MapGIS Mobile for Android SDK 开发指南

1 简介

Android SDK 支持使用 Java 语言开发应用程序，MapGIS Mobile 为 Android 平台提供了专门的开发包 MapGIS Mobile for Android，便于开发者打造炫酷的地图应用，开发好的应用能够部署在 Android2.3 及以上智能手机、平板电脑和其他智能终端上，支持 armeabi 和 armeabi-v7a 两种 cpu 架构。

MapGIS Mobile for Android 提供的功能如下：

- 地图功能：提供地图显示，地图界面控制，地图操作功能和地图文档管理功能；
- 地图文档管理功能: 支持开发者查看地图的相关信息，管理和自定义图层；
- 自定义图形：支持在地图上画点、线、圆、多边形，添加图像，添加文本；
- 地图标注：支持在地图上添加标注，添加标注视图等；
- 地图工具：通过放大镜工具，可在地图上实现交互式绘制；
- 属性查询：支持查询图层中的字段信息；
- 空间查询：支持选定空间范围对图层信息进行查询
- 空间量算与分析：支持面积距离的量算、缓冲区分析、叠加分析
- POI 查询：根据用户输入的关键字进行模糊查询，寻找到用户感兴趣的地点
- 路径规划：通过起点和终点以及偏好，规划出想要的路径

MapGIS Mobile for Android 向广大开发者提供了 jar 和 so 形式的开发包，基于 java 语言的开发，该形式开发包配置简单、使用方便，欢迎广大开发者下载使用。

2 配置开发环境

配置开发环境前请到云开发世界下载 MapGIS_Mobile_Android_SDK，下载的 MapGIS_Mobile_Android_SDK .zip 解压后里面有示例 demo，开发文档及开发所

需要的 Jar 包和 So 库，下载位置如下图所示：

Android 开发工具很多，在这我们推荐各位开发者使用 Eclipse 和 Android Studio 作为自己的开发工具。

2.1 Eclipse 配置工程

- 第一步：新建一个 Android 工程

新建一个 Empty Activity 应用工程。

- 第二步：添加 jar 和 so

在上一步创建的工程中新建 libs 文件夹，将云开发世界下载的 MapGIS_Mobile_Android_SDK .zip 解压后将 SDK 的 jar 文件和 so 库一起拷贝到 libs 的根目录下（此处截图以官方示例 Demo 为例子）。

如下图所示；

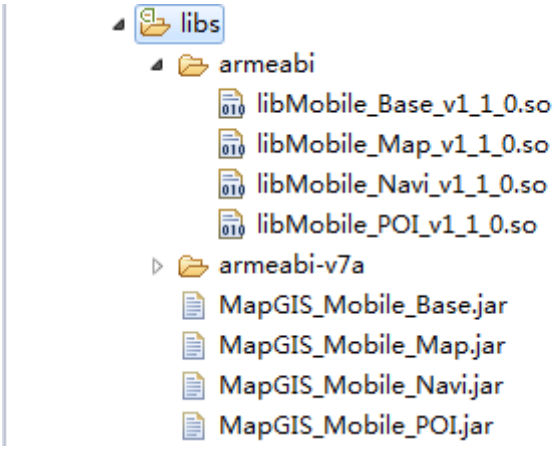


图 2-1 sdk 工程目录

- 第三步：在工程属性->Java Build Path->Libraries 中选择“Add External JARs”，选定 MapGIS_ Mobile_Base.jar、MapGIS_Mobile_Map.jar、MapGIS_Mobile_Navi.jar、MapGIS_Mobile_POI.jar，确定后返回。

通过以上两步操作后，获取云开发授权您就可以正常使用 MapGIS_Mobile_Android SDK 为您提供的全部功能了。

注意：由于 adt 插件升级，若您使用 Eclipse adt 22 的话，需要对开发环境进行相应的设置，方法如下：

1) 在 Eclipse 中选中工程，右键选 Properties->Java Build Path->Order and Export 使 Android Private Libraries 处于勾选状态；
2) Project -> clean-> clean all

2.2 Android Studio 配置工程

- 第一步：新建一个 Android 工程

新建一个 Empty Activity 应用项目

- 第二步：添加 jar 包

将下载的地图 SDK 的 jar 包复制到工程（此处截图以官方示例 Demo 为例子）的 libs 目录下。

- 第三步：添加 so 库

1) 使用默认配置，不需要修改 build.gradle。在 main 目录下创建文件夹 jniLibs (如果有就不需要创建了)，将下载文件的 armeabi 文件夹复制到

