

# LSViewer-Android 开发指南

## V2.0

撰写单位	苏州中科图新网络科技有限公司
最后修订日期	2016 年 12 月

### **法律说明**

版权所有苏州中科图新网络科技有限公司。

本文档包含的所有内容除特别声明之外，版权均属于苏州中科图新网络科技有限公司所有，图新可在不作任何声明的情况下对本文档内容进行修改。

本文档中所使用的商标所有权属于该商标的所有者。

# 目录

目录.....	3
概述： .....	5
LSViewer-Android SDK2.0.....	5
功能介绍.....	5
一、简介.....	8
1.1 LSViewer Android SDK（JAVA）功能介绍.....	8
1.2 面向的读者.....	8
1.3 问题解答.....	8
1.4 文档结构介绍.....	8
二、Hello World.....	9
2.1 Eclipse 创建过程.....	9
2.2 Android Studio 创建过程.....	22
2.3 开发进阶-添加图层和地形： .....	34
三、基本操作控制.....	36
3.1 缩放.....	37
3.2 旋转.....	39
3.3 翻转.....	40
3.4 飞行.....	42
3.5 跳转.....	45
3.6 设置视图状态.....	47
3.7 设置指南针样式.....	47
四、图层.....	48
4.1 几个概念： .....	48
4.2 添加图层： .....	48
4.3 删除图层.....	48
4.4 图层显示.....	49
4.5 图层隐藏.....	49
4.6 图层可见距离.....	49
4.7 图层不透明度.....	49
4.8 图层渲染顺序.....	50
4.9 图层范围.....	50
4.10 图层类型.....	50
4.11 图层遍历.....	51
五、点、线、面标注： .....	52
5.1 添加点.....	52
5.2 点样式设置.....	53
5.3 添加线.....	54
5.4 线样式设置.....	55
5.5 添加面.....	55
5.6 面样式设置.....	56
5.7 Feature 的删除.....	56

5.8 Feature 的显隐性控制.....	57
六、模型数据.....	57
6.1 添加模型数据.....	58
6.2 调整模型位置.....	59
6.3 删除模型数据.....	59
6.4 属性设置.....	60
七、倾斜摄影.....	60
7.1 倾斜摄影简介：.....	60
7.2 根据倾斜摄影数据生成图层文件.....	64
7.3 倾斜摄影图层文件参数说明.....	64
八、Server 连接.....	64
8.1 连接指定地址的服务器：.....	65
8.2 获取连接服务器成功后的图层信息.....	65
九、量测.....	66
9.1 平面距离量测：.....	67
9.2 三角测量.....	68
9.3 关闭量测.....	68
十、事件.....	69
10.1 要素点击事件.....	69
10.2 回调事件.....	69

## 概述：

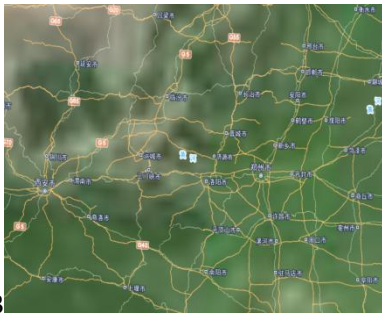
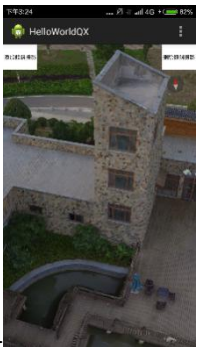
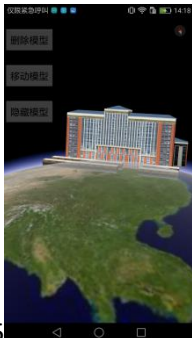

### LSViewer-Android SDK2.0

LSViewer AndroidSDK 是一套基于 Android 2.1 及以上版本设备的应用程序接口。您可以使用该套 SDK 开发适用于 Android 系统移动设备的三维地图相应应用，通过调用 SDK 接口，您可以轻松访问各种在线地图数据，加载本地图层数据，连接 server 服务，构建功能丰富、交互性强的三维地图类应用程序。

LSViewer AndroidSDK 需要申请许可后方可正常使用，如需申请 SDK，请在手机上运行示例工程下面的 apk 文件，获取机器码，提交给中科图新技术支持人员获取许可吗。也可自行编译示例工程通过代码的方式获取机器码。

## 功能介绍

<p><b>1.三维地球</b></p> <p>提供三维地球渲染、触控操作功能</p> <ul style="list-style-type: none"> <li>● 地图加载包括：在线地图、离线地图、栅格数据、矢量数据、地形数据、模型数据</li> <li>● 地图操作包括：点击、双击、缩放、旋转、改变视角等。</li> </ul>	 <p>图 1</p>
 <p>图 2</p>	<p><b>2.在线地图</b></p> <p>提供各种在线地图的无缝接入</p> <ul style="list-style-type: none"> <li>● 在线图源包括：谷歌影像、谷歌地形、天地图地图、OSM 地图、微软地图、1:5 万地质图、气象地图等等</li> <li>● 自定义图源包括：自定义瓦片规则的地图服务</li> </ul>

<p><b>3.离线地图</b></p> <p>支持各种格式的栅格、矢量、地形等数据。</p> <p>栅格数据包括：Irc, Irp, tif, img 等</p> <p>示例数据包括：lgd, kml, kmz, vec, shp 等</p> <p>地形数据包括：Irp, tif, grd, img 等</p>	 <p>图 3</p>
 <p>图 4</p>	<p><b>4.倾斜摄影数据</b></p> <p>支持多种厂家生产的倾斜摄影数据的直接加载显示。</p> <ul style="list-style-type: none"> <li>● 倾斜摄影场景支持包括：PhotoScan, Smart3d, Pixel4D</li> </ul>
<p><b>5.模型数据</b></p> <p>支持各种格式的传统三维模型数据</p> <ul style="list-style-type: none"> <li>● 模型数据格式包括：obj, fbx, osgb, stl 等等</li> </ul>	 <p>图 5</p>
 <p>图 6</p>	<p><b>6.点、线、面标注</b></p> <p>LSViewer SDK 支持多种地图标注类型</p> <p>标注类型支持包括：marker、点、线、面。</p> <p>Marker 标注支持：图像、文字</p>

### 7.Server 端连接

支持直接加载 Server 端发布的任意格式的数据图层。

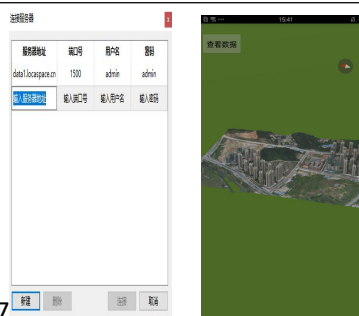


图 7

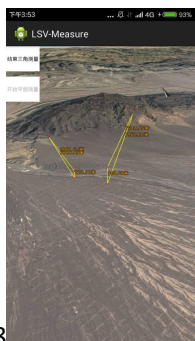


图 8

### 8.三角测量

LSViewer 独有的三角测量，不仅可以直接量测地表面积，长度，也可以量测模型的高度做到基于模型的测量。

## 一、简介

### 1.1 LSViewer Android SDK (JAVA) 功能介绍

LSViewer Android SDK 由纯 C++编写，针对 Android 平台进行 JAVA 接口封装，构成 jar 包。能够帮助您在 Android 平台上快速创建三维交互地图应用。包含了构建三维地图基本功能的各种借口，提供了诸如在线地图加载、本地数据加载、点线面标注、模型数据加载、服务器连接等基本功能和服务。

功能介绍：

- 基本地图功能：
- 点线面标注功能：
- 图层功能：
- 工具类功能：
- 手势交互功能：
- 要素选中功能：
- 倾斜摄影数据加载功能：
- Server 端连接功能：
- 量测功能：

### 1.2 面向的读者

SDK 是提供给有一定 Android 编程经验和了解面向对象概念的读者使用。此外读者还应该对三维地图产品有一定的了解。

在使用中遇到任何问题，都可以通过论坛或者 QQ 交流群反馈给我们。

### 1.3 问题解答

如果您在使用 LSViewer Android SDK 时遇到问题，请尝试通过以下途径解决：

- 确认使用了最新版本的 SDK
- 访问论坛查找问题，或者发布问题到论坛 <http://bbs.3dyuanjing.com/portal.php>
- 查看常见问题指南
- 加入用户交流 QQ 群：181261077
- 客服电话：400-867-5155

### 1.4 文档结构介绍

二次开发包里包含的内容如下图所示：





其中，

【FAQ】是常见问题集锦，里面记录了常见的问题以及解决办法。

【LSViewer-Android 开发包 20161114】是依赖库，里面包含.o 格式的库文件和资源文件

【示例工程】下面，包含了文档里面所列出的所有示例工程的源码以及编译好的 apk 文件

【在线图层】，提供了几种 Irc 格式的在线图源的示例。

【LSViewer(Android)API.pdf】是引擎库的详细 api 接口文档

【LSViewer-Android 开发指南 20161202.pdf】为本文档

## 二、Hello World

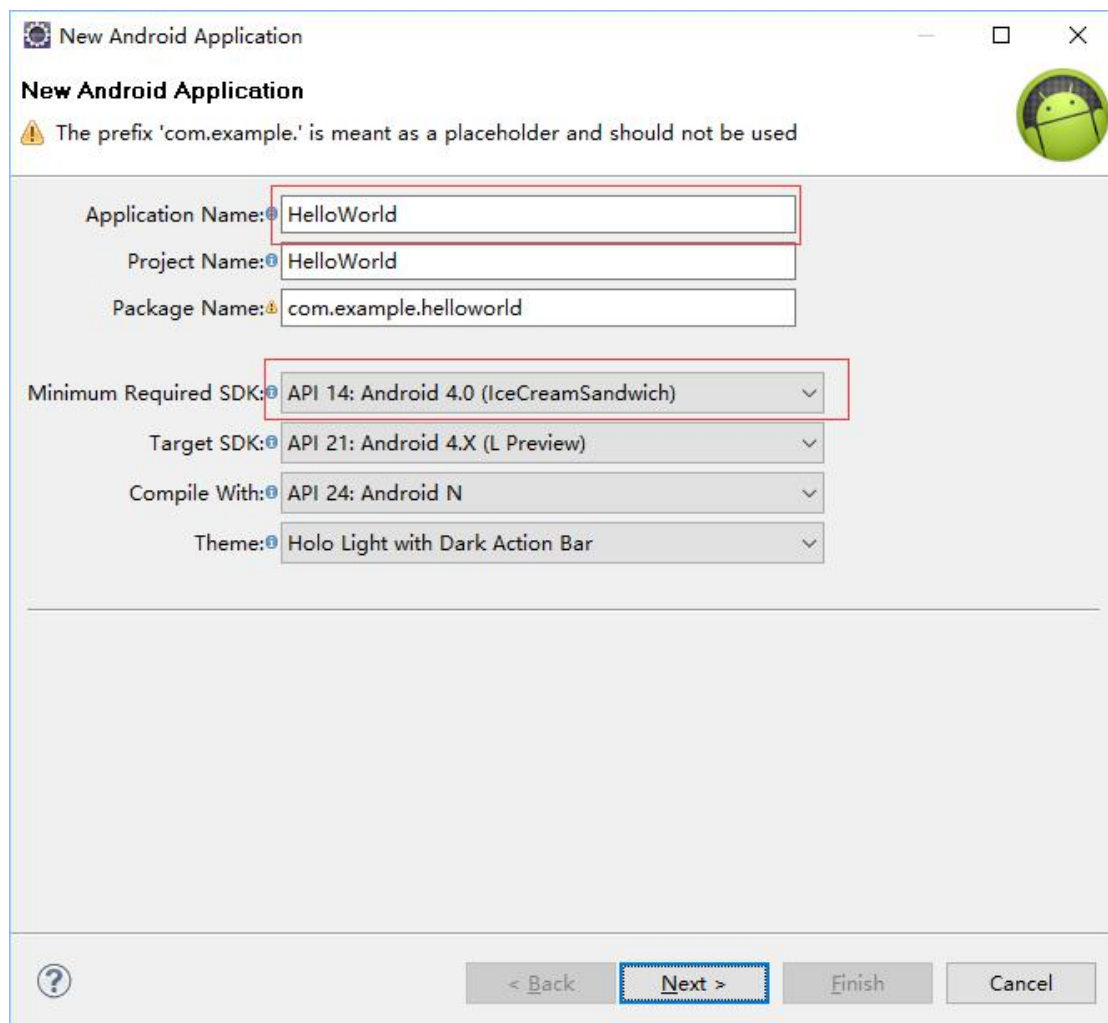
LSViewer 三维地球的“Hello, World”，向地球 say hello

