업로드

# 3 반드시 게임/앱에 포함되어야 할 기능

* About Us 관련 버튼 – SDK의 관련 함수 실행해야 함
* ~~isMusicEnabled 관련 처리 – SDK의 관련 함수를 게임/앱 시작전에 체크하여, False 인 경우, 게임/앱의 사운드와 BGM모두 끈 상태로 시작하게끔 처리되어야 함~~
* moreGames 관련 버튼 – SDK의 관련 함수 실행해야 함
* screenShotShare 관련 버튼 – SDK의 관련 함수 실행해야 함
* exit 관련 버튼 – SDK의 관련 함수 실행해야 함
* 안드로이드 뒤로가기 버튼 처리 – 게임/앱 내에서 기본적으로 뒤로가기 버튼에 대한 처리가 모든 UI에서 완벽해야 하며, 특히 제일 상단(최초 화면)에서 누를 경우, exit 관련 버튼처럼 종료 관련 함수가 실행되게 처리되어야 함
* 게임 명칭의 아이콘 표기 및 패키지 설치시 표기, 게임 타이틀에서 표기되는 내용이 모두 명확히 일치해야 함
* 위의 내용들 중 하나라도 빠져있는 경우, 일부 플랫폼에서 거부가 될 수 있음

# 4 FAQ

* 현재 정리중

# 5 자체 테스트 방법

1. 최종 빌드 실행 후, 로그인 진행 – 자동 진행 가능
2. 게임상에서 결제 진행시, 검은색 메시지 창으로 결제 진행 내용이 잠시 표기되면 문제 없음. 실제 결제는 이루어지지 않으며, 아이템 구입 과정도 완료되지 않음. 즉, 혹시나 중간 빌드 과정에서 APK가 유출되는 사고를 방지하고 최소한의 테스트만을 진행하기 위한 조치임