这个目录下,如果已经有这个目录，将下载的 so 库复制到这个目录即可。如图所示：

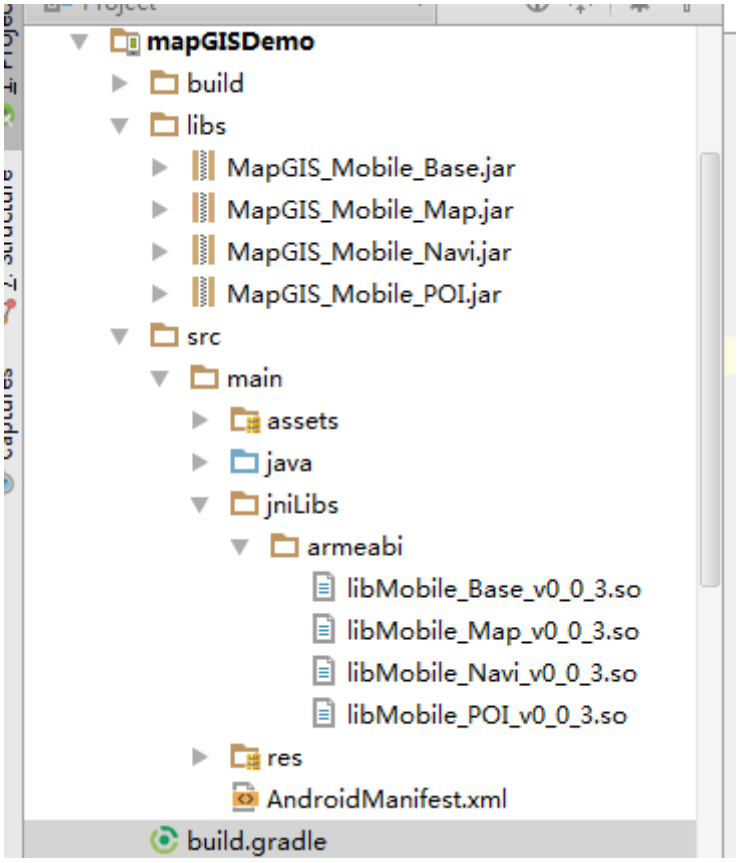
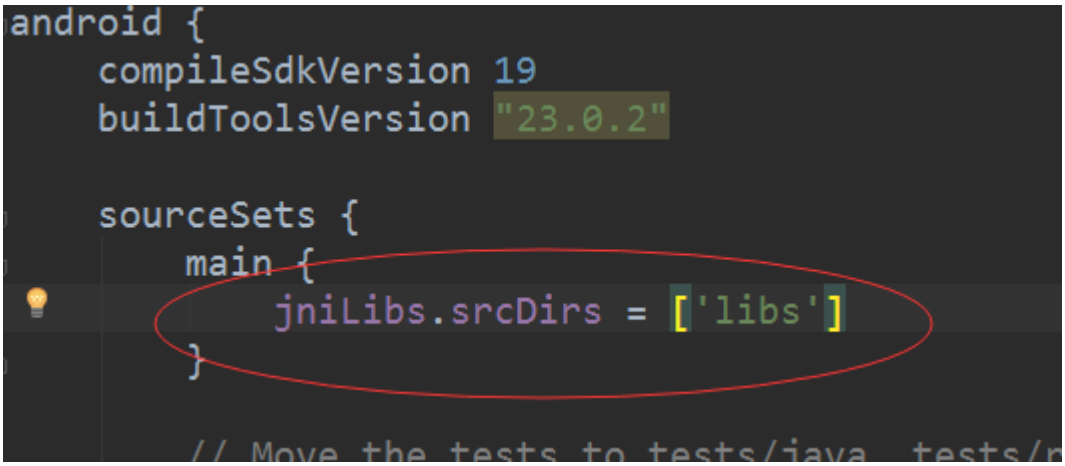


图 2-1 studio 工程目录

2) 使用自定义配置，将下载文件的 armeabi 文件夹复制到 libs 目录，如果有这个目录,请将下载的 so 库复制到这个目录，然后打开 build.gradle，找到 asourceSets 标签，在里面增加一项配置，如图所示：



3 获取云授权

- 登录获取开发授权

第一步：登录（http://www.smaryun.com）注册开发者账号

第二步：Activity 请求授权验证代码如下：

```
com.zondy.mapgis.android.environment.Environment.requestAuthorization(this,
new AuthorizeCallback()
{
@Override
public void onComplete()
{initMap();}
});
```

第三步：移动端登录注册的开发者账号获取开发授权

➢ 设备导入授权证书

第一步：登录（http://www.smaryun.com）注册开发者账号

第二步：在云开发世界->工作台->我的开发环境中下载授权

第三步：导入授权

4 基本地图功能展示

对地图文档进行加载并显示，默认情况下可以进行地图的放大、缩小、平移、倾斜、旋转等操作。

```
//加载地图
public static final String rootPath =
Environment.getExternalStorageDirectory().getAbsolutePath();
mapView = (MapView)findViewById(R.id.mapview1);
mapView.loadFromFile(MainActivity.rootPath+"/mapgis/map/wuhan/wuhan.xml");
```

5 地图界面控制

对地图显示进行基本的界面设置，是否显示缩放按钮、指南针、我的位置按钮、显示图标。

```
//显示缩放按钮
mapView.setZoomControlsEnabled(true);
//显示指南针按钮
mapView.setShowNorthArrow(true);
//显示定位按钮
mapView.setMyLocationButtonEnabled(true);
//显示图标
mapView.setShowLogo(true);
```

6 地图操作

地图操作指 MapView 提供一些一些函数包括：异步加载地图、地图基本操作、地图手势控制、地图动画控制、地图显示监听、地图手势监听等。

6.1 异步加载地图

异步加载地图，主要用于地图图层数目较多或加载在线服务图层的时候，加载比较慢，这个时候不能把地图加载放到主线程，需要在子线程加载地图，SDK

提供异步加载地图的方法 loadFromFileAsync()

```
// 异步加载地图
mMapView = (MapView) findViewById(R.id.setfolders_mapview);
mMapView.loadFromFileAsync(FilePathCfg.MAPX_FILE_PATH);
```

6.2 系统路径设置

地图都可以切换颜色库和符号库，方便个性化定制地图

```
mMapView.stopCurRequest(new MapViewStopCurRequestCallback()
{@Override
public void onDidStopCurRequest(){
Environment.setSystemLibraryPath(ANOTHER_SYSTEMPATH);
mMapView.forceRefresh();}});
```