### 2.1 Eclipse 创建过程

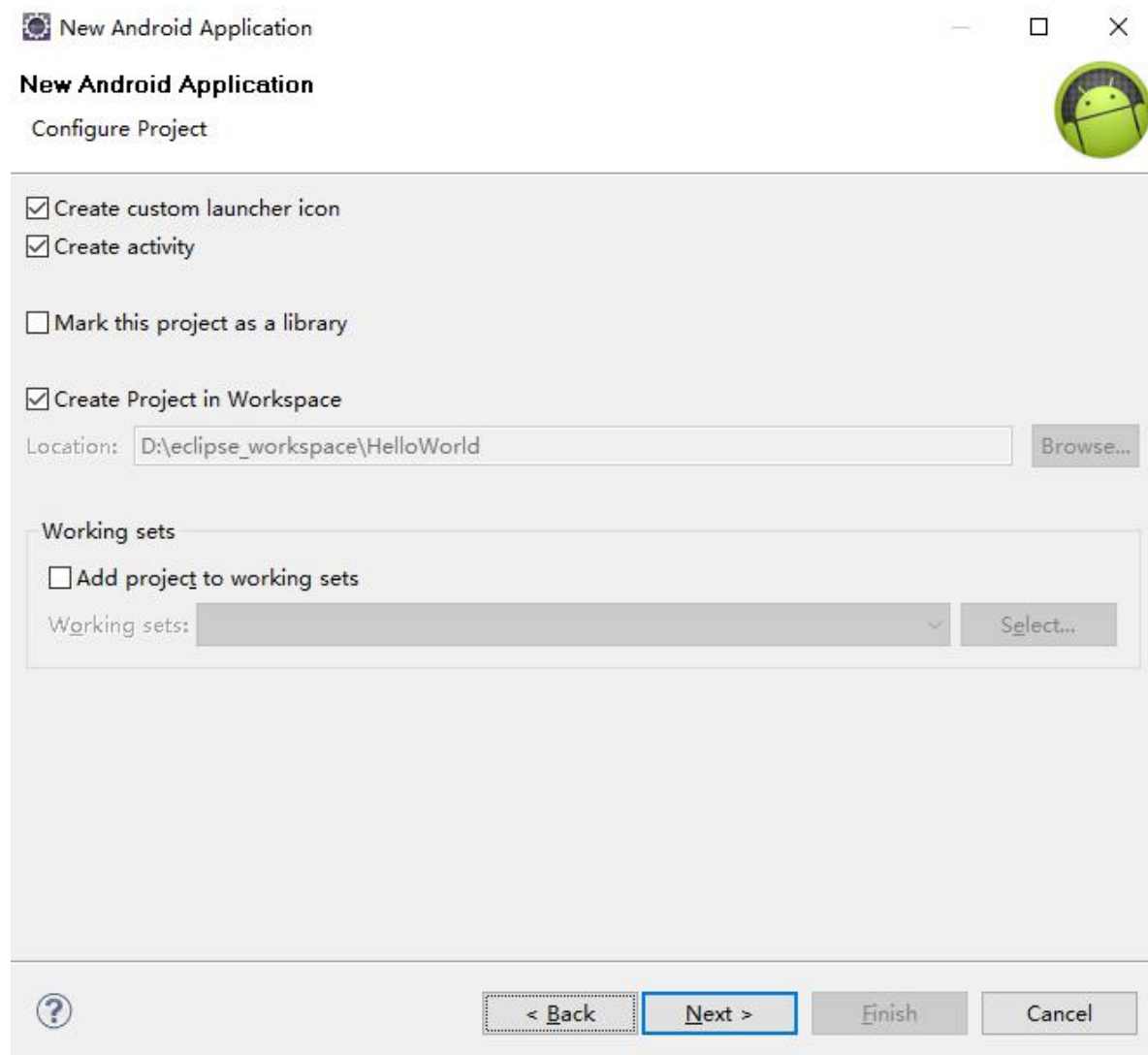
#### 2.1.1 新建工程

注：（确保所有安卓开发环境已经安装完成）

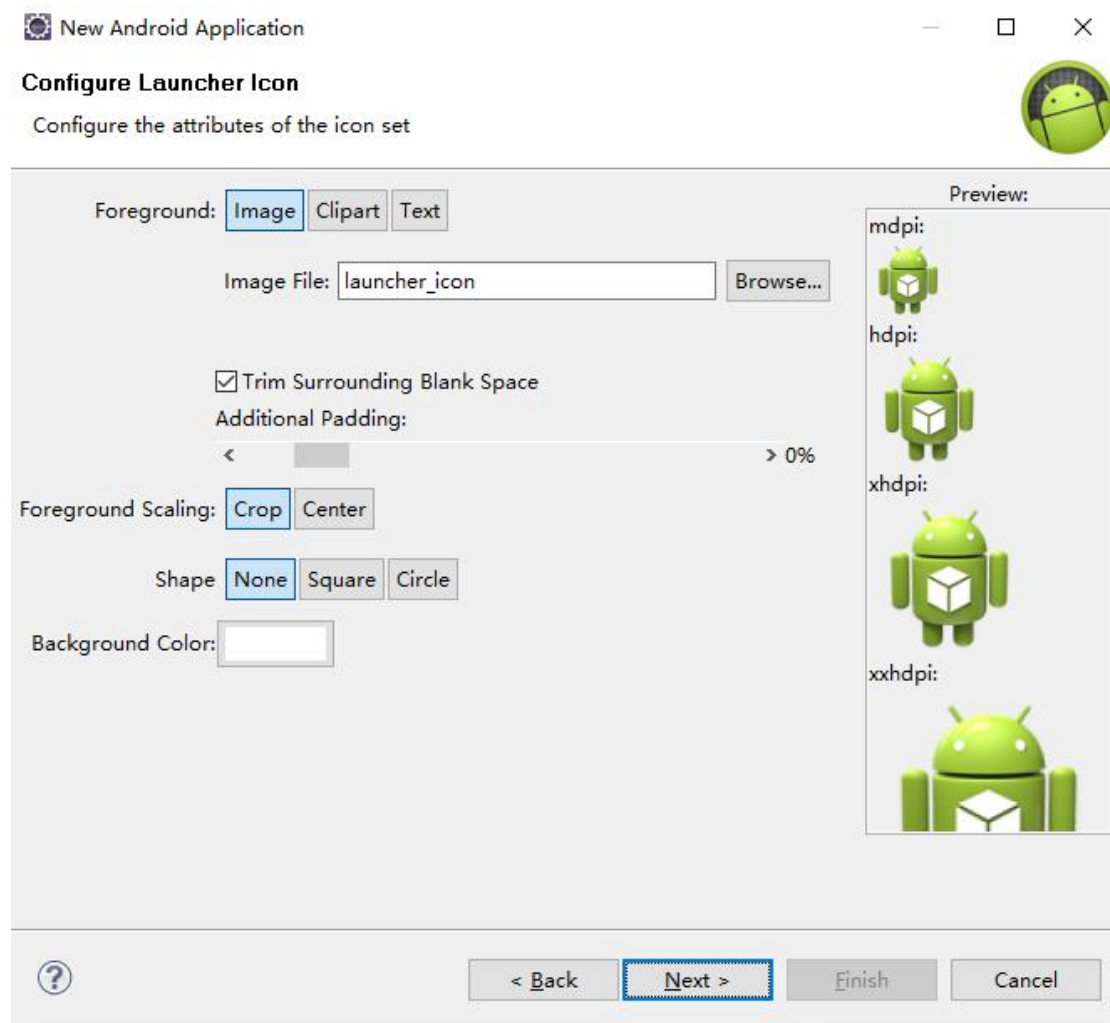
设置工程名称，sdk 版本，完成后点击 Next:



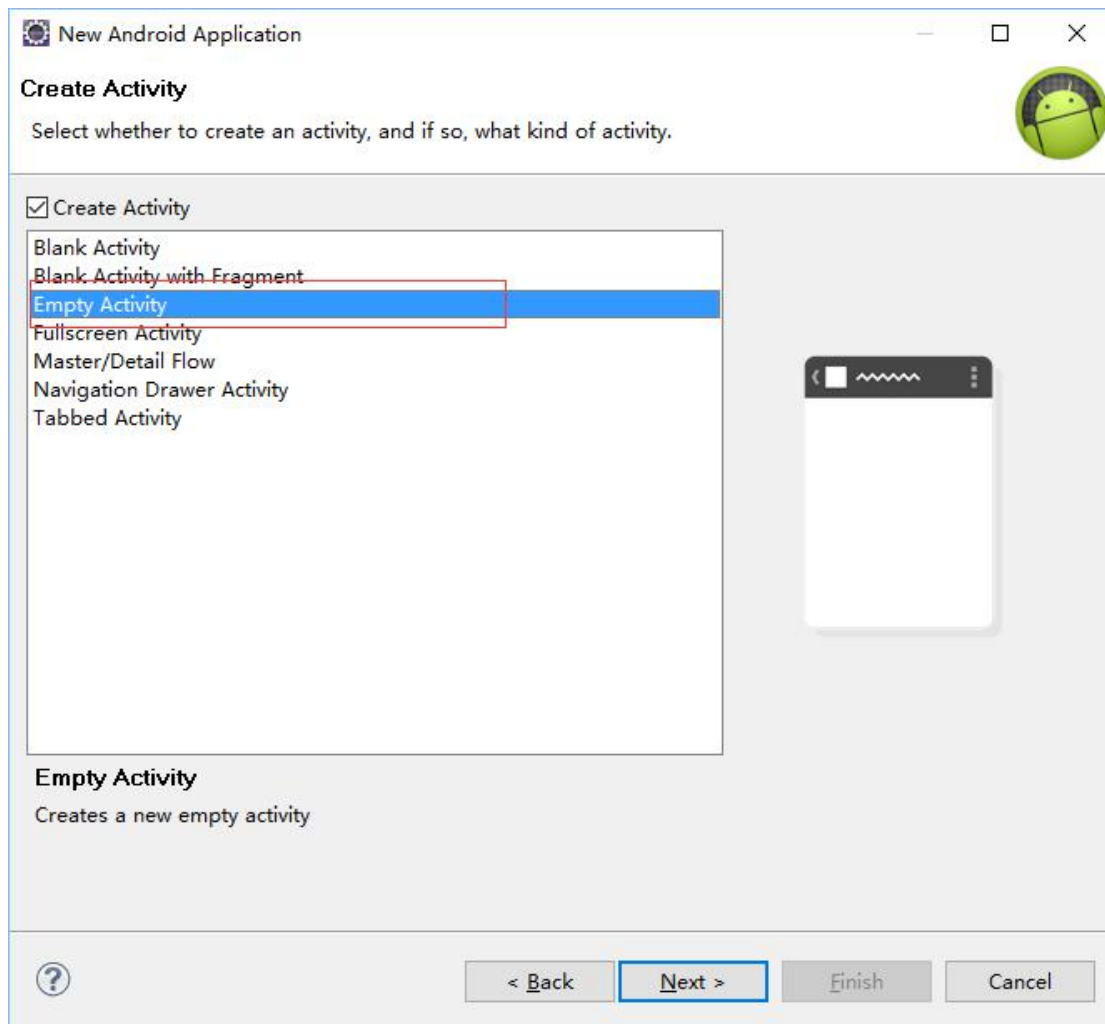
无特殊需求点击 Next 即可



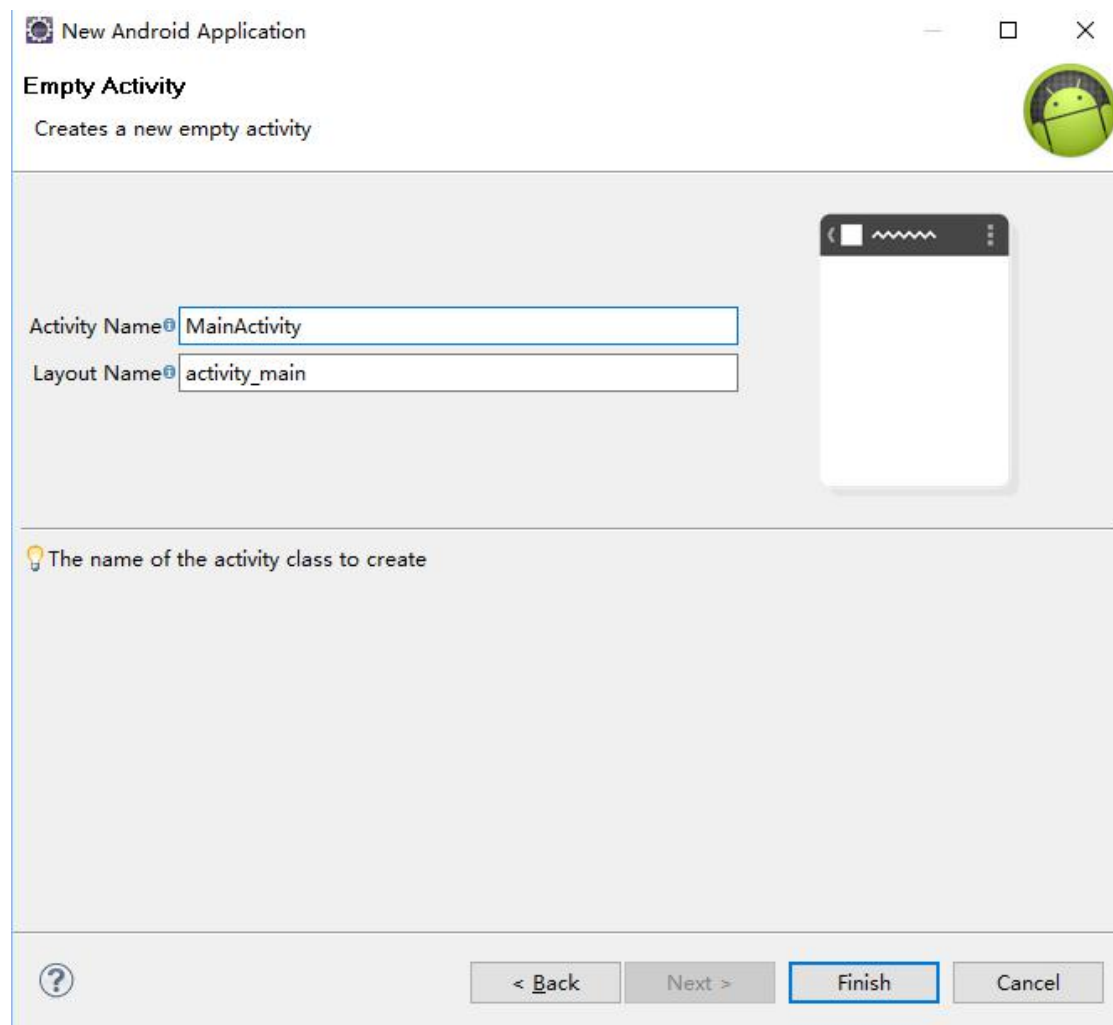
设置 apk 图标样式，无特殊需求点击 Next;



设置工程初始样式，无特殊需求，选择 Empty Activity，点击 Next；

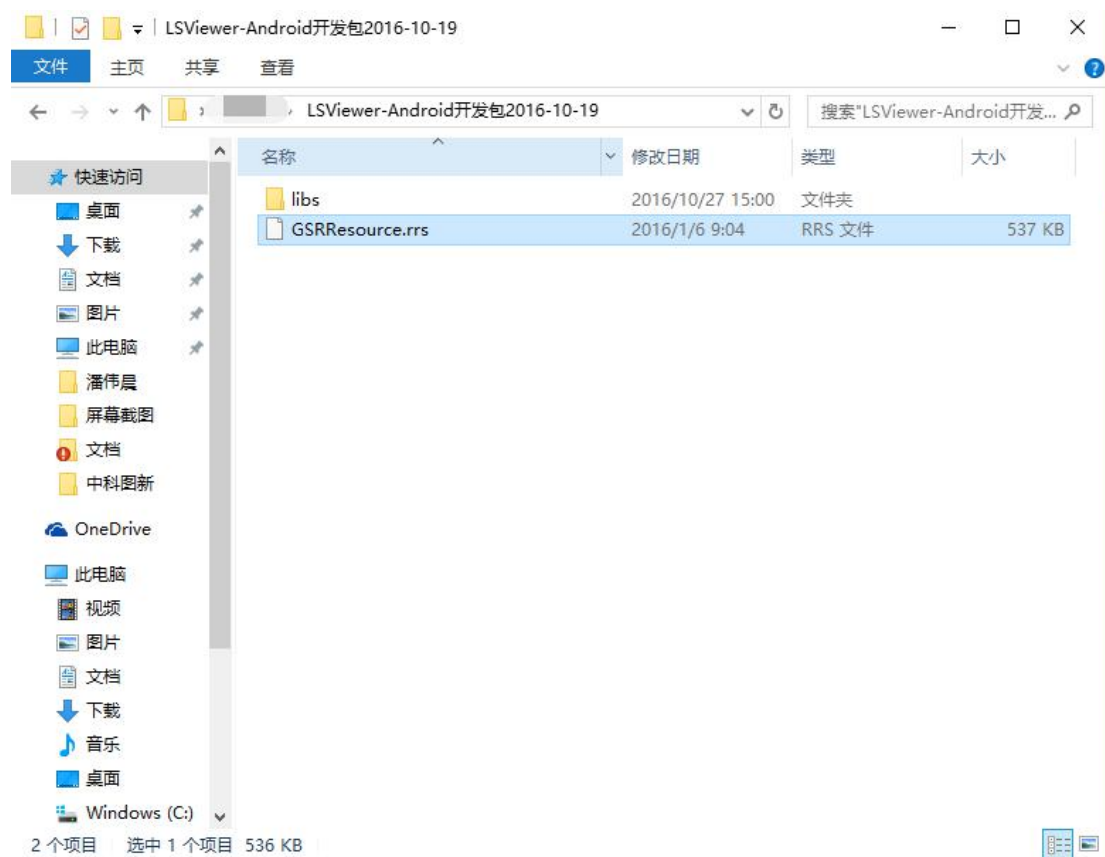


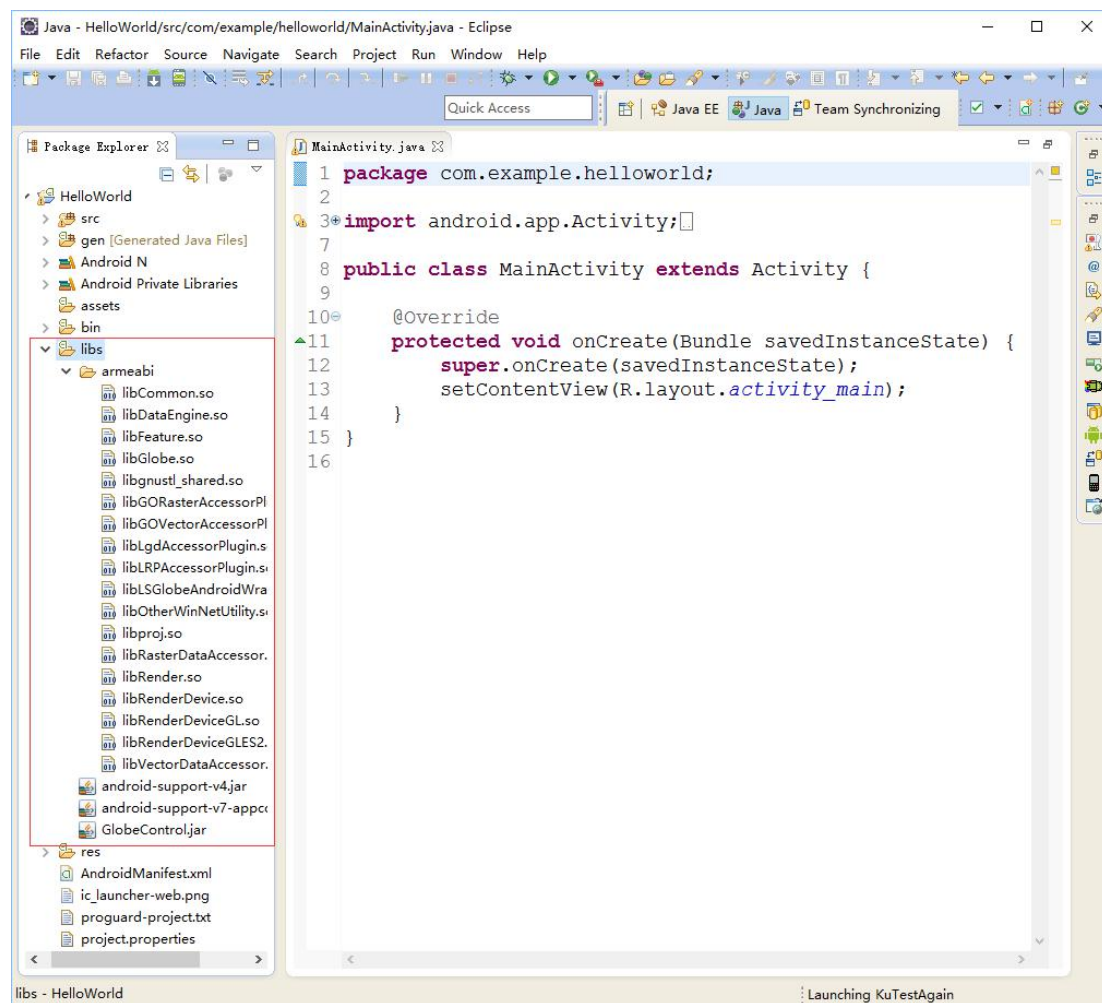
工程创建完成



### 2.1.2 添加库文件引用

打开开发包，复制 libs 文件夹到工程根目录，复制后的效果如下图：  
包含 armeabi 文件夹和 GlobeControl, v4, v7 三个 jar 包；

