KOTEI

武汉光庭信息技术股份有限公司
WUHAN KOTEI INFORMATICS CO., LTD.

Android 地图 SDK

开发者指南

Ver.1.0

目 录

Android 地图 SDK 功能简介

1 文档介绍

本 Android3D 地图 SDK 1.0 发布! 本 Android SDK 是一套地图开发调用接口，供开发者在自己的 Android 应用中加入地图相关的功能。开发者可以轻松地开发出地图显示与操作、地图定位、离线地图等功能

1.1 文档目的

全面提供本 Android SDK 开发接口以及功能介绍，完美辅助开发者进行本 SDK 开发

1.2 读者对象

所有开发本 Android SDK 的开发者

2 简介

2.1 Android SDK

本 Android SDK 可以为 Android 应用开发者提供互动的、功能丰富的 Android 手机地图。

将地图显示功能，搜索服务与定位服务分别封装为三个类库。每个类库不相互依赖，用户可以分开使用。完美支持 Android 手机、平板电脑，可在不同屏幕尺寸下呈现完美的显示效果。地图采用矢量方法绘制，使得地图处理速度更快、流量占用更少。地图支持 3D 模式。通过移动用户的视角，可以从各个角度显示地图。全新动画效果。移动地图、切换用户视角时，可体验炫酷的动画效果。支持 3D 离线地图。用户下载离线地图数据包后，可离线查看城市地图。

2.2 面向的读者

本 Android SDK 是提供给具有一定 Android 编程经验和了解面向对象概念的读者使用的。此外，读者还应该对地图产品有一定的了解。在使用中遇到任何问题，都可以随时联系我们。

3 配置工程

将给予的 jar 包放置在项目的 lib 目录下即可
如若出现不可使用的情况,请右键 jar 包→BuildPath→Add to Build Path 即可
将给予的 so 文件放置在 libs 目录下的 armeabi 目录下即可

4 地图显示

4.1 显示地图

以 MapView 显示地图为例进行说明:

MapView 是 Android View 类的一个子类，它可以帮助您在 Android View 中放置地图，它是应用程序和窗口部件的基本构建类。MapView 作为地图的容器，通过 map 对象显示地图。使用 MapView 类，必须重载 Activity 生命周期的所有方法，有 onCreate()，onDestroy()，onResume()，onPause()，onSaveInstanceState()。

4.2 旋转地图(3D)

map.setPointToCenter(100,0)设置屏幕像素点(100,0)为地图中心点,cameraupdateFactory.changeBearing(90)改变地图的旋转角度,即表示地图以屏幕像素点旋转地图

4.3 调整线级面层级(3D)

使用 map 类的 setMapTextZIndex()方法来设置地图底图文字的 z 轴指数.地图底图文字和覆盖物的 z 轴指数默认为 0，此时地图底图文字在覆盖物的下方，如果使用 map.setMapTextZIndex(2) 可以将地图底图文字设置在添加的覆盖物之上。

5 地图图层

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。开发者可以通过设置 Map 类，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。

5.1 基础地图

本 SDK 提供 3 种地图类型分别是 0(白天),1(黑夜),2(卫星)

```
map.setMapType(0); //白天
map.setMapType(1); //黑夜
map.setMapType(2); //卫星
```

5.2 定位层

开发者需要调用以下代码设置定位资源并显示定位层。

map.setLocationSource(true);// 设置定位监听

map.setMyLocationEnabled(true);// 设置为 true 表示显示定位层并可触发定位，false 表示隐藏定位层并不可触发定位，默认是 false

```
/**
* 设置定位的类型
* @param type 提供 3 种类型：
* LOCATION_TYPE_LOCATE 表示只在第一次定位移动到地图中心点，值为 1；
* LOCATION_TYPE_MAP_FOLLOW 表示定位、移动到地图中心点并跟随，值为 2；
* LOCATION_TYPE_MAP_ROTATE 表示定位、移动到地图中心点，跟踪并根据方向旋转地图，值为 3。
*/
map.setMyLocationType(1);//设置定位类型
```

更改定位图标

调用定位 SDK，点击定位按钮，默认地图上您所在的位置显示蓝色的小圆点，以及圆形精度范围。您可以根据自己的喜好，自定义设置“我的位置”的样式。绘制定位样式需要在地图初始化成功之后进行操作。

示例代码如下

```
// 自定义系统定位蓝点
MyLocationStyle myLocationStyle = new MyLocationStyle();
// 自定义定位蓝点图标
myLocationStyle.myLocationIcon(BitmapDescriptorFactory.
romResource(R.drawable.location_marker));
// 自定义精度范围的圆形边框颜色
myLocationStyle.strokeColor(Color.BLACK);
//自定义精度范围的圆形边框宽度
myLocationStyle.strokeWidth(5);
// 将自定义的 myLocationStyle 对象添加到地图上
map.setMyLocationStyle(myLocationStyle);
// 构造 LocationManagerProxy 对象
PositionManager pm= PositionManager.getInstance();
//设置定位资源。如果不设置此定位资源则定位按钮不可点击。
map.setLocationSource(true);
//设置默认定位按钮是否显示
map.getUiSettings().setMyLocationButtonEnabled(true);
// 设置为 true 表示显示定位层并可触发定位，false 表示隐藏定位层并不可触发定位，默认是 false
map.setMyLocationEnabled(true);
```