6.3 地图基本操作

地图基本操作包括，地图的放大、缩小、复位、旋转、倾斜、跳转（中心点+级别范围）、显示模式（2D,3D）、截屏、指定出图，每个功能所用到的类与方法

如下所示：

```
    //地图放大
    mapView.zoomIn(true);
    //地图缩小
    mapView.zoomOut(true);
```

```
    //地图复位
    mapView.setRotateAngleAndSlopeAngle(0.0f, 0.0f, false);
    mapView.setRotateCenter(new Dot(0.0f,0.0f));
    Rect entireRange = mapView.getMap().getEntireRange();
    mapView.zoomToRange(entireRange, false);
```

```
    //地图旋转
    rotateAngle += 10.0f;
    dispRange = mapView.getDispRange();
    Dot dotcenter = new Dot((dispRange.xMax + dispRange.xMin) / 2,
                            (dispRange.yMin + dispRange.yMax) / 2);
    mapView.setRotateCenter(dotcenter);
    mapView.setRotateAngle(rotateAngle, false);
```

```
    //地图倾斜
    mapView.setSlopeAngle(slopeAngle, false);
    //地图跳转
    mapView.zoomToCenter(mapView.getCenterPoint(), 2.0f, true);
    //地图显示模式
    mapView.setShowMode(mapView.ViewShowMode3D);
```

```
    //地图截屏
    Bitmap bitmapCapture = Bitmap.createBitmap(mapView.getMeasuredWidth(),
                                mapView.getMeasuredHeight(),Bitmap.Config.ARGB_
8888);
    mapView.capture(bitmapCapture);
    //指定出图
    Bitmap toBitmap = Bitmap.createBitmap(mapView.getMeasuredWidth(),
                        mapView.getMeasuredHeight(), Bitmap.Config.ARGB_8888);
    dispRange = mapView.getDispRange();
    mapView.setSupportTransparency(true);
```

```
mapView.getBitmap(dispRange, toBitmap);
//指定出图接口 2
Bitmap toBitmap2 = Bitmap.createBitmap( mapView.getMeasuredWidth() / 2,
                    mapView.getMeasuredHeight() / 2,
Bitmap.Config.ARGB_8888);
mapView.capture(mapView.getMeasuredWidth() / 4,
                    mapView.getMeasuredHeight() / 4, toBitmap2);
```

6.4 地图手势控制

- 禁用和使能手势操作：

单指双击是否放大地图，双指单击是否缩小地图，双指竖直下滑是否使地图倾斜，自由手势缩放地图，控制地图是否可以滑动，双指旋转是否可以旋转地图，传感器是否使地图器倾斜。每个功能所用到的类与方法如下所示：

```
//双指单击缩小地图
mapView.setTwoFingerTapZooming(true);
//双指放大地图
mapView.setDoubleTapZooming(true);
//自由缩放地图
mapView.setMapZoomGesturesEnabled(true);
//手势滑动地图
mapView.setMapPanGesturesEnabled(true);
//双指竖直下滑倾斜
mapView.setMapSlopeGesturesEnabled(true);
//双指旋转地图
mapView.setMapRotateGesturesEnabled(true);
//传感器倾斜
mapView.setSensorSlopeEnabled(true);
```

6.5 地图动画控制

- 禁用和使能动画效果：

是否禁用平移结束动画（当抬起手指的时候地图立马停止动画），缩放超出级别动画（手指放大过程中允许超出缩放级别，但手势结束后，动画缩放回来）每个功能用到的类与方法如下所示：

```
//禁用超出缩放级别动画
mapView.setZoomOutOfLevelEnabled(false);
```

```
//禁用平移结束动画
mapView.setPanEndAnimating(false);
```

6.6 地图显示监听

当对地图做操作的时候，其实地图内部不停刷新、变化，地图显示监听在界面显示：级别变化、角度变化、中心点变化、动画监听、地图是否在刷新。

注意：监听的触发是在子线程中，可以通过 Handler 建立子线程与主线程的通信。

6.7 地图手势监听

当对地图短按，双击、触摸等，地图会捕获相应的触摸点，地图手势监听在界面显示用户：短按监听，长按监听，双击监听，触摸监听动态显示当前坐标。

地图长按实现 MapViewLongTapListener 接口并实现构造方法获取长按坐标

地图双击实现 MapViewDoubleTapListener 接口并实现构造方法获取双击坐标

地图短按实现 MapViewTapListener 接口并实现构造方法获取短按坐标

地图触摸监听实现 MapViewTouchListener 接口并实现构造方法获取触摸坐标

7 地图文档管理

7.1 地图基本信息

该页面显示地图基本信息：地图名称、范围、参照系、符号比等

每张地图都有名字，范围、参照系、符号比等通过 Map 对象获得地图名称，地图范围，符号比等信息

```
//地图名字
String mapName = mapView.getMap().getName();
//地图显示范围
Rect range =mapView.getMap().getRange();
//获取投影坐标系的名称
String strSRef = sRef.getPCSName();
//符号比
double symbol = mapView.getMap().getSymbolScale();
```

7.2 图层管理

7.2.1 图层基本信息

地图是由多个图层组成，每个图层会显示一部分的地理或交通信息。开发者可以通过设 MapLayer，灵活控制图层的显示状态。例如，用户所看到包括街道、兴趣点、学校、公园等内容的地图展现就是一个图层。实时路况等的展现也是通过图层来实现的。

点击图层查看按钮，可以查看地图所有图层，在图层列表中点击详情可以查看图层详细信息，比如：图层名称、图层范围、图层 URL，显示比等

长按图层列表，可以删除地图图层，核心代码如下所示：

```
map.remove(mapLayer);
mMapLayers.remove(finalPosition);
adapterLayers.notifyDataSetChanged();
mapView.refresh();
```

7.2.2 图层控制

地图是由很多个图层叠加到一起组成，该页中图层控制功能展示地图所有的图层名称、状态进行显示，点击图层可以控制图层可见或不可见。

```
//获取图层数目。
int layerCount = map.getLayerCount();
//根据图层数目，对图层进行遍历，遍历到 ListView 中，取出相应图层的名称和状态。
final CommAdapter adapterLayers = new CommAdapter();
adapterLayers.setContext(getApplicationContext());
adapterLayers.setListField(getData());
//并控制图层可见不可见
map.getLayer(i).setVisible(true);
```