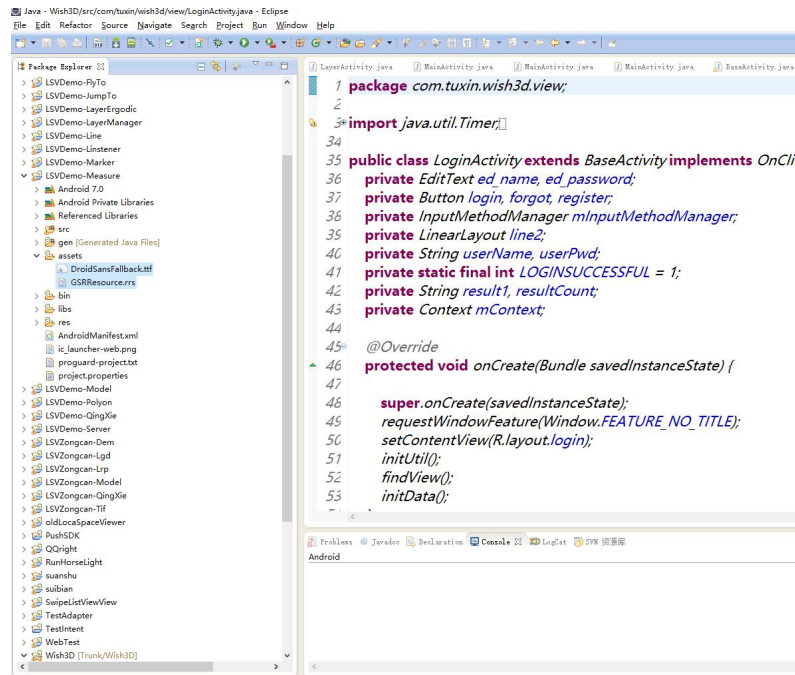




### 2.1.3 添加资源库文件 GSRResource.rrs

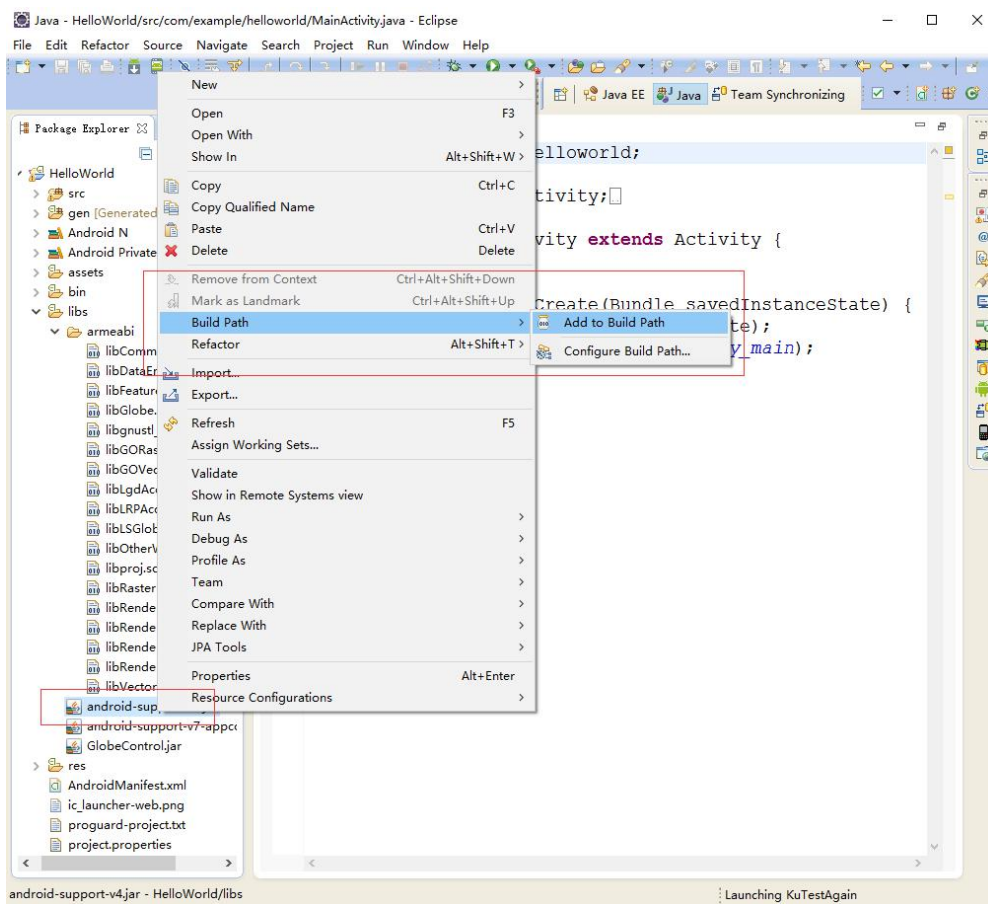
说明：该 rrs 资源文件是引擎正常运行所必需的资源文件，从开发包中复制 GSRResource.rrs 和 DroidSansFallback.ttf(字库文件)到工程根目录下的 assets 文件夹下



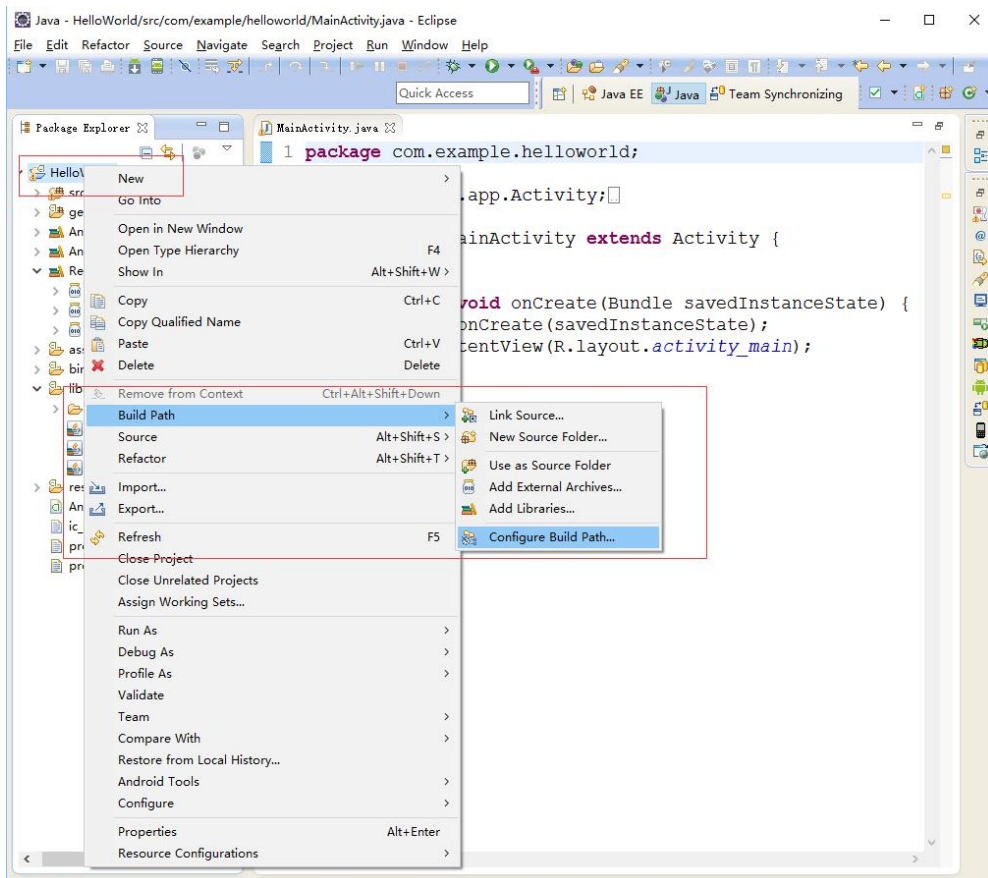


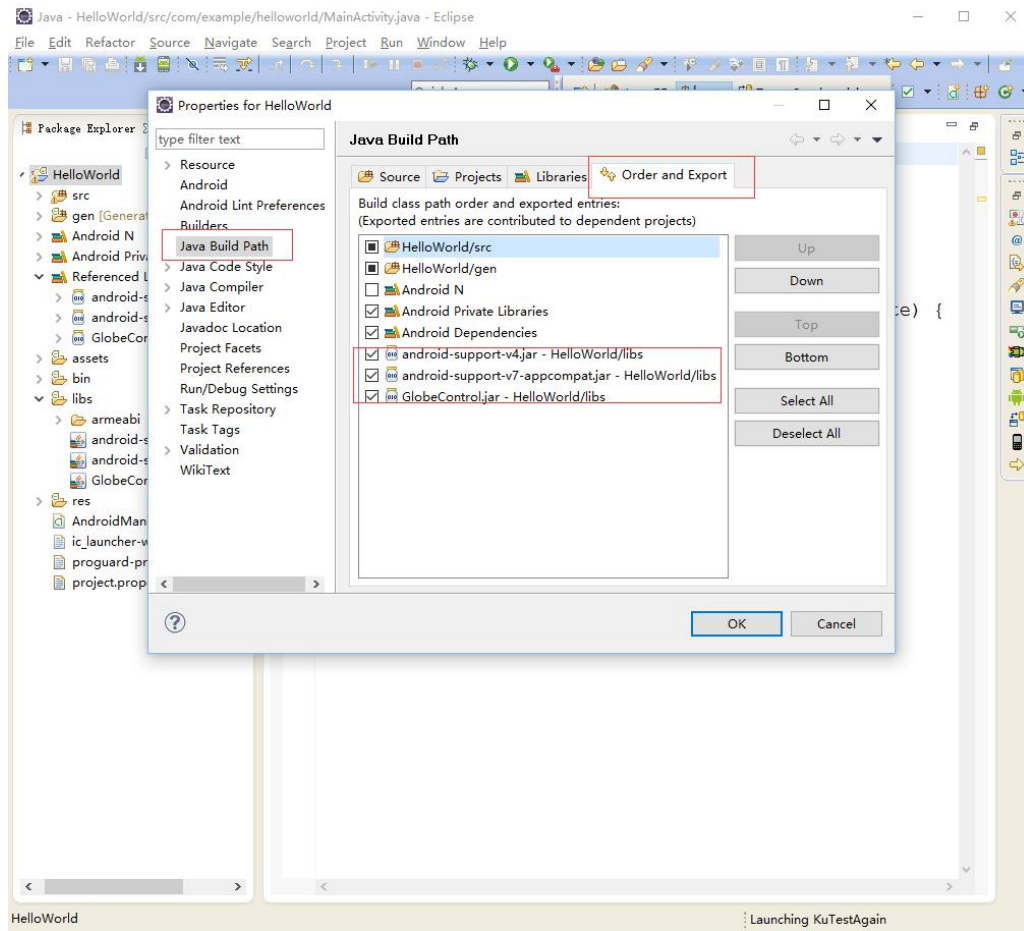
### 2.1.4 配置 jar 包到 build path

在 jar 包上右击选择如下图所示选项，所有 jar 包都需要进行如下操作包括  
 android-support-v4.jar  
 android-support-v7-appcompat.jar  
 GlobeControl.jar



在工程名上右击，执行如下图所示操作





## 2.1.5 配置权限（其中 SD 卡权限必须开启）

为工程添加权限和设置应用属性，如下

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="21" />
    <!-- 在SDCard中创建与删除文件权限 -->
    <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <!-- 往SDCard写入数据权限 -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <!-- 连接网络权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <application
```

```

    android:allowBackup="true"
    android:allowClearUserData="true"
    android:hardwareAccelerated="false"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>

```

## 2.1.6 创建三维地球实例

在 MainActivity.java 中添加如下代码：

在类 MainActivity 中定义三位球控件（文档中所有示例均以此命名）

```

//声明类型为 LSJGlobeControl 三位球控件的对象 mGlobeControl
private LSJGlobeControl mGlobeControl;
// ////////////////////////////////////////示例代码//////////////////////////////////////
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // ////////////////////////////////////////示例代码//////////////////////////////////////
    // 实例化三维球控件
    mGlobeControl = new LSJGlobeControl(this);
    // 用控件对象创建视图
    setContentView(mGlobeControl);
    // 用控件对象的 getDeviceKey 方法得到设备机器码
    String strDeviceKey = mGlobeControl.getDeviceKey(this);
    // 判断许可是否有效
    if (!mGlobeControl.isLicValid(this)) {
        // 无效的原因可能因为试用版许可过期了，如果出现过期，请及时联系工作人员
        if (mGlobeControl.isLicExpired(this)) {
        }
        // 注册许可，在添加三维球控件之后
    }
}

```