7.2.3 图层添加

该模块图层的添加，一种添加在线 Google 在线服务图层，一种是添加离线瓦片服务图层，添加这两种类型服务图层都用到 ServerLayer 这个类，在武汉地

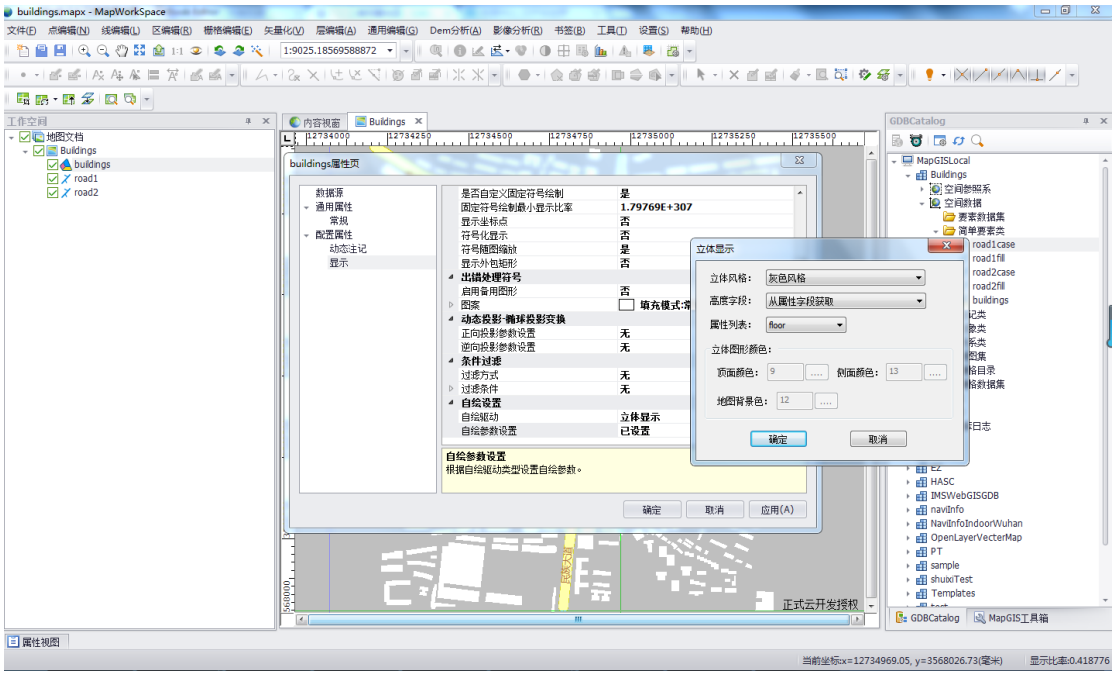
图上添加在线 Google 服务图层的时候，需要注意，由于添加的 Google 在线服务图层地理坐标显示范围是全世界的，当前地图显示范围是武汉的地理坐标，为了能看到当前武汉的显示范围，添加完在线服务图层的时候利用 mapView.zoomToRange()这个方法就可以显示当前武汉地图的显示范围，核心代码如下所示：

```
serverLayer.setMapServerByType(MapServerType.GoogleMap);
serverLayer.setName("Google 服务图层");
map = mapView.getMap();
Rect rect = map.getRange();
map.insert(0, serverLayer);
mapView.setMap(map);
mapView.zoomToRange(rect, false);
mapView.refresh();
```

7.3 三维场景地图

移动端采用 3DBuildingLayer 图层进行三维楼块显示，目前提供数据制作方式，采用基于二维区图层的方式组织三维模型数据具体步骤如下所示：

(1) 准备 3D 楼块源数据。在 MapGIS 10 桌面工具（MapWorkSpace）中制作二维区图层，building 区图层右键属性-->显示-->自绘参数设置如下图所示：



(2) 组织地图文档。将二维区图层和其他需要的二维图层一起组织成地图文档

buildings.mapx；

(3) 数据转换。将上一步生成的地图文档 buildings.mapx 使用数据转换工具进行转换；方法为运行“MapMigrateTool.exe”选择需要转换的数据（即上一步中桌面端组织的地图文档 buildings.mapx），指定转换后数据的存放路径，点击“确定”即可，转换完成后会在指定存放路径下生成一个文件夹，文件夹包含两个文件：一个 buildings.mapx，一个 buildings.db，把转换后的数据放到移动端加载出来的效果如下图所示：

7.4 自定义地图服务

地图服务负责给地图提供数据，自定义地图服务用户可以自定义切片规则，指定地图原点，地图范围，自定义缩放级别，并可以规定只让用户看到部分缩放级别。

Mapserver 是各图层提供数据，ServerLayer 调用 MapServer 获取数据，自定义一个地图服务继承 MapServer 类，该模块在武汉地图上添加一张瓦片，具体实现步骤如下：

第一步：实现自定义地图服务类 DefineMapServer 继承 MapServer

1) 重写 getEntireExtent（）给定整个地图数据范围 dataRect，图片的高宽和地

图数据范围高宽成比例，计算 dataRect 代码如下所示，其中 entireRect 为传递过来的地图范围：

```
double entireDataWidth = (entireRect.xMax - entireRect.xMin) / 2;
double entireDataHeight = (entireRect.yMax - entireRect.yMin) / 2;
bitmapWidth = bitmapData.getWidth();
bitmapHeight = bitmapData.getHeight();
entireDataHeight = entireDataWidth * bitmapHeight / bitmapWidth;
//求出地理范围
dataRect.xMin = centerX - entireDataWidth / 4;
dataRect.xMax = centerX + entireDataWidth / 4;
dataRect.yMin = centerY - entireDataHeight / 4;
dataRect.yMax = centerY + entireDataHeight / 4;
```

2) 重写 getMinZoom（）最小缩放级别

3) 重写 getMaxZoom（）最大缩放级别

4) 重写 getZoomCapacity()方法，设置瓦片自身缩放能力

5) 重写 getTileResolution（），这个方法返回一个不同级别下不同分辨率的数组，计算分辨率代码如下所示：

```
//设置级别 2 显示原始图片大小，
resolutions[refZoom] = (dataRect.xMax - dataRect.xMin) / bitmapWidth;
for(int i = minZoomCapacity;i < refZoom;i++)
{
resolutions[i] = resolutions[refZoom] * Math.pow(2, refZoom - i);
}
for(int j = refZoom + 1;j <= maxZoomCapacity;j++)
{
resolutions[j] = resolutions[refZoom] / Math.pow(2, j - refZoom);}
```

6) 重写 getTileOriginXY（）方法，给定瓦片原点 dotOrigin，在这里地图原点设为左上角：

```
dotOrigin.x = dataRect.xMin;
dotOrigin.y = dataRect.yMax;
```

7) 重写 getTileSize（）设置瓦片大小，一般 256 显示效果要好点

8) 重写 getTileMatrix(long zoom, LongUser topRow, LongUser leftCol, LongUser bottomRow,

LongUser rightCol)传递每个级别下其实行列号，计算行列号代码如下所示：

```
// 计算瓦片起始行列号
topRow.setValue((long) Math.floor((dotOrigin.y - dataRect.yMax) /
(resolutions[(int) zoom] * tileSize)));
leftCol.setValue((long) Math.floor((dataRect.xMin - dotOrigin.x) /
(resolutions[(int) zoom] * tileSize)));
//终止行列号向下取整
bottomRow.setValue((long) Math.floor((dotOrigin.y - dataRect.yMin) /
(resolutions[(int) zoom] * tileSize)));
rightCol.setValue((long) Math.floor((dataRect.xMax - dotOrigin.x) /
(resolutions[(int) zoom] * tileSize)));
```

9) 重写 getTileImage(long row, long col, long zoom)根据 getTileMatrix()方法计算出来的行列号，在这里反算出行列号对应的地图范围，图片的地理范围包含在其中，然后根据地理范围跟 dataRect 的比值，计算图片尺寸，并返回当前瓦片二进制，代码如下所示：

```
//tileDataRect 为根据传递行列号计算出当前矩阵行列，对应地理范围
Rect tileDataRect = new Rect();
double tileDataSize = resolutions[(int) zoom] * tileSize;
tileDataRect.xMin = dotOrigin.x + tileDataSize * col;
tileDataRect.xMax = dotOrigin.x + tileDataSize * (col + 1);
//根据瓦片外包矩形与地理范围的比值，计算出当前瓦片矩形对应图片的高宽
int xmin = (int) ((tileDataRect.xMin - dataRect.xMin) / (dataRect.xMax –
dataRect.xMin) * bitmapWidth);
int ymin = (int) ((dataRect.yMax - tileDataRect.yMax) / (dataRect.yMax –
dataRect.yMin) * bitmapHeight);;
int xmax = (int) ((tileDataRect.xMax - dataRect.xMin) / (dataRect.xMax –
dataRect.xMin) * bitmapWidth);
int ymax = (int) ((dataRect.yMax - tileDataRect.yMin) / (dataRect.yMax –
dataRect.yMin) * bitmapHeight);
//在当前分辨率下，根据行列号反算的比值，计算当前行列号对应图片的尺寸，
并返回当前尺寸的二进制
Bitmap bitmapCanvas = Bitmap.createBitmap(tileSize, tileSize,
Bitmap.Config.ARGB_8888);
Paint paint = new Paint();
```

```
Canvas canvas = new Canvas();
canvas.setBitmap(bitmapCanvas);
canvas.drawBitmap(bitmapData, srcRect, dstRect, paint);
```

第二步：实现每个级别瓦片显示，并添加到地图上，代码如下所示：

```
// 地图范围及中心点
rect = map.getRange();
DefinedMapServer defineServer = new DefinedMapServer();
defineServer.setName("自定义地图服务");
defineServer.setData(rect,bitmap);
serverLayer.setMapServer(defineServer);
serverLayer.setName("自定义地图服务图层");
map.append(serverLayer);
mMapView.setMap(map);
mMapView.refresh();
```

第三步：在不同手机上测试自定义地图服务

由于不同手机上显示图片的分辨率不一样，导致计算出的缩放级别不一致，因此把图片放到 assets 文件夹下，读取原始图片的二进制，代码如下所示：

```
//获取 assets 下图片资源
AssetManager am = getAssets();
InputStream is = am.open("test.jpg");
bitmap = BitmapFactory.decodeStream(is);
```

8 自定义图形

自定义图形模块包括在地图上画点，画线，画虚线，画纹理线，添加文本，添加图像，画圆，画多边形功能，所用到的类都在 com.zondy.mapgis.android.graphic 包下面，每个功能都有相通性，画线思路及核心代码如下所示：

```
//将屏幕坐标转化成地图坐标。
mapView.viewPointToMapPoint(viewPoint);
//构造一个线对象。
GraphicPolylin graphicPolylin = new GraphicPolylin();
//线对象属性设置。
graphicPolylin.appendPoint(point);
```

```
        graphicPolylin.setColor(Color.argb(255, 0, 0, 255));
        graphicPolylin.setLineWidth(6);
        // 将绘制的图形添加到自定义图层 GraphicLayer 中，想看到效果记得刷新地图。
        mapView.getGraphicLayer().addGraphic(graphicPolylin);
        mapView.refresh();
```

画纹理线，虚线，多边形，圆，类似以上思路，效果如下图所示：

9 标注及标注视图

9.1 添加标注

■ 实现在地图上添加标注，以标记当前位置，以及清除地图上所有标注的核心代码如下所示：

```
        // 创建 annotation
        Annotation annotation1 = new Annotation("Annotation1","标注 1", position1, bmp1);
        //  获取标注图层，添加标注
        mapView.getAnnotationLayer().addAnnotation(annotation1);
        //清除地图上所有标注的功能
        mapView.getAnnotationLayer().removeAllAnnotations();
```