```
mGlobeControl.setSerialKey(this,
    "302429815#961K16775401601PS943JLJLJL53I8600C4000006008JL5189242UX6201549U591010018Q0JL
01699,0x1206b677,0x1346959f,0x131d3607,0x1346bb99,0xffff,0x0,0x1,0x1,0x1,0x1,0x1,0x1");
}
if (!mGlobeControl.isLicValid(this)) {
}
// //////////////////////////////////////
}
```

### 2.1.7 运行工程

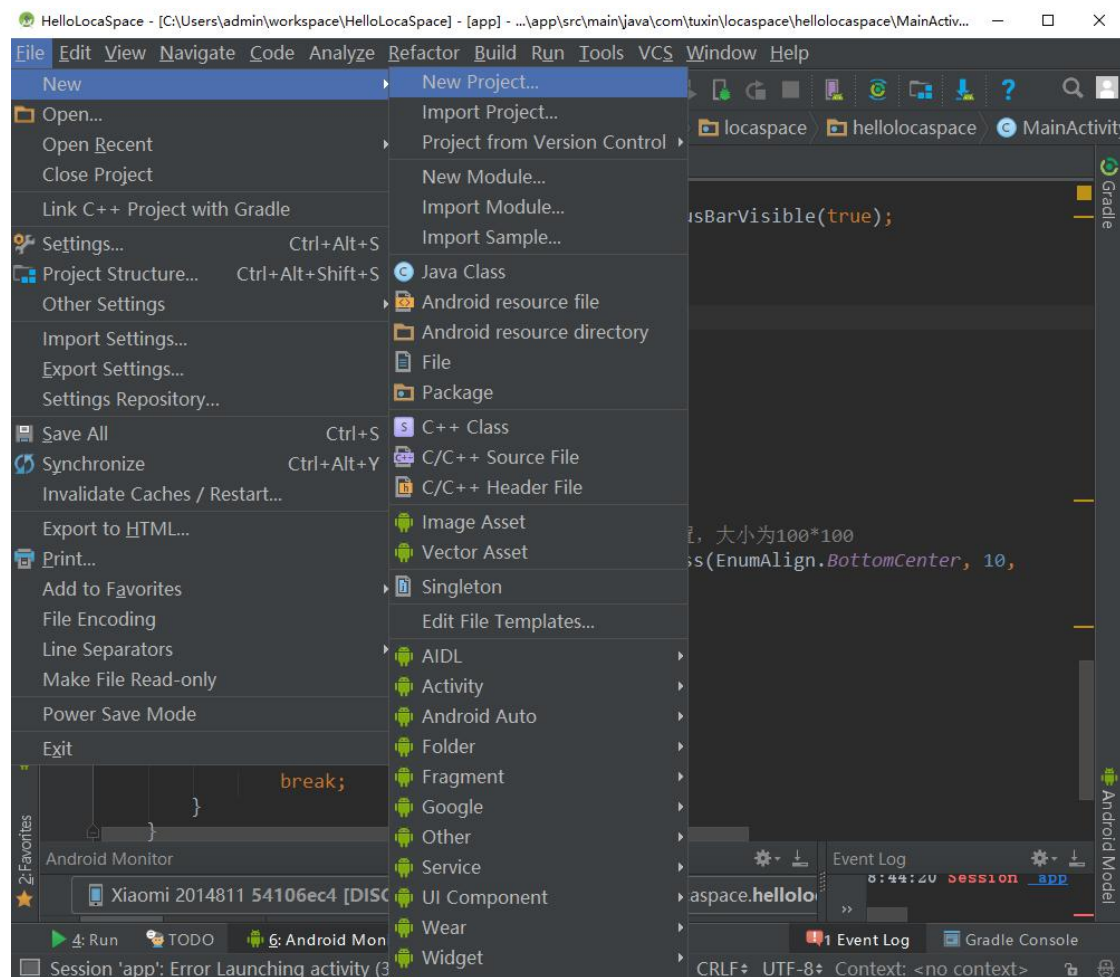
到此为止，一个三维地球就创建成功了。运行截图如下：



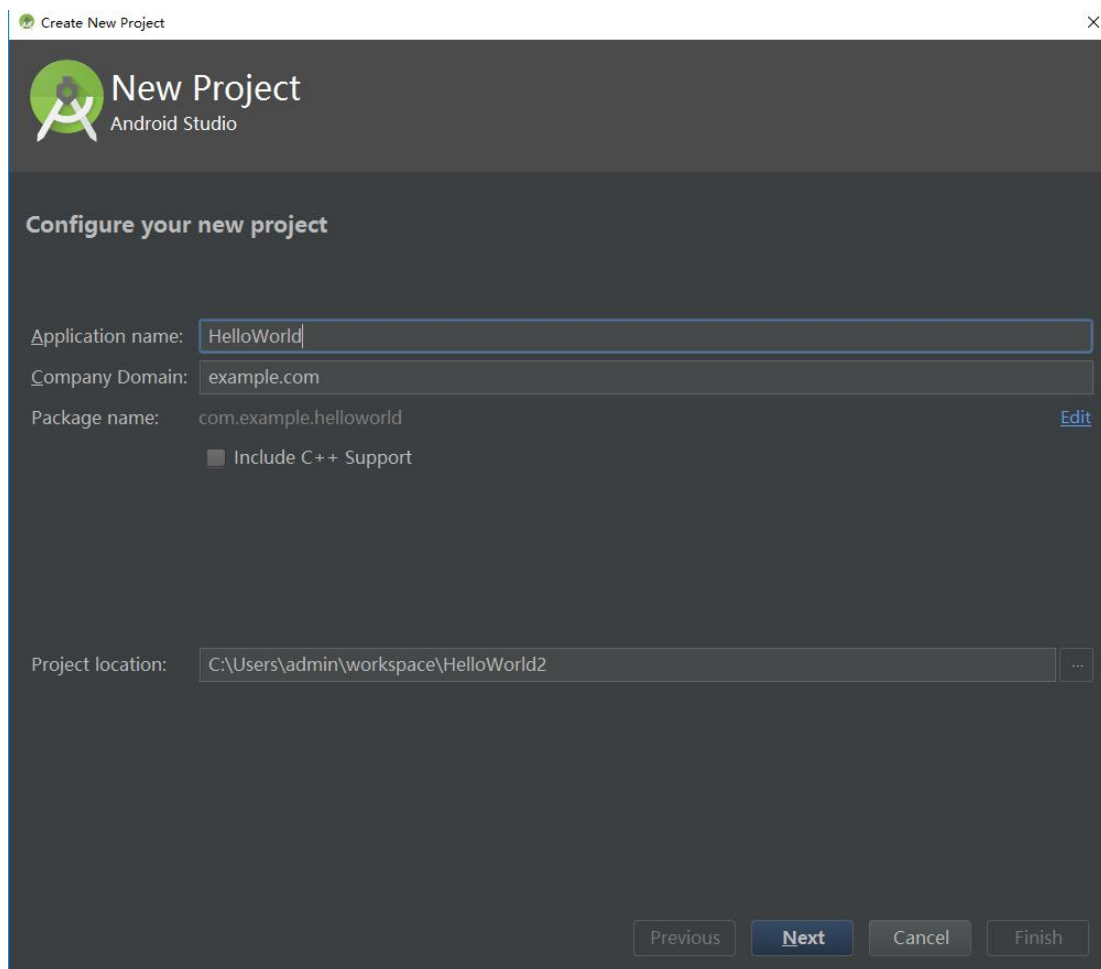
## 2.2 Android Studio 创建过程

### 2.2.1 新建工程

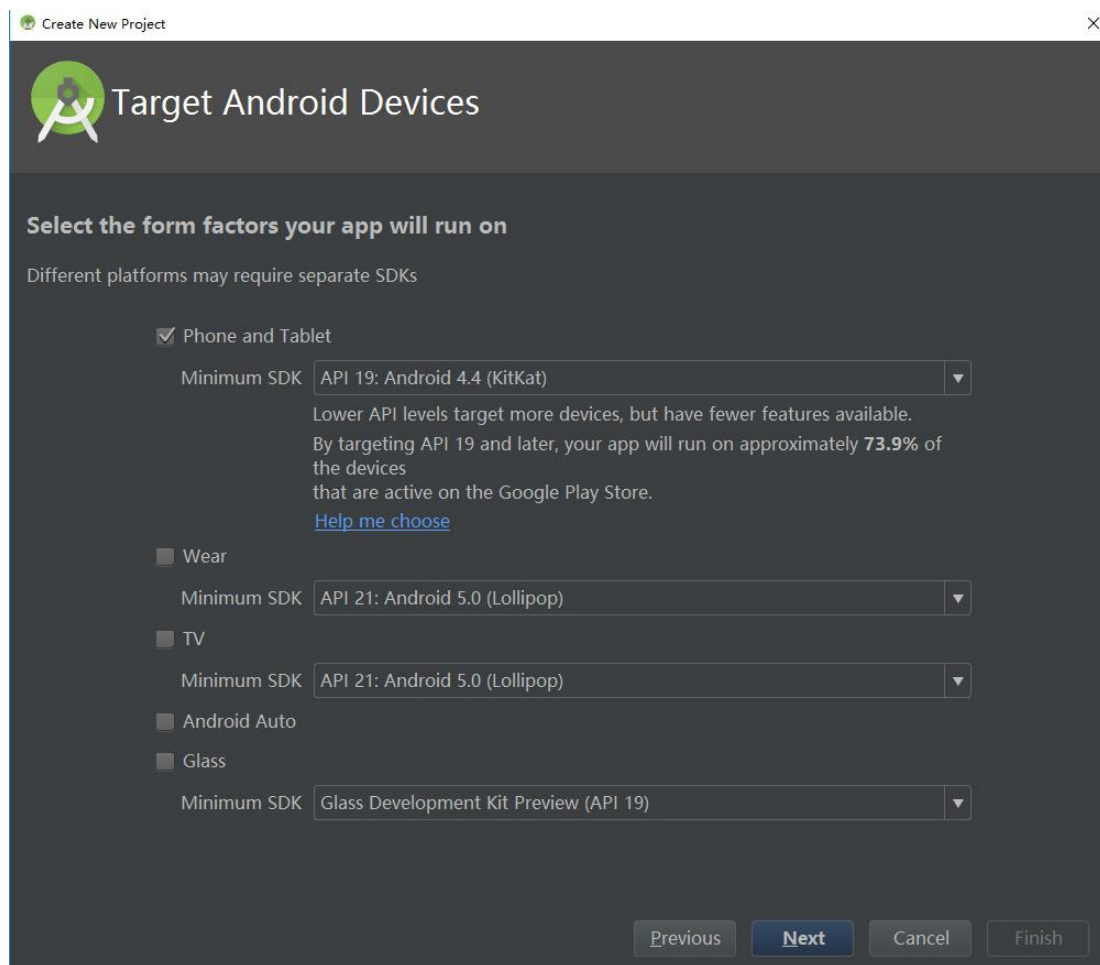
依次选择 File—New—New Project



创建工程名和工程路径

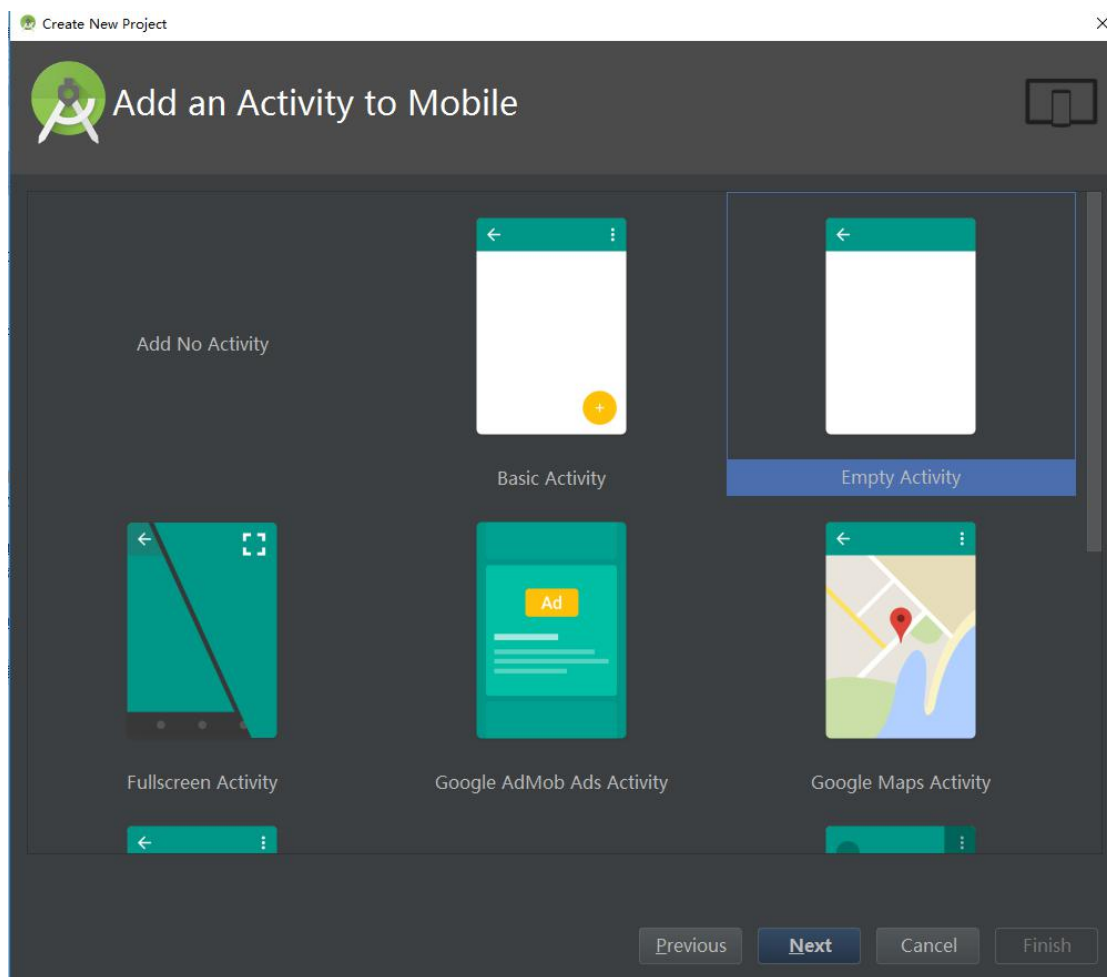


选择 sdk 版本

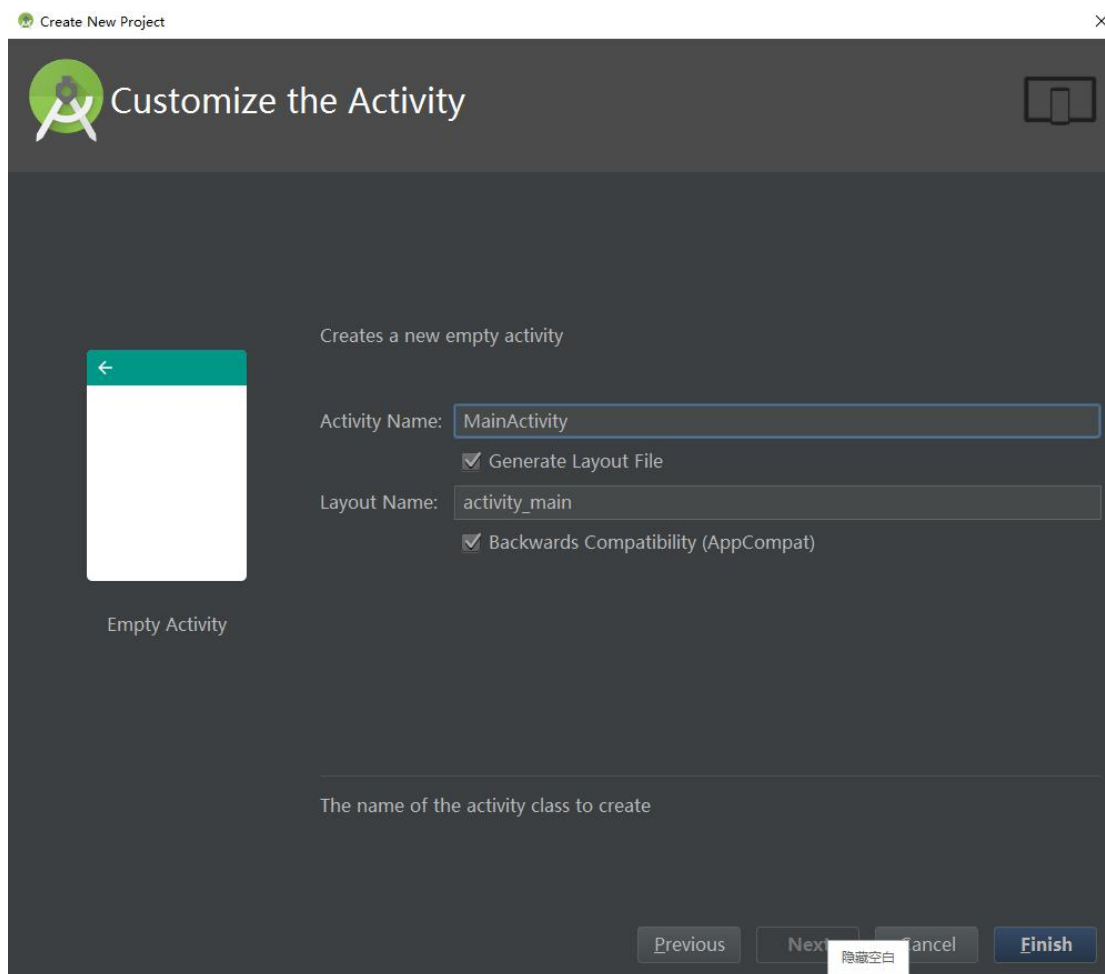


选择初始界面类型



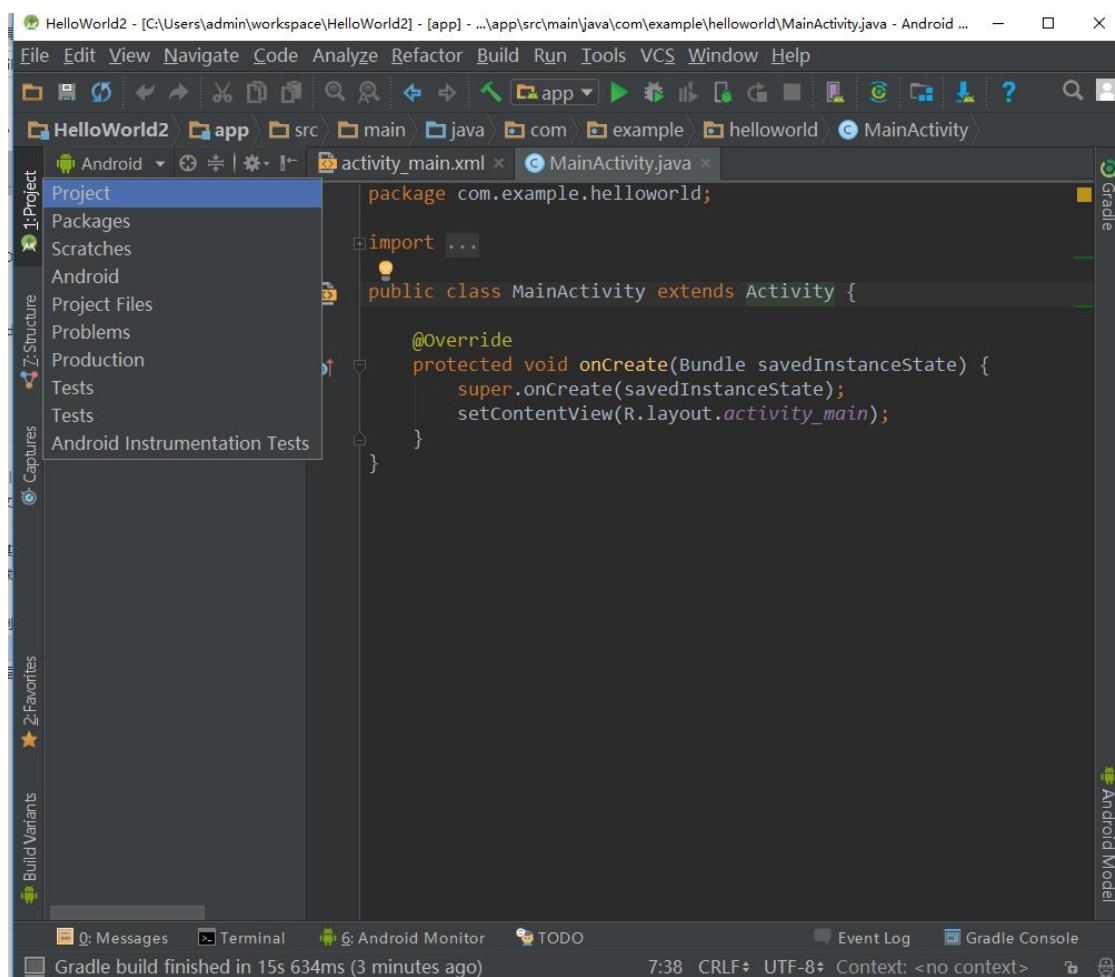