■ 在地图添加不可弹出标注视图标注调用 Annotation 类中方法

```
//设置标注不可弹出标注视图
annotation1.setCanShowAnnotationView(false)
```

9.2 添加标注视图

实现添加标注视图：标注分为可弹出标注视图，不可弹出标注视图，同时可以更改可弹出标注视图样式，添加标注视图核心代码与思路如下所示：

■ 创建 MapViewAnnotationListener 的监听，并在监听 mapViewViewForAnnotation()方法里面创建 AnnotationView

```
annotationView = new AnnotationView(annotation, this);
//设置 AnnotationView 样式
annotationView.getCalloutTitleTextView().setTextColor(Color.BLUE);
annotationView.getCalloutDescriptionTextView().setTextColor(Color.BLACK);
// 设置 callout 相对于标注或视图点的偏移量
annotationView.setCalloutOffset(new Point(0, 15));
annotationView.getCalloutDescriptionTextView().setSingleLine(false);
// 将 annotationview 平移到视图中心
annotationView.setPanToMapViewCenter(true);
```

10 地图工具

10.1 地图放大镜工具

地图放大镜工具在 MapView 类中有两个方法，一种是默认放大镜位置自动调整模式，当手指在移动的过程中，可以放大手指所在位，另一种方法是用户自定义放大镜位置，交互绘制的时候会用到。

```
//创建 MagnifierOption，pointf（自定义放大镜位置）
MagnifierOption magnifier = new MagnifierOption(width/3, 2.5f,
MagnifierOption.AUTO_ADJUST_POINT, pointf);
//设置放大镜
magnifier.setPointAdjustMode(MagnifierOption.AUTO_ADJUST_POINT);
//调用 MapView 中方法，打开放大镜
mapView.turnOnMagnifier(magnifier);
```

10.2 交互式绘制

1. 交互式绘制，动态的类似橡皮筋拉伸方式绘制线，同时拉伸的过程中调用放大镜，对线末端区域进行放大，实现步骤以及核心代码如下所示：

(1) 首先实现 MapTool 接口，并添加未实现的方法

(2) 当手机接触屏幕时候，获取当前地图坐标，并以这个点做为交互绘制的起点

```
PointF pointf = new PointF(event.getX(), event.getY());
Dot dot = mapView.viewPointToMapPoint(pointf);
graphicPolylin.appendPoint(dot);
```

(3) 在 panStateBegan（）方法中获得当前手指触摸的视图坐标，然后把地图坐标转化为视图坐标，并把当前地图坐标追加到线中，并添加到图层中

```
PointF pointf = new PointF(event.getX(), event.getY());
Dot dot = mapView.viewPointToMapPoint(pointf);
graphicPolylin.appendPoint(dot);
mapView.getGraphicLayer().addGraphic(graphicPolylin);
mapView.refresh();
```

(4) 在 panStateChanged（）方法中，当用户手发生改变，移除最后一个追加的点，把当前动态改变的点追加到线中，

```
int index = graphicPolylin.getPointCount();
graphicPolylin.removePoint(index - 1);
```

```
PointF pointf = new PointF(event.getX(), event.getY());
Dot dot = mapView.viewPointToMapPoint(pointf);
graphicPolylin.appendPoint(dot);
```

(5) 当用户手指离开屏幕，调用 panStateEnded()方法把最后一个点追加到线中

2. 交互绘制同时加入自定义放大镜

(1) 当手指在屏幕触摸点发生改变的时候，在这个点位置加入放大镜

```
PointF magnifierPosition = new PointF(45.0f, 45.0f);
MagnifierOption magnifier = new MagnifierOption(180, 2.5f,
MagnifierOption.USER_CUSTOM_POINT, magnifierPosition);
magnifier.setPointAdjustMode(MagnifierOption.USER_CUSTOM_POINT);
mapView.showMagnifier(pointf, magnifier);
```

(2) 手指触摸点发生改变放大镜内容也以这个点为中心，当手指离开触摸屏，调用 panStateEnded（），然后隐藏放大镜

```
mapView.hideMagnifier();
```

11 属性查询

根据选择图层、字段、关键字对图层属性进行查询并显示查询结果。

1. 首先获取当前显示地图的图层。
2. 其次获取图层属性，根据获取到的属性选择要查询的字段。

1) 查询并存储要素结果

```
//存储所有图层信息
for (int ii = 0; ii < layerCount; ii++) {
CommField field = new CommField(map.getLayer(ii).getName(), " ");
field.setUserMoreImg(R.drawable.item_selected);
listLayers.add(field);}
```

2) 获取总的要素数目

```
FeatureQuery featureQuery = new FeatureQuery();
//设置查询字段
featureQuery.setWhereClause(strWhereClause);
//设置查询图层
featureQuery.setVectorLayer(selAttLayer);
//属性查询
FeaturePagedResult featurePagedResult = featureQuery.query();
//获取总的要素数目
```

```
int totalCount = featurePagedResult.getTotalFeatureCount();
```

3) 获取要素结果的页数

```
featureList=featurePagedResult.getPage((int) pageIndex);
```

4) 获取要素，获取要素的字段结构

```
//获取要素属性
Record record = feature.getAtt();
//获取要素字段结构
Fields fields = feature.getFields();
//获取字段数目
short fieldCount = fields.getFieldCount();
```

5) 获取字段名及字段值

```
Field field = fields.getField(j);
//获取字段名称
String strFieldName = field.getFieldName();
//获取对应字段的值
Object fieldValue = record.getFldVal(strFieldName);
```

3. 最后，输入关键字查询具有相应字段值的要素。

12 空间查询

进行空间属性查询，查询方式有通过点、矩形、多边形查询。

1. 将屏幕坐标转化成地图坐标。

2. 创建矩形查询范围

```
//创建矩形查询范围
Rect rect = new Rect();
rect.setXMin(mapPoint.x - mapView.getResolution(mapView.getZoom()) * 20);
rect.setYMin(mapPoint.y - mapView.getResolution(mapView.getZoom()) * 20);
rect.setXMax(mapPoint.x + mapView.getResolution(mapView.getZoom()) * 20);
rect.setYMax(mapPoint.y + mapView.getResolution(mapView.getZoom()) * 20);
```