创建完成

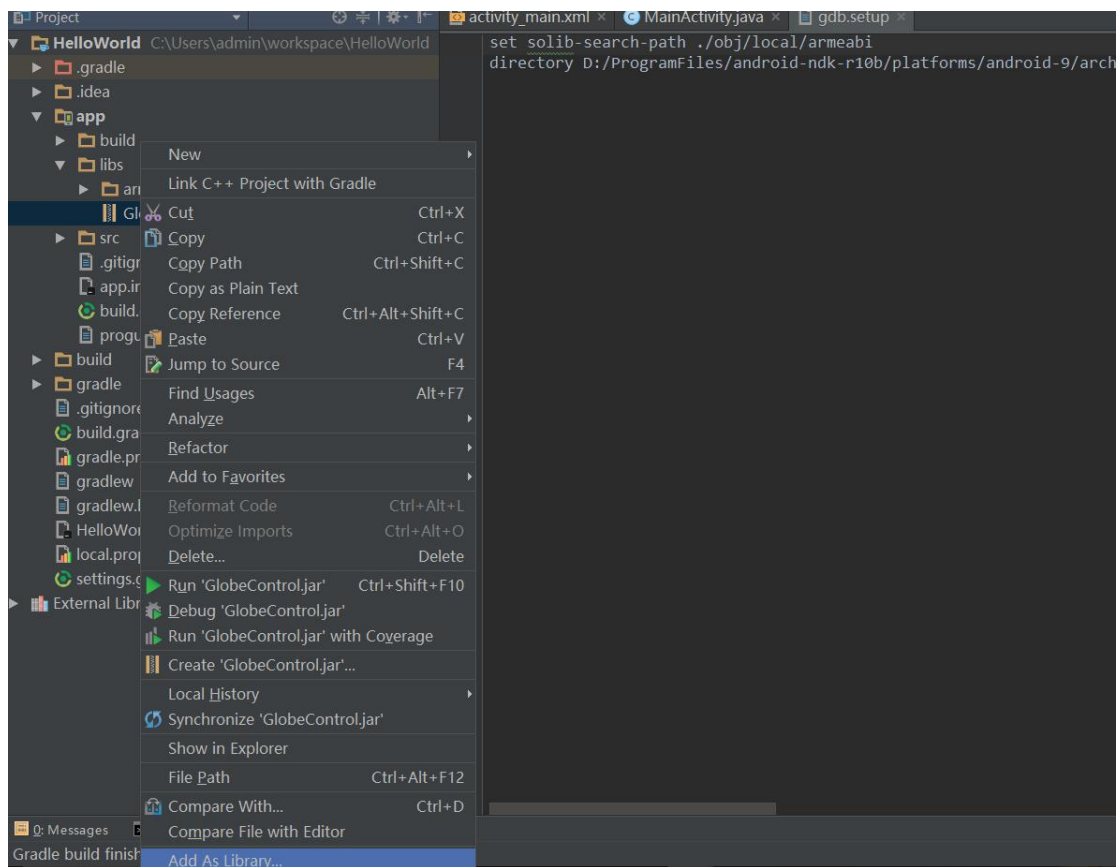


## 2.2.2 添加 jar 包（和 Eclipse 的方法不同）

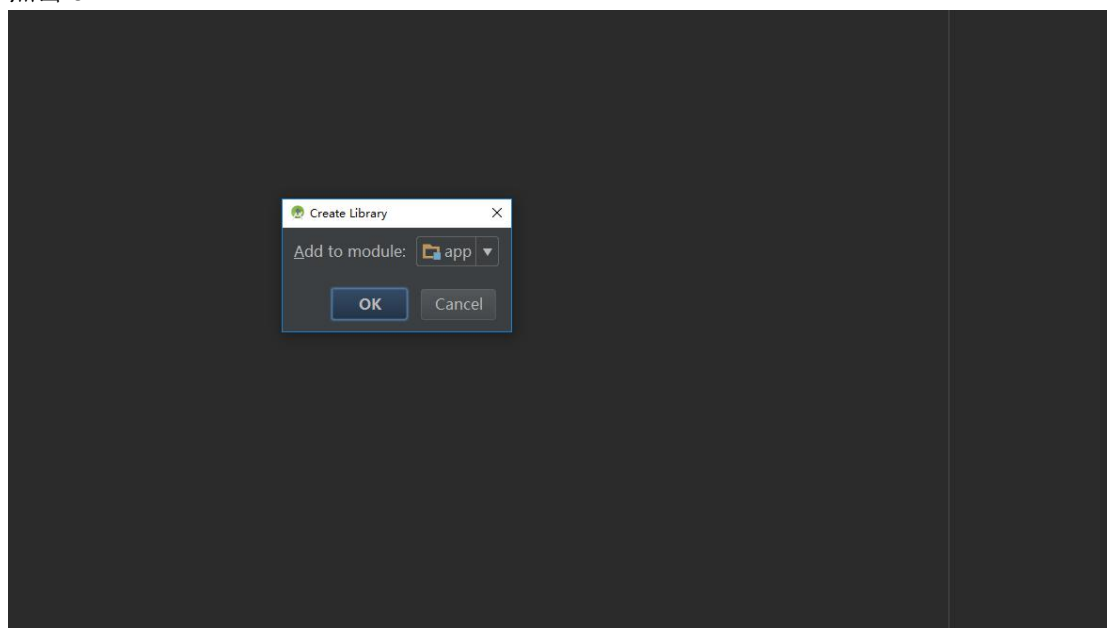
将 Android 模式改为 Project 模式



依次打开 HelloWorld—app 选项, 将开发包中的 GlobeControl 包和 armeabi 文件夹复制到 libs 文件夹下, 右键 GlobeControl 包, 点击 Add as Library 选项

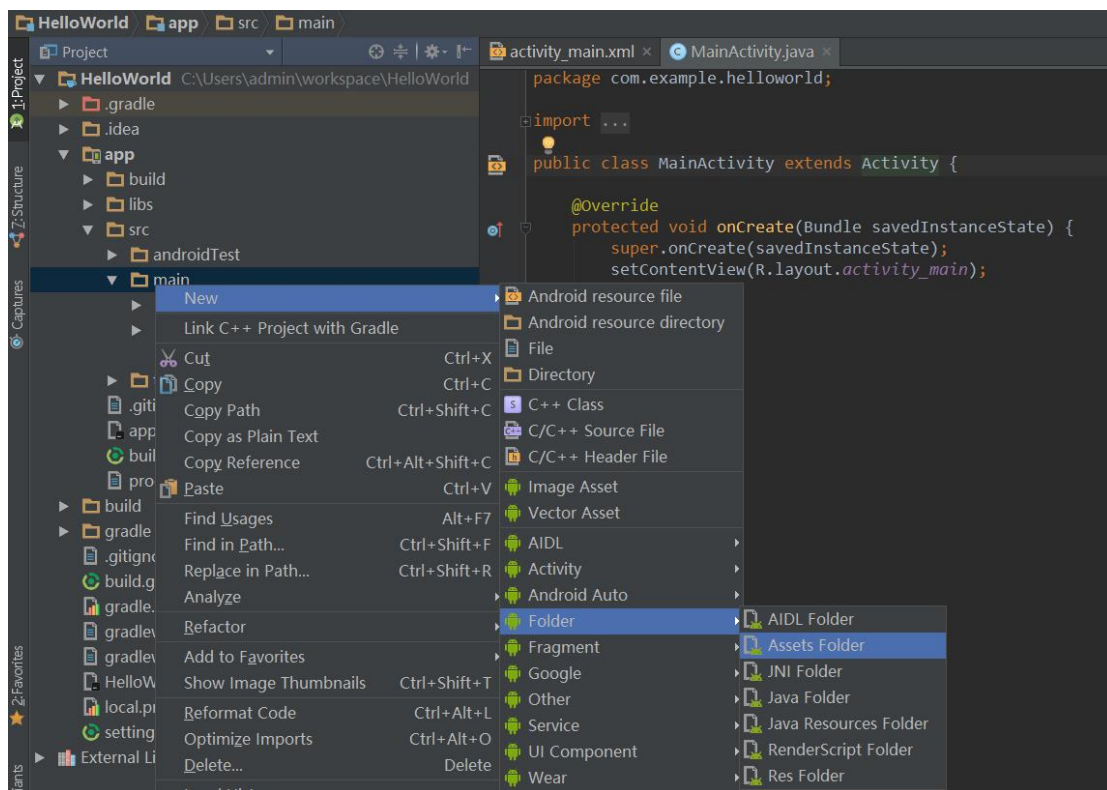


点击 ok

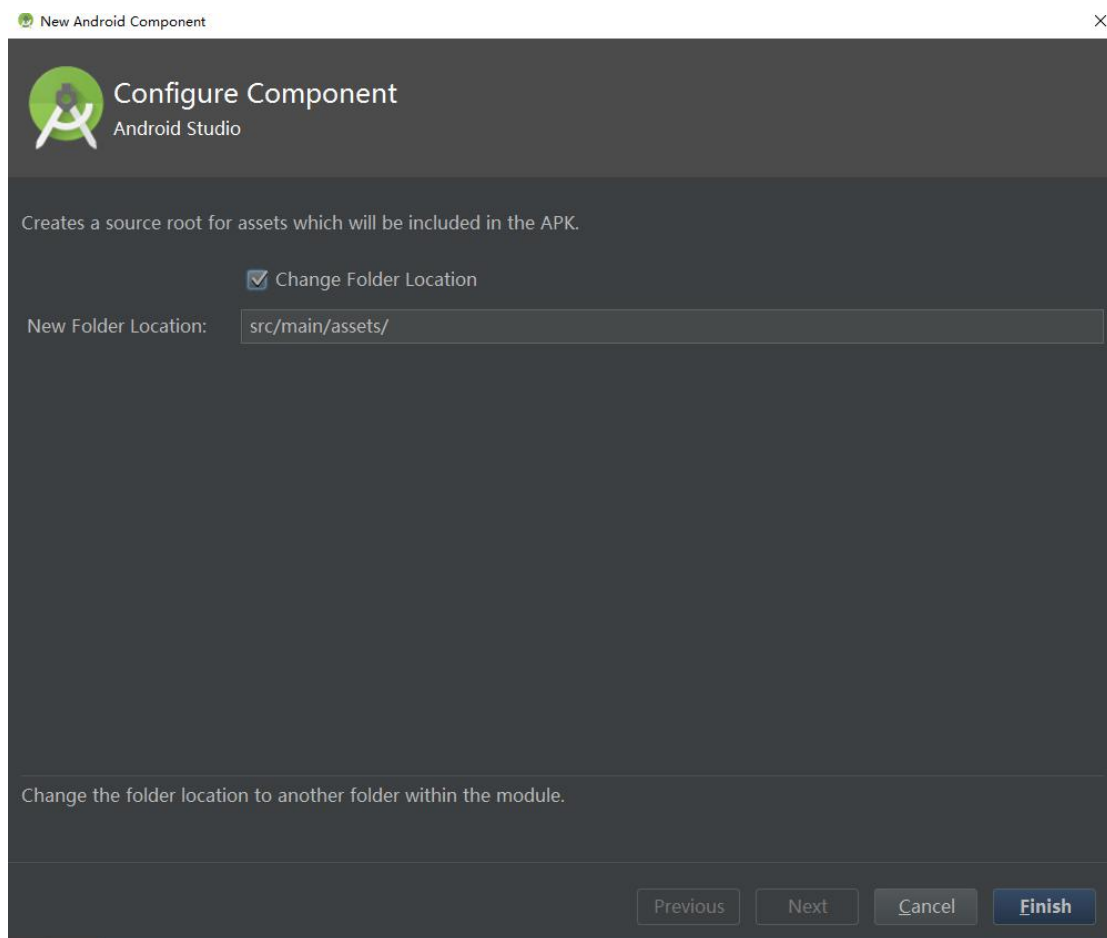


### 2.2.3 添加资源文件

依次打开 HelloWorld—app—src—main，右击 main 文件夹，选择 new—Folder—Assets Folder，

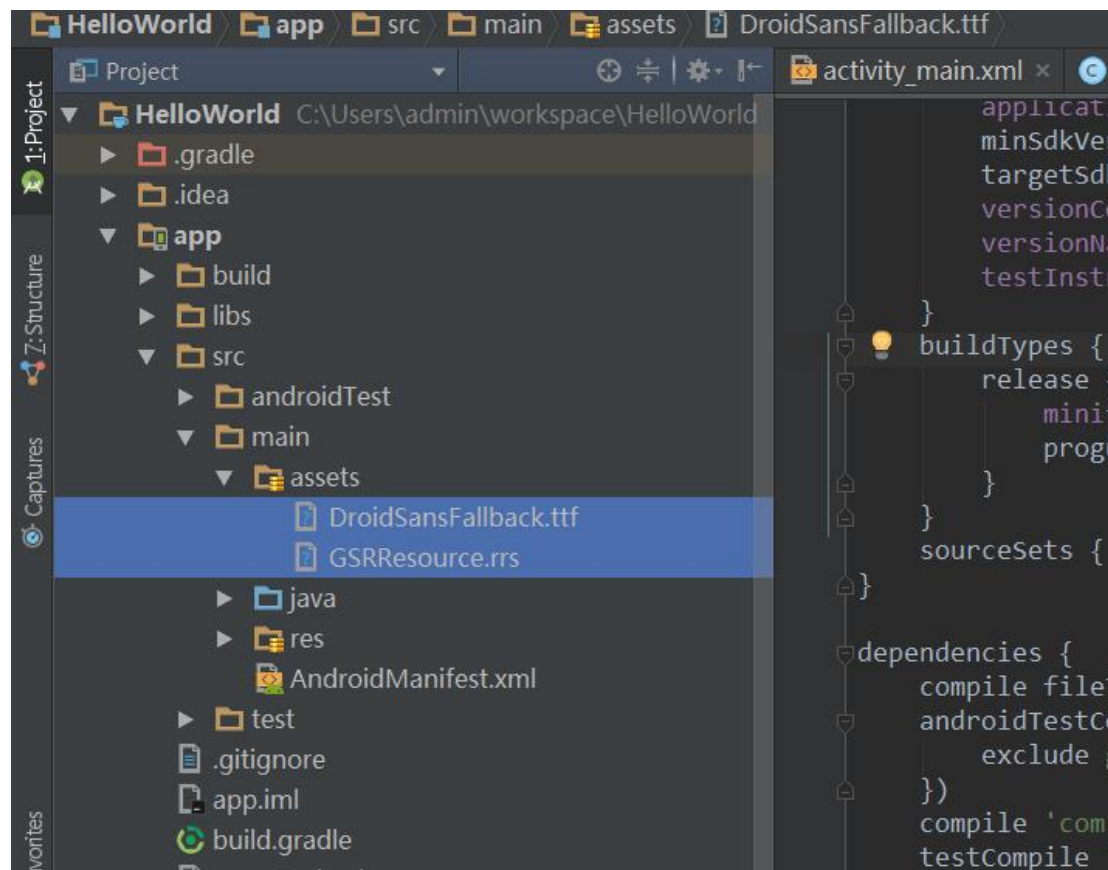


勾选 Change 选项（必须），点击 Finish



将开发包中的添加资源库文件 GSRResource.rrs 和 DroidSansFallback.ttf 文件复制到 assets 文

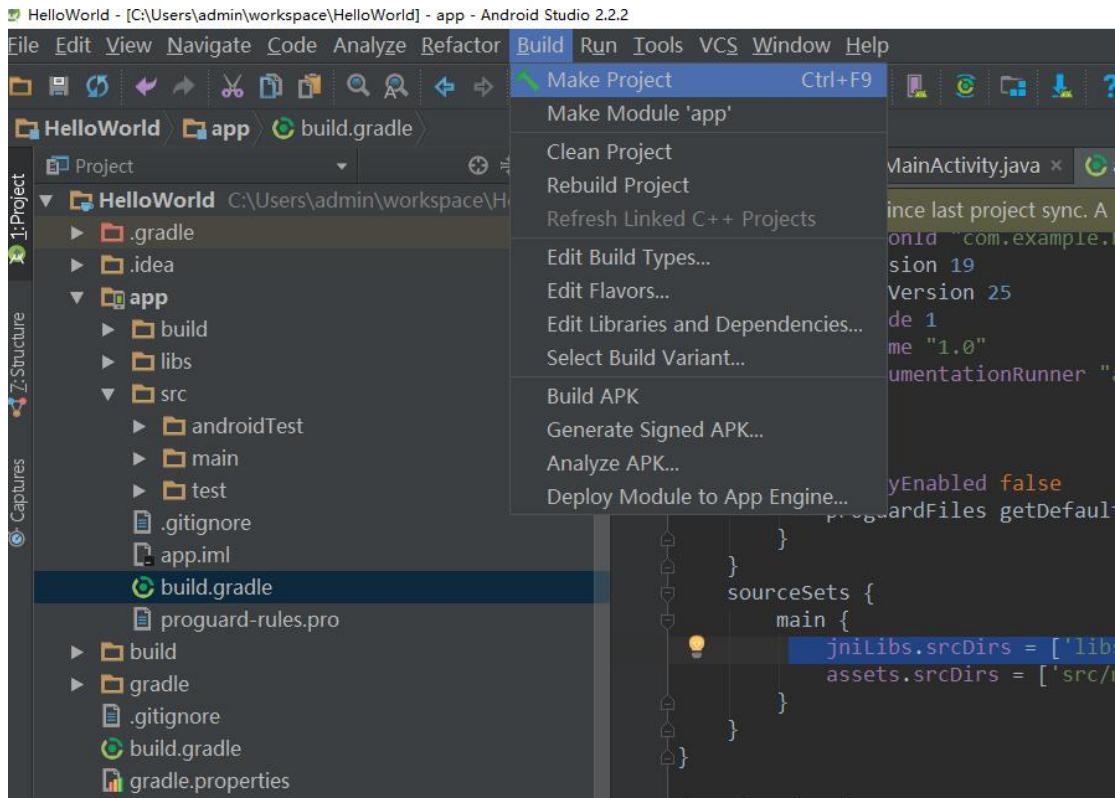
文件夹下



依次打开 `HelloWorld`—`app`—`src`—`build.gradle` 文件的 `sourceSets` 中,配置下列代码

```
sourceSets {
    main {
        jniLibs.srcDirs = ['libs']
        assets.srcDirs = ['src/main/assets', 'src/main/assets/']
    }
}
```

点击 `Build`—`Make Project` 选项, 配置完成。



## 2.2.4 配置权限

为工程添加权限和设置应用属性，如下

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="21" />
    <!-- 在SDCard中创建与删除文件权限 -->
    <uses-permission
        android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <!-- 往SDCard写入数据权限 -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    />
    <!-- 连接网络权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:allowClearUserData="true"
        android:hardwareAccelerated="false"

```

```

        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

## 2.2.5 创建三维球体示例

在 MainActivity.java 中添加如下代码：

在类 MainActivity 中定义三位球控件（文档中所有示例均以此命名）

```

public class MainActivity extends Activity {
//声明类型为 LSJGlobeControl 三位球控件的对象 mGlobeControl
    private LSJGlobeControl mGlobeControl;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
// 示例代码////////////////////////////////////
// 实例化三维球控件
        mGlobeControl = new LSJGlobeControl(this);
        setContentView(mGlobeControl);
//为引擎配置字库，Android Studio 必须添加
        mGlobeControl.setAppConfigPath(Environment.getExternalStorageDirectory().getAbsolutePath()
            + "/LocaSpace/DroidSansFallback.ttf");
// 用控件对象的 getDeviceKey 方法得到设备机器码
        String strDeviceKey = mGlobeControl.getDeviceKey(this);
// 判断许可是否有效
        if (!mGlobeControl.isLicValid(this)) {
            // 无效的原因可能因为试用版许可过期了，如果出现过期，请及时联系工作人员
        }
        if (mGlobeControl.isLicExpired(this)) {
            // 注册许可，在添加三维球控件之后
            mGlobeControl.setSerialKey(this,
                "302427121#Z2715197897603648669JLJL53I8600C400006008JL1217242UX20181312018999990218960017
                999969999340,0x1206abf1,0x1346959f,0x131d3607,0x1346bb99,0xffff,0x0,0x1,0x1,0x1,0x1,0x1,0x1"
            );
        }
    }
}

```



```
        if (!mGlobeControl.isLicValid(this)) {  
            }  
// ///////////////////////////////////////  
        }  
/////////////////////////////////////  
    }  
}
```

### 2.2.6 运行工程

到此为止，一个三维地球就创建成功了。运行截图如下：



## 2.3 开发进阶-添加图层和地形:

在原工程的基础之上，增加谷歌图层和谷歌地形，代码如下：

```

package com.example.helloworld;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import android.view.Menu;
import android.view.MenuItem;
import com.LocaSpace.Globe.EnumAltitudeMode;
import com.LocaSpace.Globe.LSJGlobe;
import com.LocaSpace.Globe.LSJGlobeControl;
import com.LocaSpace.Globe.LSJGlobeControl.ISurfaceCreatedEvent;
import com.LocaSpace.Globe.LSJLayers;

public class MainActivity extends Activity {

    // ////////////////示例代码//////////////////////
    // 声明类型为LSJGlobeControl三位球控件的对象mGlobeControl
    private LSJGlobeControl mGlobeControl;
    // ////////////////示例代码//////////////////////

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // ////////////////示例代码//////////////////////
        // 实例化三维球控件
        mGlobeControl = new LSJGlobeControl(this);
        // 用控件对象创建视图
        setContentView(mGlobeControl);
        // 用控件对象的getDeviceKey方法得到设备机器码
        String strDeviceKey = mGlobeControl.getDeviceKey(this);
        // 判断许可是否有效
        if (!mGlobeControl.isLicValid(this)) {
            // 无效的原因可能因为试用版许可过期了，如果出现过期，请及时联系工作人员
            if (mGlobeControl.isLicExpired(this)) {
            }
            // 注册许可，在添加三维球控件之后
            mGlobeControl
                .setSerialKey(this,
                    "302429815#961K16775401601PS943JLJL53I8600C4000006008JL5189242UX6201549U591010018Q0JL
                    01699,0x1206b677,0x1346959f,0x131d3607,0x1346bb99,0xffff,0x0,0x1,0x1,0x1,0x1,0x1,0x1");
        }
    }
}

```

```

        if (!mGlobeControl.isLicValid(this)) {
        }
        BackEvent();
        // //////////////////////////////////////
    }
    // //////////////////////////////////////
    /**
     * 后台事件，视图的创建和显示
     */
    private void BackEvent() {
        // 创建界面
        mGlobeControl.setOnSurfaceCreated(new ISurfaceCreatedEvent() {
            @Override
            public void onSurfaceCreated() {
                // 判断视图是否加载
                if (mGlobeControl != null && mGlobeControl.getGlobe() != null) {
                    // 跳转到对应位置，绝对地点，既初始视图
                    mGlobeControl.getGlobe().JumpTo(0, 0, 0, 0, 0, 19089420.91,
                        EnumAltitudeMode.Absolute);
                    // 添加本地图层、添加本地地形，lrc文件需要预先放入手机相应文件夹，lrc
                    文件可以从开发包找到，更多的lrc文件请联系工作人员
                    mGlobeControl
                        .getGlobe()
                        .getLayers()
                        .AddLayer(
                            Environment.getExternalStorageDirectory()
                                .getAbsolutePath()
                                +
"/LocaSpace/googleLayer203.208.46.128.lrc");
                    mGlobeControl
                        .getGlobe()
                        .getTerrains()
                        .AddTerrain(
                            Environment.getExternalStorageDirectory()
                                .getAbsolutePath()
                                +
"/LocaSpace/googleTerrain203.208.46.128.lrc");
                    mGlobeControl.getGlobe().setStatusBarVisible(true);
                }
            }
        });
    }
}

```

经过上述图层的添加，已经可以看到起伏的地形，高耸的山头，深深的沟，全球的卫星影像。如下图所示：



### 三、基本操作控制

三维地球的基本操作控制，介绍对三位地球的底图进行放大、缩小、旋转、平移等操作。具体工程请查阅示例工程的【LSV-BasicControl】



### 3.1 缩放

功能描述：缩放当前视图

方法：mGlobeControl.globeZoom(boolean)

boolean 值为 true 时为放大，false 时为缩小

mGlobeControl.stop() 停止当前动作

具体代码如下：

```
public class MainActivity extends Activity implements OnTouchListener {
// ////////////////////////示例代码//////////////////////////////////////
// 声明类型为 LSJGlobeControl 三位球控件的对象 mGlobeControl
private LSJGlobeControl mGlobeControl;
// ////////////////////////示例代码//////////////////////////////////////
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
// ////////////////////////示例代码//////////////////////////////////////
// 实例化三维球控件
mGlobeControl = new LSJGlobeControl(this);
// 设置标题栏
requestWindowFeature(Window.FEATURE_NO_TITLE);
// 用控件对象创建视图,用于加载三维球
setContentView(R.layout.activity_main);
}
```

```

        RelativeLayout globe = (RelativeLayout) findViewById(R.id.globe);
        globe.addView(mGlobeControl);
        // 初始化所有控件
        initView();
        BackEvent();
        // //////////////////////////////////////
    }

    private void initView() {
        findViewById(R.id.big).setOnTouchListener(this);
        findViewById(R.id.small).setOnTouchListener(this);
    }
    // //////////////////////////////////////
    /**
     * 后台事件，视图的创建和显示
     */
    private void BackEvent() {
        // 创建界面
        mGlobeControl.setOnSurfaceCreated(new ISurfaceCreatedEvent() {
            @Override
            public void onSurfaceCreated() {
                // 判断视图是否加载
                if (mGlobeControl != null && mGlobeControl.getGlobe() != null) {
                    // 跳转到对应位置，绝对地点，既初始视图
                    mGlobeControl.getGlobe().JumpTo(0, 0, 0, 0, 19089420.91,
                        EnumAltitudeMode.Absolute);
                    mGlobeControl.getGlobe().setStatusBarVisible(false);
                }
            }
        });
    }

    // 触碰事件
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        switch (v.getId()) {
            // 放大
            case R.id.big:
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        mGlobeControl.globeZoom(true);
                        break;
                    case MotionEvent.ACTION_UP:
                        mGlobeControl.stop();
                }
            default:
                break;
        }
    }

```

```
        break;
    default:
        break;
    }
    break;
// 缩小
case R.id.small:
    switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
        mGlobeControl.globeZoom(false);
        break;
    case MotionEvent.ACTION_UP:
        mGlobeControl.stop();
        break;
    default:
        break;
    }
    break;
    default:
    break;
}
return false;
}
}
```

## 3.2 旋转

功能描述：绕当前屏幕中线点旋转

方法：mGlobeControl.globeMov(**boolean**)

**boolean** 值为 **true** 时为右旋，**false** 时为左旋

mGlobeControl.stop()

停止当前动作

具体代码如下：

```
// 左旋
case R.id.left:
    switch (event.getAction()) {
        // 手指按下时, 下同
        case MotionEvent.ACTION_DOWN:
            mGlobeControl.globeMove(false);
            break;
        // 手指抬起时, 下同
        case MotionEvent.ACTION_UP:
            mGlobeControl.stop();
            break;
        default:
            break;
    }
    break;

// 右旋
case R.id.right:
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            mGlobeControl.globeMove(true);
            break;
        case MotionEvent.ACTION_UP:
            mGlobeControl.stop();
            break;
        default:
            break;
    }
    break;
```

### 3.3 翻转

功能描述：绕当前屏幕中线点翻转

方法：mGlobeControl.globeTilt(**boolean**)

**boolean** 值为 true 时为上翻，false 时为下翻

mGlobeControl.stop()

停止当前动作

具体代码如下：

```
// 上旋
case R.id.up:
```



```
switch (event.getAction()) {
case MotionEvent.ACTION_DOWN:
    mGlobeControl.globeTilt(true);
    break;
case MotionEvent.ACTION_UP:
    mGlobeControl.stop();
    break;
default:
    break;
}
break;
// 下旋
case R.id.down:
    switch (event.getAction()) {
case MotionEvent.ACTION_DOWN:
        mGlobeControl.globeTilt(false);
        break;
case MotionEvent.ACTION_UP:
        mGlobeControl.stop();
        break;
default:
        break;
}
break;
```

### 3.4 飞行



具体工程请查阅示例工程的【LSV-FlyTo】

功能描述：从当前点飞行到目标点

方法：`mGlobeControl.getGlobe().FlyTo(dLon,dLat,dAlt,dHeading,dTilt,dDistance,intAltMode);`

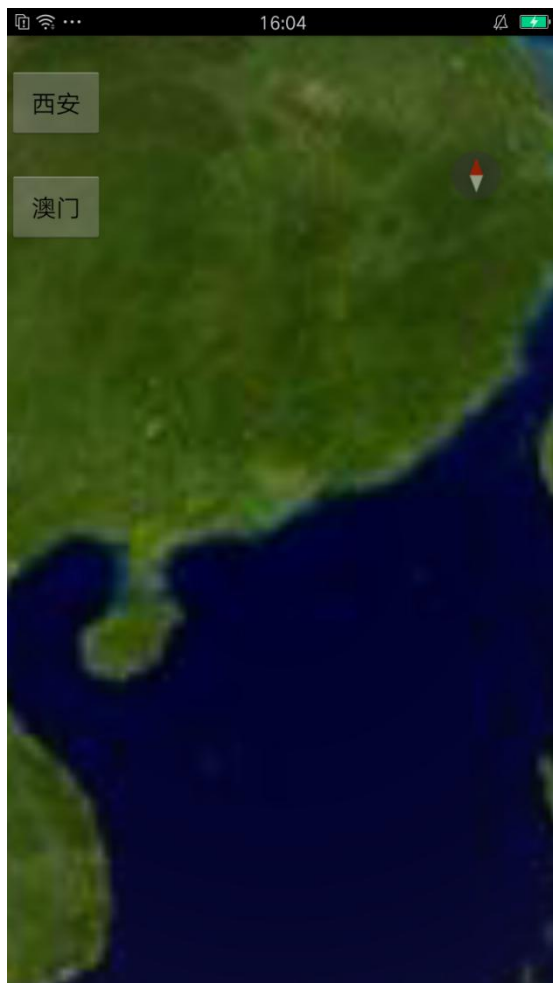
具体代码如下：

```
public class MainActivity extends Activity implements OnClickListener {  
    // ////////////////示例代码////////////////////  
    // 声明类型为 LSJGlobeControl 三位球控件的对象 mGlobeControl  
    private LSJGlobeControl mGlobeControl;  
    // ////////////////示例代码////////////////////  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // ////////////////示例代码////////////////////  
        // 实例化三维球控件  
        mGlobeControl = new LSJGlobeControl(this);  
        // 设置标题栏  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
    }  
}
```

```
// 用控件对象创建视图,用于加载三维球
setContentView(R.layout.activity_main);
RelativeLayout globe = (RelativeLayout) findViewById(R.id.globe);
globe.addView(mGlobeControl);
// 初始化所有控件
initWidget();
BackEvent();
}
private void initWidget() {
    findViewById(R.id.beijing).setOnClickListener(this);
    findViewById(R.id.hongkong).setOnClickListener(this);
}
/**
 * 后台事件, 视图的创建和显示
 */
private void BackEvent() {
    // 创建界面
    mGlobeControl.setOnSurfaceCreated(new ISurfaceCreatedEvent() {
        @Override
        public void onSurfaceCreated() {
            // 判断视图是否加载
            if (mGlobeControl != null && mGlobeControl.getGlobe() != null) {
                // 跳转到对应位置, 绝对地点, 既初始视图
                mGlobeControl.getGlobe().JumpTo(0, 0, 0, 0, 0, 19089420.91,
                    EnumAltitudeMode.Absolute);
                // 添加本地图层、添加本地地形, lrc 文件需要预先放入手机相应文件夹, lrc 文件
                // 可以从开发包找到, 更多的 lrc 文件请联系工作人员
                mGlobeControl.getGlobe().getLayers().AddLayer(
                    Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/LocaSpace/googleLayer203.208.46.128.lrc");
                mGlobeControl.getGlobe().getTerrains().AddTerrain(
                    Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/LocaSpace/googleTerrain203.208.46.128.lrc");
                mGlobeControl.getGlobe().setStatusBarVisible(false);
            }
        }
    });
}
// 点击事件
@Override
public void onClick(View v) {
    switch (v.getId()) {
        // 跳转到北京
        case R.id.beijing:
```

```
// 创建一个点的对象
    LSJPoint3d point = new LSJPoint3d();
    point.setValue(116.403875, 39.915168, 3000000);
    // 设置飞行速度
    mGlobeControl.getGlobe().setFlyToPointSpeed(60000);
    // 飞行到设定好的点的位置
    mGlobeControl.getGlobe().flyToPosition(point,
        EnumAltitudeMode.Absolute);
    break;
// 跳转到香港
case R.id.hongkong:
    mGlobeControl.getGlobe().setFlyToPointSpeed(60000);
    // 直接飞行到点的坐标
    mGlobeControl.getGlobe().flyTo(114, 22, 2000000, 0, 0, 0,
        EnumAltitudeMode.RelativeToGround);
    break;
default:
    break;
}
}
}
```

### 3.5 跳转



具体工程请查阅示例工程的【LSV-JumpTo】

功能描述：从当前点跳转到目标点，没有飞行过程

方法：`mGlobeControl.getGlobe().JumpTo(dLon,dLat,dAlt,dHeading,dTilt,dDistance,intAltMode);`

具体代码如下：

```
public class MainActivity extends Activity implements OnClickListener {
    // ////////////////示例代码////////////////////
    // 声明类型为 LSJGlobeControl 三位球控件的对象 mGlobeControl
    private LSJGlobeControl mGlobeControl;
    // ////////////////示例代码////////////////////
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // ////////////////示例代码////////////////////
        // 实例化三维球控件
        mGlobeControl = new LSJGlobeControl(this);
    }
}
```

```
// 设置标题栏
requestWindowFeature(Window.FEATURE_NO_TITLE);
// 用控件对象创建视图,用于加载三维球
setContentView(R.layout.activity_main);
RelativeLayout globe = (RelativeLayout) findViewById(R.id.globe);
globe.addView(mGlobeControl);
// 初始化所有控件
initWidget();
BackEvent();
}
private void initWidget() {
    findViewById(R.id.xian).setOnClickListener(this);
    findViewById(R.id.aomen).setOnClickListener(this);
}
/**
 * 后台事件, 视图的创建和显示
 */
private void BackEvent() {
    // 创建界面
    mGlobeControl.setOnSurfaceCreated(new ISurfaceCreatedEvent() {
        @Override
        public void onSurfaceCreated() {
            // 判断视图是否加载
            if (mGlobeControl != null && mGlobeControl.getGlobe() != null) {
                // 跳转到对应位置, 绝对地点, 既初始视图
                mGlobeControl.getGlobe().JumpTo(0, 0, 0, 0, 0, 19089420.91,
                    EnumAltitudeMode.Absolute);
                // 添加本地图层、添加本地地形, lrc 文件需要预先放入手机相应文件夹, lrc 文件
                // 可以从开发包找到, 更多的 lrc 文件请联系工作人员
                mGlobeControl.getGlobe().getLayers().AddLayer(
                    Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/LocaSpace/googleLayer203.208.46.128.lrc");
                mGlobeControl.getGlobe().getTerrains().AddTerrain(
                    Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/LocaSpace/googleTerrain203.208.46.128.lrc");
                mGlobeControl.getGlobe().setStatusbarVisible(false);
            }
        }
    });
}
@Override
public void onClick(View v) {
    switch (v.getId()) {
        // 跳转到西安
        case R.id.xian:
```