3. 获取当前显示的地图

4. 获取待查询图层

```
//可见,并且是矢量图层,则查询
for(int ii = 0; ii < map.getLayerCount(); ii++){
//图层的最大最小显示比
double maxScale = mapView.getMap().getLayer(ii).getMaxScale();
double minScale = mapView.getMap().getLayer(ii).getMinScale();
//判断当前地图是否显示
if (scaleCompare(minScale,maxScale) || ((scaleCompare(minScale,displayScale)
||
displayScale > minScale)&& (scaleCompare(maxScale,displayScale)
|| displayScale < maxScale))){
vlayer = (VectorLayer) map.getLayer(ii);
String str=vlayer.getName();
//查询 Water_polygon 图层
if(str.equals("Water_polygon"))
break; }}
```

5. 获取查询结果，并设置查询条件，获取查询结果

```
//设置要素查询条件
FeatureQuery.QueryBound featureQueryBound = new
FeatureQuery.QueryBound(rect);
//存储要素查询结果
FeaturePagedResult featurePagedResult
=FeatureQuery.query(vlayer,"",featureQueryBound,
```

```
0,true, false, "", 1);
```

6. 转换查询结果相应的图形，并高亮显示查询结果

```
//获取要素
List<Feature> featureList = featurePagedResult.getPage(1);
for(int i = 0;i<featureList.size();i++){
Feature feature = featureList.get(i);
//获取要素对应的图形
List<Graphic> graphics = feature.toGraphics(true);
//高亮显示查询的结果
mapView.getGraphicLayer().addGraphics(graphics);      }
```

13 空间量算与空间分析

13.1 距离，面积量算

该模块分为地图空间距离量算和面积量算，用来计算自定义图形折线的长度，并把计算结构放到 GraphicText 中并展示出来，自定义多边形的面积，计算出来面积同时把结果展示到界面上。

13.1.1 计算线长度

调用 GeoVarLine 类中 calLength()方法计算自定义折线长度，具体核心代码与思路如下所示：

1. 自定义 5 个视图坐标，并把视图坐标转化为地图坐标

```
//根据屏幕的高宽自定义坐标点
DisplayMetrics dm = getResources().getDisplayMetrics();
width = dm.widthPixels;
height = dm.heightPixels;
point1 = new PointF(width * 1/ 10, height * 1 / 10);
point2 = new PointF(width * 4/ 10, height * 1f/ 10);
point3 = new PointF(width * 5.0f/ 10, height * 3 / 10);
point4 = new PointF(width * 3.5f/10, height * 4.5f/ 10);
point5 = new PointF(width * 2.0f/ 10, height * 4.5f/ 10);
```

2. 把这 5 个点转化成地理坐标并设置到 GraphicPolylin 对象中

```
// 将视图坐标转化为地图坐标
Dot dot1 = mapView.viewPointToMapPoint(point1);
Dot dot2 = mapView.viewPointToMapPoint(point2);
Dot dot3 = mapView.viewPointToMapPoint(point3);
Dot dot4 = mapView.viewPointToMapPoint(point4);
Dot dot5 = mapView.viewPointToMapPoint(point5);
```

3. 将自定义图形转化为几何图形，然后计算折线长度

```
GeoVarLine geoVarLine = (GeoVarLine) graphicPolylin.toGeometry(graphicPolylin);
double length = geoVarLine.calLength();
```

4. 把计算的长度设置到 GraphicText 对象，并展示到界面中

```
graphicText.setPoint(dot3);
graphicText.setFontSize(20);
graphicText.setText("折线的长度：" + df.format(length));
```

13.1.2 计算面积

自定义多边形的面积计算调用 GeoPolygon 类中 calArea()方法得到多边形的面积，具体核心实现代码以及思路如下所示：

1. 自定义 5 个视图坐标，并把视图坐标转化为地图坐标
2. 把这 5 个点转化成地理坐标并设置到 GraphicPolygon 对象中
3. 将自定义图形转化为几何图形，然后计算多边形面积

```
GeoPolygon geoPolygon = (GeoPolygon) graphicPolygon.toGeometry(graphicPolygon);
double area = geoPolygon.calArea();
```

4. 把计算的面积设置到 GraphicText 对象中，并展示到界面中

```
// 设置 text
graphicText.setPoint(dot3);
graphicText.setText("多边形的面积： " + df.format(area));
graphicText.setFontSize(20);
mapView.getGraphicLayer().addGraphic(graphicText);
mapView.refresh();
```

13.2 缓冲区分析

缓冲分析就是在点、线、面实体（缓冲目标）周围建立一定宽度范围的多边形

该模块为点缓冲分析，线缓冲分析，面缓冲分析，一般用来计算点、线、面附近 1000 米区域范围，点、线、面缓冲分析，都调用 SpaAnalysis 类中 buffer（）方法进行缓冲分析，面缓冲分析核心代码与思路如下所示：

```
//自定义 5 个视图坐标，并把视图坐标转化为地图坐标
Dot dot1 = mapView.viewPointToMapPoint(point1);
Dot dot2 = mapView.viewPointToMapPoint(point2);
Dot dot3 = mapView.viewPointToMapPoint(point3);
Dot dot4 = mapView.viewPointToMapPoint(point4);
Dot dot5 = mapView.viewPointToMapPoint(point5)

//把地图坐标 S 设置到 GraphicPolygon 对象中
graphicPolygon.setPoints(dots);
//把自定义图形转化为几何图形，并进行缓冲分析
Geometry geometry = Graphic.toGeometry(graphicPolygon);
GeoPolygons geoPolygons = spaAnalysis.buffer(geometry, 1000,1000);
List<Graphic> graphics = Graphic.toGraphicsFromGeometry(geoPolygons);
//把缓冲的图形添加到地图图层中显示出来
for(int i=0;i<graphics.size();i++){
```

```
graphics.get(i).setColor(resultColor);
mapView.getGraphicLayer().addGraphic(graphics.get(i));}
mapView.getGraphicLayer().addGraphic(graphicPolygon);
mapView.refresh();
```