```

        mGlobeControl.getGlobe().JumpTo(108.56, 34.15, 2000000,
            0, 0, 0, EnumAltitudeMode.RelativeToGround);
        break;
// 跳转到澳门
case R.id.aomen:
    mGlobeControl.getGlobe().JumpTo(113.5, 22.2, 2000000,
        0, 0, 0, EnumAltitudeMode.RelativeToGround);
    break;
default:
    break;
}
}
}

```

### 3.6 设置视图状态

具体工程请查阅示例工程的【LSV-FlyTo】【LSV-JumpTo】【LSV-CameraState】

功能描述：LocaSpace 提供了对 Camera 的属性进行访问的接口，可以获取或修改 Camera 的参数，保存或改变当前的场景。Camera 的属性有：

具体代码如下：

```

mGlobeControl.getGlobe().getCameraState().setLatitude(value);    //视点的经度
mGlobeControl.getGlobe().getCameraState().setLongitude(value);  //视点的纬度
mGlobeControl.getGlobe().getCameraState().setAltitude(value);   //视点离地面的垂直距离
mGlobeControl.getGlobe().getCameraState().setHeading(value);    //视线的方向与正北的夹角，0
度表示正北，90 度表示正东，180 度表示正南
mGlobeControl.getGlobe().getCameraState().setTilt(value);       //视线与铅垂线的夹角，0 度表示垂
直向下看，90 度表示沿水平方向看
mGlobeControl.getGlobe().getCameraState().setAltitudeMode(enum);//高度模式，支持 3 种模
式，绝对高度、相对地面高度、贴地。

```

### 3.7 设置指南针样式

具体工程请查阅示例工程的【LSVDemo-Compass】

功能描述：LocaSpace 提供了对指南针的属性进行设置

方法：mGlobeControl.getGlobe().setNaviCompass(EnumAlign align, float offsetX, float offsetY, float width, float height, boolean bPixelOffset, boolean bPixelSize);

参数说明：EnumAlignalign，是指南针以屏幕哪个位置对齐

float offsetX, offsetY 是指南针区域起始点相对 align 位置的偏移量

float width, height 是指南针区域的大小,0 和负数的话表示取纹理原始像素

大小

boolean bPixelOffset 表示偏移量是像素大小还是比例大小，true 表示像素大

小

boolean bPixelSize 表示指南针区域是像素大小还是比例大小，true 表示像素大小

具体代码：

```
// 指南针位于底部居中向右 10 个像素的位置，大小为 100*100
mGlobeControl.getGlobe().SetNaviCompass(EnumAlign.BottomCenter, 10, 0, 100, 100, true, true);
mGlobeControl.getGlobe().refresh();//球体刷新
```

## 四、图层

LSViewer Android SDK 支持多种格式的图层，包括在线图层，本地图层，矢量数据、栅格数据、地形数据等等。

具体工程请查阅示例工程的【LSV-LayerManager】，【LSV-LayerErgodic】

### 4.1 几个概念：

**在线图层：** 在线图层指地图瓦片数据是从互联网上获取的地图图层。这样的图层数据可以通过配置 lrc 文件的方式来支持各种在线地图服务。此类数据的使用需要连接互联网。

**本地图层：** 指把地图数据直接放在手机存储空间的数据，此类数据的使用可以不用连接互联网。

**栅格数据：** 指以真彩色形式存储的地图数据，具体表现形式包括：tif, img, lrp 等。

**Lrp：** 是 LSV 独有的一种自带分级的栅格数据图源，

**地形数据：** 常见的用于创建三维地形显示效果的地形数据，多为 tif, grd 等格式。

**矢量数据：** lgd, kml, shp, gpx 等以矢量坐标形式存储的地图数据。

**模型数据：** 指常见的 3ds, obj, osgb, osg 等三维模型数据，gcm 是 LSV 所独有的模型文件格式，比传统的 obj, 3ds 等格式的数据体积小，加载速度快。需要通过 builder 工具对数据格式进行转换，可以把 obj 格式的数据转换为 gcm。

对于图层可以控制其显示、隐藏、可见距离、不透明度、渲染顺序、格式转换等等，具体接口如下：

### 4.2 添加图层：

功能描述：添加一个图层到三维地球

具体代码如下：

```
mGlobeControl.getGlobe().getLayers().AddLayer(layerPath);
```

### 4.3 删除图层

功能描述：添加一个图层到三维地球

具体代码如下：



```
//通过图层名删除图层
mGlobeControl.getGlobe().getLayers().RemoveLayerByName(strName);
//通过标题删除图层
mGlobeControl.getGlobe().getLayers().RemoveLayerByCaption(strCaption);
//通过 ID 删除图层
mGlobeControl.getGlobe().getLayers().RemoveLayerByID(nID);
//通过索引删除图层
mGlobeControl.getGlobe().getLayers().RemoveLayerByIndex(nIndex);
```

## 4.4 图层显示

功能描述：修改图层从不可见状态到可见状态

具体代码如下：

```
// 显示图层
mGlobeControl.getGlobe().getLayers().GetLayerByName(layerPath).setVisible(true);
```

## 4.5 图层隐藏

功能描述：隐藏一个图层

具体代码如下：

```
//隐藏图层
mGlobeControl.getGlobe().getLayers().GetLayerByName(layerPath).setVisible(false);
```

## 4.6 图层可见距离

功能描述：设置图层可见距离，超过此范围，则图层不可见

具体代码如下：

```
//设置最大可见距离
mGlobeControl.getGlobe().getLayers().GetLayerByName(layerPath).setObjectMaxVisibleDistance(
dVal);
//设置最小可见距离
mGlobeControl.getGlobe().getLayers().GetLayerByName(layerPath).setObjectMinVisibleDistance(
dVal);
```

## 4.7 图层不透明度

功能描述：控制图层的不透明度

具体代码如下：

```
//设置图层不透明度
mGlobeControl.getGlobe().getLayers().GetLayerByName(layerPath).setOpaque(fVal);
```

## 4.8 图层渲染顺序

功能描述：设置图层在三维球体上渲染的顺序

参数说明：index：图层序号，表示移动的是第几个图层

具体代码如下：

```
//设置图层上移
mGlobeControl.getGlobe().getLayers().MoveUp(index);
//设置图层下移
mGlobeControl.getGlobe().getLayers().MoveDown(index);
```

## 4.9 图层范围

功能描述：获取图层的位置范围，包括经纬度，

具体代码如下：

```
//得到图层范围
mGlobeControl.getGlobe().getLayers().GetLayerByName(layerPath).getBounds();
double x=bounds.getCenter().getX();得到图层中心点的经度坐标
double y=bounds.getCenter().getY();得到图层中心点的纬度坐标
```

## 4.10 图层类型

### 4.10.1 影像数据

LocaSpace 中支持的影像数据格式包括：tif、img 等，另外 LocaSpace 的原始影像数据格式是 LRP，LRP 是一种高压缩比的二进制栅格文件，可以使用 LocaBuilder 这个程序把 tif、img 等数据转换成 LRP，LRP 内建了金字塔文件，读取速度更快。

```
mGlobeControl.getGlobe().getLayers().AddLayer("/mnt/sdcard/LocaSpace/AppData/beijing.lrp");
```

### 4.10.2 地形数据

LocaSpace 中支持的地形数据格式包括：tif、img 等，另外 LocaSpace 的原始地形数据格式是 LRP，LRP 是一种高压缩比的二进制文件，可以使用 LocaBuilder 这个程序把 tif、img 等数据转换成 LRP，LRP 内建了金字塔文件，读取速度更快。

```
mGlobeControl.getGlobe().getTerrains().AddTerrain("/mnt/sdcard/LocaSpace/AppData/DEM.lrp")
```

### 4.10.3 矢量数据

目前 LocaSpace 对矢量的支持情况为：可以直接支持 lgd 文件，支持 shp。

（一）Lgd 格式数据

```
mGlobeControl.getGlobe().getLayers().AddLayer("/mnt/sdcard/LocaSpace/AppData/GlobeBkIma
```

```
ge.lgd");
```

## （二）Shapefile 格式数据

LocaSpace 可以直接加载经纬度的 Shapefile 矢量数据。

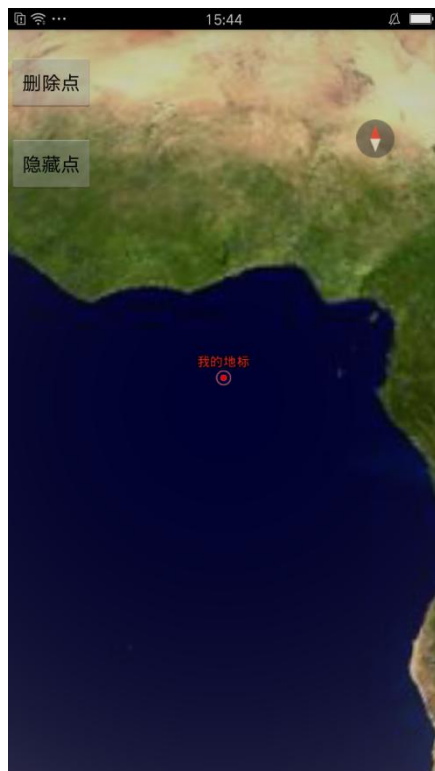
```
mGlobeControl.getGlobe().getLayers().AddLayer("/mnt/sdcard/LocaSpace/AppData/pipeline.shp");
```

## 4.11 图层遍历

```
//获取所有图层  
LSJLayers layers = mGlobeControl.getGlobe().getLayers();  
//图层数量  
int length = layers.GetCount();  
//遍历图层  
for (int i = 0; i < count; i++) {  
    LSJLayer layer = layers.GetLayerByIndex(i)  
}
```

## 五、点、线、面标注：

### 5.1 添加点



具体工程请查阅示例工程的【LSV-Marker】

功能描述：添加点

具体代码如下：

```
//添加点
LSJFeature feature = new LSJFeature();
LSJGeoMarker marker = new LSJGeoMarker();
// 位置
double x = mGlobeControl.getGlobe().getCameraState().getLongitude();
double y = mGlobeControl.getGlobe().getCameraState().getLatitude();
double z = 0;
marker.setPosition(x, y, z);
marker.setAltitudeMode(EnumAltitudeMode.RelativeToGround);
feature.setGeometry(marker);
feature.setDescription("我的地标");
feature.setName("地标 1");
// 添加到图层
mGlobeControl.getGlobe().getMemoryLayer().addFeature(feature);
mGlobeControl.getGlobe().refresh();
mGlobeControl.getGlobe().flyToFeature(feature);
```

## 5.2 点样式设置

具体工程请查阅示例工程的【LSV-Marker】

功能描述：设置点的样式

具体代码如下：

```
LSJGeoMarker marker = new LSJGeoMarker();
LSJMarkerStyle3D style = new LSJMarkerStyle3D();
style.setTextVisible(true); // 是否显示文字
LSJTextStyle textStyle = new LSJTextStyle(); // 显示文字大小
textStyle.setFontSize(20); // 显示文字位置
textStyle.setAlignment(EnumAlign.TopCenter); // 显示是否有下划线
textStyle.setUnderlined(true); // 显示文字颜色
textStyle.setForeground("#CD2626");
style.setTextStyle(textStyle);
// 显示图标
style.setIconPath(Environment.getExternalStorageDirectory()
    .getAbsolutePath() + "/LocaSpace/" + "marker.gif"); // 图标的路径
style.setIconScale(10); // 图标的缩放：1代表原本大小
style.setIconVisible(true); // 是否显示图标
marker.setStyle(style); // 为图标添加样式
```

## 5.3 添加线



具体工程请查阅示例工程的【LSV-Line】

功能描述：添加线

具体代码如下：

```
//添加线
LSJGeoPolyline3D line = new LSJGeoPolyline3D(); // 创建线对象
LSJPoint3d[] pnts = new LSJPoint3d[4]; // 创建点数组
pnts[0] = new LSJPoint3d(116.6, 9.9, 1000); // 把点添加到点数组中
pnts[1] = new LSJPoint3d(16.62, 9.9, 3000);
pnts[2] = new LSJPoint3d(16.62, 45, 2000);
pnts[3] = new LSJPoint3d(116.6, 45, 2500);
line.addPart(pnts); // 把点数组添加到线上
// 创建几何对象并设置属性
LSJFeature f = new LSJFeature();
f.setGeometry(line); // 把线对象添加到几何对象上
f.setName("线 01"); // 设置几何对象的名称
// 把几何要素添加到内存图层中
mGlobeControl.getGlobe().getMemoryLayer().addFeature(f);
mGlobeControl.getGlobe().refresh();
mGlobeControl.getGlobe().flyToFeature(f); // 刷新场景
```

## 5.4 线样式设置

具体工程请查阅示例工程的【LSV-Line】

功能描述：设置线的样式

具体代码如下：

```
LSJGeoPolyline3D line = new LSJGeoPolyline3D(); // 创建线
    LSJSimpleLineStyle3D style = new LSJSimpleLineStyle3D(); // 创建线的风格
        style.setLineColor("#CD2626"); // 显示线的颜色为黑色
        style.setLineWidth(3); // 设置线宽
        line.setStyle(style); // 把风格添加到线上
```

## 5.5 添加面



具体工程请查阅示例工程的【LSV-Polygon】

功能描述：添加一个面

具体代码如下：

```
//添加面
LSJGeoPolygon3D geoPolygon = new LSJGeoPolygon3D(); // 创建多边形对象
// 创建点数组对象
```

```

LSJPoint3d[] polygonPnts = new LSJPoint3d[4];
polygonPnts[0] = new LSJPoint3d(116.7, 9.8, 1000);
polygonPnts[1] = new LSJPoint3d(16, 9.8, 2000);
polygonPnts[2] = new LSJPoint3d(16, 39.8, 3000);
polygonPnts[3] = new LSJPoint3d(116.7, 39.8, 6000);
geoPolygon.addPart(polygonPnts); // 把点数组添加到多边形对象上
// 创建几何对象并设置属性
LSJFeature f = new LSJFeature();
f.setGeometry(geoPolygon);
f.setName("多边形 01");
mGlobeControl.getGlobe().getMemoryLayer().addFeature(f); // 把几何要素添加到内存图层中
mGlobeControl.getGlobe().refresh();
mGlobeControl.getGlobe().flyToFeature(f);

```

## 5.6 面样式设置

具体工程请查阅示例工程的【LSV-Polygon】

功能描述：设置面的样式

具体代码如下：

```

LSJGeoPolygon3D geoPolygon = new LSJGeoPolygon3D(); // 创建多边形对象
LSJPolygonStyle3D stylePolygon = new LSJPolygonStyle3D(); // 创建风格
stylePolygon.setFill(true); // 是否填充
stylePolygon.setOutlineVisible(true); // 显示多边形的边缘线
geoPolygon.setStyle(stylePolygon); // 把风格添加到多边形上

```

## 5.7 Feature 的删除

具体工程请查阅示例工程的【LSV-Marker】，【LSV-Line】，【LSV-Polygon】

功能描述：点线面数据都要放在 **feature** 几何要素中添加到图层才能展示，删除点线面的时候也要删除对应的集合对象才能删除

方法：removeAllFeatures()；删除所有的集合对象

removeFeatureByCustomID(nID)；根据自定义的 id 删除对象

removeFeatureByID(nID)；根据引擎自带的 id 删除对象

removeFeatureByIndex(nIndex)；根据编号删除对象

removeFeatureByName(strName)；根据名称删除对象

参数在添加点线面对象的时候设置

```

//删除点
mGlobeControl.getGlobe().getMemoryLayer().removeFeatureByName("地标 1");
//删除线
mGlobeControl.getGlobe().getMemoryLayer().removeFeatureByName("线 01");
//删除面

```



```
mGlobeControl.getGlobe().getMemoryLayer().removeFeatureByName("多边形 01");  
mGlobeControl.getGlobe().refresh();
```

## 5.8 Feature 的显隐性控制

具体工程请查阅示例工程的 **【LSV-Marker】**，**【LSV-Line】**，**【LSV-Polygon】**

功能描述：点线面数据都要放在 **feature** 几何要素中添加到图层才能展示，删除点线面的时候也要删除对应的集合对象才能设置是否可见

方法：mGlobeControl.getGlobe().getMemoryLayer().getFeatureByName(strName, bEqual).GetAt(0).setVisible(false);根据对象名称设置显示或隐藏

getFeatureByCustomID(nID).setVisible(false);根据对象自定义 id 设置显示或隐藏

getFeatureByDescription(strDescription, bEqual).GetAt(0).setVisible(false);根据对象描述设置显示或隐藏

getFeatureByID(nID).setVisible(false);根据引擎自带 id 设置显示或隐藏

getFeatureByIndex(nIndex).setVisible(false);根据对象编号设置显示或隐藏

参数在添加点线面对象的时候设置

```
//点线面 Demo 中的隐藏  
mGlobeControl.getGlobe().getMemoryLayer().getAllFeatures().GetAt(0)  
    .setVisible(false);  
//点线面 Demo 中的显示  
mGlobeControl.getGlobe().getMemoryLayer().getAllFeatures().GetAt(0)  
    .setVisible(true);
```

## 六、模型数据

具体工程请查阅示例工程的 **【LSV-Model】**



## 6.1 添加模型数据

功能描述：添加模型数据

添加模型数据前，应先将模型放在手机存储目录下，并记下存储路径

具体代码如下：

```
// 创建模型
LSJGeoModel model = new LSJGeoModel(); // 创建模型
LSJPoint3d pt = new LSJPoint3d(); // 创建点
pt.setX(116.403875); // 设置点的 x 值，单位为度
pt.setY(39.915168); // 设置点的 y 值，单位为度
pt.setZ(3000000); // 设置点的高度，单位为米
// 模型可以是 3ds、gcm 格式的三维模型
// 模型所在路径，用户可根据实际情况进行设置
String filepath = Environment.getExternalStorageDirectory().getAbsolutePath()
    + "/LocaSpace/bone_blade/bone_blade.3ds";
// 设置模型
LSJEntityStyle3D style = new LSJEntityStyle3D();
style.setPolygonMode(EnumPolygonMode.Wireframe);
style.setEntityColor("#FFF68F");
style.setUsingBothFace(false);
style.setUsingLight(true);
```

```
style.setUsingTexture(false);
model.setFilePath(filepath);
model.setStyle(style);
model.setPosition(pt);
model.setAltitudeMode(EnumAltitudeMode.ClampToGround); // 把几何体放到地面上
LSJFeature f = new LSJFeature(); // 创建几何要素
f.setGeometry(model);
f.setName("坦克");
f.setDescription("95式");
f.setVisible(true);
// 设置 feature description 的值,
mGlobeControl.getGlobe().getMemoryLayer().addFeature(f); // 把几何要素添加到内存图层中
// 双面渲染
mGlobeControl.getGlobe().setBothFaceRendered(true);
// 不透明度
mGlobeControl.getGlobe().setGroundOpaque(100);
mGlobeControl.getGlobe().refresh(); // 刷新场景
mGlobeControl.getGlobe().flyToFeature(f);
```

## 6.2 调整模型位置

功能描述：调整模型的位置

方法：`model.move(dx,dy,dz)`

具体代码如下：

```
// 移动模型
LSJFeature ff = mGlobeControl.getGlobe().getMemoryLayer()
    .getFeatureByName("坦克", false).GetAt(0);
ff.setGeometry().move(100, 100, 100);
mGlobeControl.getGlobe().refresh();
```

## 6.3 删除模型数据

功能描述：删除模型数据

具体代码如下：

```
// 删除模型
mGlobeControl.getGlobe().getMemoryLayer().removeFeatureByName("坦克");
mGlobeControl.getGlobe().refresh();
```

## 6.4 属性设置

功能描述：对模型的参数进行设置

具体代码如下：

```
LSJGeoModel model = new LSJGeoModel(); // 创建模型
// 设置模型
LSJEntityStyle3D style = new LSJEntityStyle3D();
style.setPolygonMode(EnumPolygonMode.Wireframe); // 渲染模式
style.setEntityColor("#FFF68F"); // 模型颜色
style.setUsingBothFace(false); // 是否双面渲染
style.setUsingLight(true); // 是否高亮
style.setUsingTexture(false); // 是否显示纹理
model.setFilePath(filepath);
model.setStyle(style);
```

## 七、倾斜摄影

### 7.1 倾斜摄影简介：

具体工程请查阅示例工程的【LSVDemo-QX】

工程效果：



```
package com.example.helloworldqx;
import android.app.Activity;
```

```
import android.os.Bundle;
import android.os.Environment;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.Toast;
import com.LocaSpace.Globe.EnumAltitudeMode;
import com.LocaSpace.Globe.LSJGlobeControl;
import com.LocaSpace.Globe.LSJGlobeControl.ISurfaceCreatedEvent;
import com.LocaSpace.Globe.LSJLayer;
import com.LocaSpace.Globe.LSJLineStyle3D;
import com.LocaSpace.Globe.LSJStyle;
public class MainActivity extends Activity implements OnClickListener {
    // 示例代码
    // 声明类型为 LSJGlobeControl 三位球控件的对象 mGlobeControl
    private LSJGlobeControl mGlobeControl;
    // 声明按钮对象
    private Button add;
    // 判断按钮点击次数
    private boolean isFirst = true;
    // 添加的倾斜摄影文件路径
    private String Path = Environment.getExternalStorageDirectory()
        .getAbsolutePath() + "/dysmodel/Data/DaYangShan.lfp";
    // 示例代码
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 示例代码
        // 实例化三维球控件
        mGlobeControl = new LSJGlobeControl(this);
        // 用控件对象创建视图
        setContentView(R.layout.activity_main);
        try {
            // 将 assets 资源文件中的资源写入到手机中
            this.getAssets();
            LocaSpaceData data = new LocaSpaceData(MainActivity.this);
            data.onCreate();
        } catch (Exception e) {
            Toast.makeText(MainActivity.this, "写入数据失败!", Toast.LENGTH_SHORT)
                .show();
        }
    }
}
```

```

// 用控件对象的 getDeviceKey 方法得到设备机器码
String strDeviceKey = mGlobeControl.getDeviceKey(this);
// 判断许可是否有效
if (!mGlobeControl.isLicValid(this)) {
    // 无效的原因可能因为试用版许可过期了，如果出现过期，请及时联系工作人员
    if (mGlobeControl.isLicExpired(this)) {
    }
    // 注册许可，在添加三维球控件之后
    mGlobeControl.setSerialKey(this,
"302429815#961K16775401601PS943JLJLJL53I8600C4000006008JL5189242UX6201549U591010018Q0JL01699
,0x1206b677,0x1346959f,0x131d3607,0x1346bb99,0xffff,0x0,0x1,0x1,0x1,0x1,0x1,0x1");
    }

    if (!mGlobeControl.isLicValid(this)) {
    }
    initView();
    BackEvent();
    // 设置按钮点击事件
    add = (Button) findViewById(R.id.add);
    add.setOnClickListener(this);
    // //////////////////////////////////////
}
// //////////////////////////////////////
/**
 * 后台事件，视图的创建和显示
 */
private void BackEvent() {
    // 创建界面
    mGlobeControl.setOnSurfaceCreated(new ISurfaceCreatedEvent() {
        @Override
        public void onSurfaceCreated() {
            // 判断视图是否加载
            if (mGlobeControl != null && mGlobeControl.getGlobe() != null) {
                // 跳转到对应位置，绝对地点，既初始视图
                mGlobeControl.getGlobe().JumpTo(0, 0, 0, 0, 0, 19089420.91,
                    EnumAltitudeMode.Absolute);
            }
            // 添加本地图层、添加本地地形，lrc 文件已经打包到工程文件，更多的 lrc 文件请联系工作人员
            mGlobeControl.getGlobe().getLayers().AddLayer(
                Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/LocaSpace/googleLayer203.208.46.128.lrc");
            mGlobeControl.getGlobe().getTerrains().AddTerrain(
                Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/LocaSpace/googleTerrain203.208.46.128.lrc");
            // 打开状态栏

```

```
        mGlobeControl.getGlobe().setStatusBarVisible(false);
    }
}
});
}
private void initView() {
    // 自定义一个 View, 把地球控制视图传递进去
    RelativeLayout globe = (RelativeLayout) findViewById(R.id.globe);
    globe.addView(mGlobeControl);
}
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.add:
            if (isFirst == true) {
                // 得到已有图层数量
                int layerCount = mGlobeControl.getGlobe().getLayers()
                    .GetCount();
                //遍历已有图层, 查找是否有重复项
                for (int i = 0; i < layerCount; i++) {
                    if (mGlobeControl.getGlobe().getLayers().GetLayerByIndex(i)
                        .getName().equals(Path)) {
                        Toast.makeText(this, "图层已添加", Toast.LENGTH_SHORT)
                            .show();
                    } else {
                        // 将倾斜摄影添加到图层
                        mGlobeControl.getGlobe().getLayers().AddLayer(Path);
                        add.setText("飞行");
                        isFirst = false;
                        // 刷新三维地球
                        mGlobeControl.getGlobe().refresh();
                    }
                }
            } else {
                // 得到目标图层
                LSJLayer layer = mGlobeControl.getGlobe().getLayers()
                    .GetLayerByName(Path);
                // 飞行到指定图层
                mGlobeControl.layerFlyToPosition(layer);
                add.setText("添加倾斜摄影");
                isFirst = true;
            }
            break;
        default:
```

```
        break;
    }
}
}
```

## 7.2 根据倾斜摄影数据生成图层文件

```
// 将倾斜摄影添加到图层，文件路径可自己修改
    mGlobeControl.getGlobe().getLayers().AddLayer(
        Environment.getExternalStorageDirectory().getAbsolutePath()
            + "/dysmodel/Data/DaYangShan.lfp");
// 得到用于飞行的目标图层，既刚添加的倾斜摄影
    LSJLayer layer = mGlobeControl.getGlobe().getLayers().GetLayerByName(
        Environment.getExternalStorageDirectory().getAbsolutePath()
            + "/dysmodel/Data/DaYangShan.lfp");
```

## 7.3 倾斜摄影图层文件参数说明

可以通过 `layer.setStyle(style)`方法设置图层样式  
`LSJLineStyle3D style=new LSJLineStyle3D();`  
具体参数设置详见 API 文档

## 八、Server 连接

功能介绍：连接特定服务器，查看服务器上的在线图层  
具体工程请查阅示例工程的【LSV-Server】





## 8.1 连接指定地址的服务器：

方法：`mGlobeControl.getGlobe().connectServer(strIP, nPort, strUser, strPsw)`;

参数说明：`strIP`：IP 地址，`nPort`：端口号，`strUser`：用户名，`strPsw`：密码

具体代码如下：

```
// 连接服务器
try {
    boolean bool = mGlobeControl.getGlobe().connectServer(
        "\"" + "data1.locaspace.cn" + "\", 1500, \"admin\", \"admin\");
    if (bool == true) {
        Toast.makeText(MainActivity.this, \"连接服务器成功!\",
            Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(MainActivity.this, \"连接服务器失败!\",
            Toast.LENGTH_SHORT).show();
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

## 8.2 获取连接服务器成功后的图层信息

功能描述：查看图层信息并飞行

具体代码如下：

```
// 查看数据
LSJLayers layers = mGlobeControl.getGlobe().getLayers();
int count=layers.GetCount();
Random r = new Random();
LSJLayer layer = mGlobeControl.getGlobe().getLayers().GetLayerByIndex(r.nextInt(count));
mGlobeControl.layerFlyToPosition(layer);
```

## 九、量测

功能描述：对三维球体及其加载的图层模型等进行距离量测

地面距离：地球表面两点间的球面距离

空间距离：立体几何中三维空间中点、线、面之间的距离。

具体工程请查阅示例工程的【LSV-Measure】

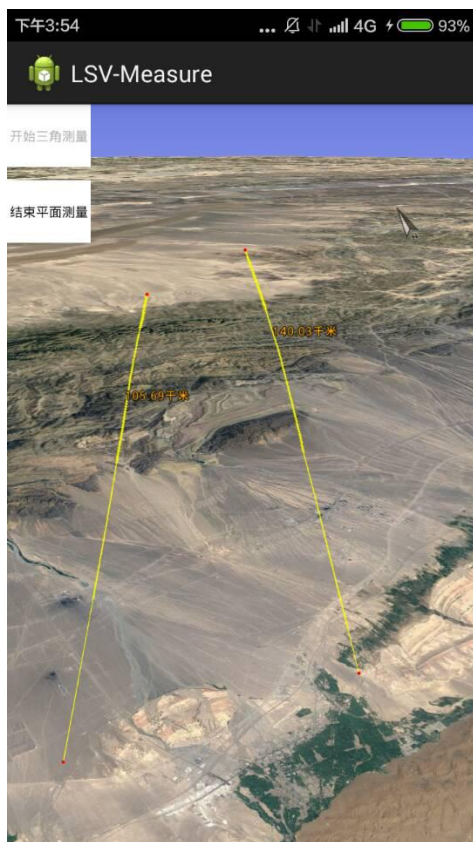
通过改变 mGlobeControl.getGlobe().setAction()的参数值，使 LSJGlobeControl 视图的操作动作进入测量状态。

mGlobeControl.getGlobe().setAction(EnumAction3D.MeasureDistance); //进入距离测量状态

mGlobeControl.getGlobe().setAction (EnumAction3D.MeasureHeight); //进入高度测量状态

mGlobeControl.getGlobe().setAction (EnumAction3D.MeasureArea); //进入面积测量状态

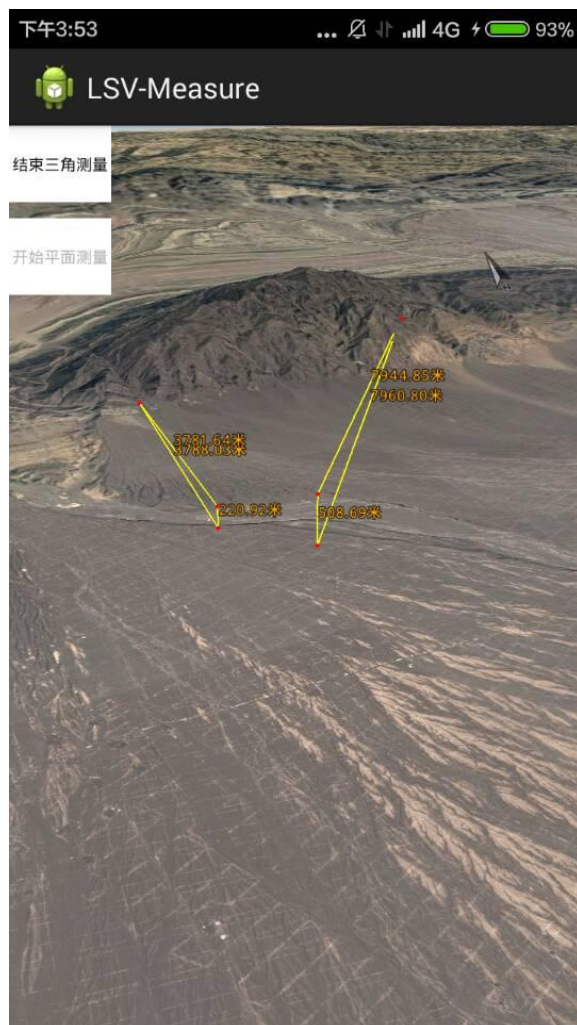
## 9.1 平面距离量测：



具体代码如下：

```
//进入距离量测状态
mGlobeControl.getGlobe().setAction(EnumAction3D.MeasureDistance);
//平面距离量测模式
mGlobeControl.getGlobe().getDistanceRuler().setMeasureMode(EnumDistanceMeasureMode.GroundLineMeasure);
```

## 9.2 三角测量



功能描述：打开三角测量

具体代码如下：

```
//进入距离量测状态  
mGlobeControl.getGlobe().setAction(EnumAction3D.MeasureDistance);  
//三角量测模式  
mGlobeControl.getGlobe().getDistanceRuler().setMeasureMode(EnumDistanceMeasureMode.HVSLineMeasure);
```

## 9.3 关闭量测

具体代码如下：

```
mGlobeControl.getGlobe().setAction(EnumAction3D.ActionNull); //关闭量测状态  
mGlobeControl.getGlobe().clearMeasure(); //清理量测数据  
mGlobeControl.getGlobe().refresh(); //球体刷新
```

## 十、事件

具体工程请查阅示例工程的【LSV-Listener】

### 10.1 要素点击事件

功能描述：点击了三维球体或者球上的要素触发的事件

具体代码如下：

```
// 长按监听
mGlobeControl.setOnSceneLongPressUp(new ISceneLongPressUpEvent() {
    @Override
    public void onSceneLongPressUp(LSJPoint3d arg0, float arg1, float arg2) {
        Toast.makeText(MainActivity.this, "触发了要长按事件监听",
            Toast.LENGTH_SHORT).show();
    }
});

// 要素点击
mGlobeControl.setOnFeatureClick(new IFeatureClickEvent() {
    @Override
    public void onFeatureClick(LSJFeature arg0, LSJPoint3d arg1, float arg2, float arg3) {
        Toast.makeText(MainActivity.this, arg0.getDescription(),
            Toast.LENGTH_SHORT).show();
    }
});
```

### 10.2 回调事件

功能描述：触发了某个条件后的回调事件处理，比如创建了三维球体，开始和结束绘制

具体代码如下：

```
// 创建界面
mGlobeControl.setOnSurfaceCreated(new ISurfaceCreatedEvent() {
    @Override
    public void onSurfaceCreated() {
        Toast.makeText(MainActivity.this, "创建了界面", Toast.LENGTH_SHORT).show();
    }
});

//开始绘制
mGlobeControl.setOnDrawBeginEvent(new IDrawBeginEvent() {
```

```
@Override
public void onDrawBegin(LSJFeature arg0, LSJLayer arg1) {
    Toast.makeText(MainActivity.this, "触发了开始绘制事件监听",
        Toast.LENGTH_SHORT).show();
    }
});
//结束绘制
mGlobeControl.setOnDrawEndEvent(new IDrawEndEvent() {
    @Override
    public void onDrawEnd(LSJFeature arg0, LSJLayer arg1) {
        Toast.makeText(MainActivity.this, "触发了绘制结束事件监听",
            Toast.LENGTH_SHORT).show();
        }
});
```