13.3 叠加分析

叠加分析是地理信息系统最常用的提取空间隐含信息的手段之一。地理信息系统的叠加分析是将有关主题层组成的数据层面进行叠加，产生一个新的数据层面的操作，其结果是原来的要素被分割、剪断、套合，然后生成新的要素，新要素综合了原来两层要素所具有的属性，叠加结果是新的矢量数据图层，量叠加分析中至少涉及到三个数据集，其中进行叠加的两个数据集可以是点、线、面类型。叠加分析类型： 点对区叠加：相交、相减、叠加结果是一串带有附加属性的点

要素； 点对线叠加：相交，叠加结果为点简单要素类； 区对点叠加：相交叠加和相减叠加两种方式，叠加结果为区简单要素类，结果属性和原始区简单要素类相同； 区对区叠加: 合并、相交、相减，思路及核心代码如下所示：

- 叠加分析之裁剪分析

```
// 自定义图形转换为几何模型
Geometry geometryA = Graphic.toGeometry(graphicPolygonA);
Geometry geometryB = Graphic.toGeometry(graphicPolygonB);
// 得到裁剪区
Geometry geometryClip = spaAnalisis.clip(geometryA,(GeoPolygon) geometryB);
```

- 叠加分析之求交分析

```
Geometry geometryA = Graphic.toGeometry(graphicPolygonA);
Geometry geometryB = Graphic.toGeometry(graphicPolygonB);
// 得到相交图形
Geometry geoIntersection = spaAnalisis.intersection(geometryA,geometryB);
```

- 叠加分析之擦除分析

```
//A,B 两个多边形，得到擦除后几何图形
Geometry geoDifference = spaAnalisis.difference(geometryA, geometryB);
```

- 叠加分析之求并分析

```
//得到 A，B 两个几何图形的并集
Geometry geoPolygonA = Graphic.toGeometry(graphicPolygonA);
Geometry geoPolygonaB = Graphic.toGeometry(graphicPolygonB);
Geometry geometryUnion = spaAnalisis.union(geoPolygonA, geoPolygonaB);
```

14 POI 查询

POI 查询，根据用户输入的关键字进行模糊查询，寻找到用户感兴趣的地点并展示出来核心代码以及思路如下所示：

1. 设置 Poi 查询的条件

```
//设置查询条件
PoiSearch.Query mQuery = new PoiSearch.Query(inputString, null, null);
PoiSearch mPoiSearch = new PoiSearch(mQuery);
```

2. 设置 Poi 查询的相关地图

```
mPoiSearch.initWithMap(mapView.getMap());
```

3. 获取 Poi 查询的结果

```
//获取 Poi 查询的结果
searchPoiResult = mPoiSearch.searchPOI();
```

4. 显示查询结果

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

```
//加载自定义布局
LinearLayout linLayout = (LinearLayout) getLayoutInflater().inflate(
                                                    R.layout.common_list, null);
ListView listView = (ListView) linLayout.findViewById(R.id.common_listview);
TextView textTitle = (TextView) linLayout.findViewById(R.id.common_title);
textTitle.setText("POI 信息");
//定义适配器 ListAdapter
CommAdapter adapterResult = new CommAdapter();
adapterResult.setContext(getApplicationContext());
adapterResult.setListField(searchResultList);
listView.setAdapter(adapterResult);
builder.setView(linLayout);
final AlertDialog dlg = builder.create();
dlg.show();
```

15 路径规划

通过起点和终点以及偏好，规划出想要的路径。核心代码及思路如下所示：

1. 确定地图上的两点

2. 获取两点之间存在的路径

```
//地图坐标转换成定位坐标
transPnts[0]=mapView.mapPointToLocation(points[0]);
transPnts[1]=mapView.mapPointToLocation(points[1]);
FromAndTo fromAndTo = new FromAndTo(transPnts[0],transPnts[1]);
//获取两点之间存在的路径
List<Route> testRoutes = rtAnalysis.calculateRoute(fromAndTo,
                                    rtAnalysis.DrivingLeastDistance, null, null, 0);
```

3. 取出一条路径，并获取外包

```
//取出第一条路径
testRoute = testRoutes.get(0);
//获取外包
Rect bRect = testRoute.getBoundingRect();
```

4. 获取该路径的路段数目以及路径描述

```
//获取路段数目
int segCount = testRoute.getStepCount();
//获取路径描述
String overViewString = testRoute.getOverview();
```

5. 获取路径坐标

```
//获取路径坐标
Dot [] coorsDots = testRoute.getCoors();
Dot [] routeDots    = new Dot[coorsDots.length];
for (int i = 0; i < coorsDots.length; i++){
routeDots[i] = mapView.locationToMapPoint(coorsDots[i]);}
```

6. 设置路径显示样式及注记

```
//设置显示的文本
GraphicText routeText = new GraphicText(routeDots[0], overViewString);
routeText.setFontSize(25);
//设置绘制的路径
GraphicPolylin routeLine = new GraphicPolylin(routeDots);
routeLine.setColor(Color.argb(255, 0, 0, 255));
routeLine.setLineWidth(10);
```

7. 绘制路径及注记，如下图所示：


计算路径
模拟导航
请输出起点:
设置起点
请输出终点:
设置终点
路径计算
华中科技大学
珞瑜路
旺山公园
虎泉街
雄楚大道
高新大道
南湖园
武汉职业技术学院
南湖大道
南湖
授权给85099100开发使用
三环线
1公里
MapGIS