定位类型

目前支持 3 种定位类型：

LOCATION_TYPE_LOCATE：只在第一次定位移动到地图中心点。

LOCATION_TYPE_MAP_FOLLOW：定位、移动到地图中心点并跟随。

LOCATION_TYPE_MAP_ROTATE：定位、移动到地图中心点，跟踪并根据面向方向旋转地图。

```
map.setLocationSource(true);// 设置定位监听
map.getUiSettings().setMyLocationButtonEnabled(true);// 设置默认定位按钮是否显示
map.setMyLocationEnabled(true);// 设置为 true 表示显示定位层并可触发定位，false 表示隐藏定位层并不可触发定位，默认是 false
//设置定位的类型为定位模式 ，可以由定位、跟随或地图根据面向方向旋转几种
map.setMyLocationType(1);
// 设置定位的类型为定位模式
map.setMyLocationType(1);
```

```
// 设置定位的类型为 跟随模式
map.setMyLocationType(2);
// 设置定位的类型为根据地图面向方向旋转
map.setMyLocationType(3);
```

6 定位

6.1 简介

获取当前位置,以下将会针对地图上定位显示效果进行介绍

地图支持定位效果如下，可以根据应用实际场景进行选择。

定位模式，即只第一次定位到地图中心点显示

跟随模式，即每次定位结果地图居中显示

旋转模式，即每次定位结果地图居中显示，并且定位图标跟随手机方向旋转

6.2 示例代码

```
map.setLocationSource(false);// 设置定位监听 false 为启动
map.getUiSettings().setMyLocationButtonEnabled(true);// 设置默认定位按钮是否显示
map.setMyLocationEnabled(true);// 设置为 true 表示显示定位层并可触发定位，false 表示隐藏定位层并不可触发定位，默认是 false
// 设置定位的类型为定位模式，参见类 map。
map.setMyLocationType(1);
```

7 覆盖物显示

所有叠加或覆盖到地图的内容，统称为地图覆盖物。如标记、矢量图形元素（包括：折线、多边形和圆）等。覆盖物拥有自己的地理坐标，当您拖动或缩放地图时，它们也会随地图移动。

7.1 折线(polyline)

折线的关键类为 Polyline，在地图上定义了一组相连的线段。Polyline 对象由一组经纬度坐标组成，并以有序序列形式建立一系列的线段。

7.2 添加折线

类 PolylineOptions，折线的选项类，增加线段、可见性，设置实线、虚线等。在 PolylineOptions 对象中调用 PolylineOptions.add()方法添加经纬度的点对象。绘制线段，将它们以点点之间的顺序添加到 PolylineOptions 对象中。起点与终点坐标相同，则为闭合的多边形。

PolylineOptions.add() 方法存在可变数目的参数，所以您可以在同一时间添加多个点（如果点已经在列表中，您可以调用 PolylineOptions.addAll() 方法。最后，调用 map.addPolyline(PolylineOptions) 将折线添加到地图上。这个方法返回一个 Polyline 对象。

7.3 设置外观

边框宽度

使用 PolylineOptions.width() 设置折线的宽度。默认边框宽度为 10 像素。折线添加到地图之后，可以使用 Polyline.getWidth() 和 Polyline.setWidth(float width) 方法查看或改变边框宽度。

边框颜色

使用 PolylineOptions.color() 设置边框颜色。边框颜色是 ARGB 格式。默认画笔颜色为黑色。折线添加到地图之后，可以使用 Polyline.getColor() 和 Polyline.setColor(int color) 查看或改变边框颜色。

Z 轴指数

Z 轴指数是控制地图覆盖物（overlay）之间的绘制层次的参数。这个参数能够控制 Circle、Polygon、Polyline 的绘制层次，但不会影响标记 Marker。Z 轴数值越大的覆盖物（overlay）将会绘制在更上层。如果两个及两个以上覆盖物（overlay）的 Z 轴数值相同，则最后的绘制结果是随机的。覆盖物（overlay）Z 轴指数默认为 0。
使用 PolylineOptions.zIndex() 设置折线的 Z 轴指数。折线添加到地图之后，可以通过 Polyline.getZIndex() 或 Polyline.setZIndex (float zIndex) 查看或改变绘制层次。

可见性

可见性决定形状是否绘制在地图上，其中，“true”表示体现在地图上，“false”表示不出现。它允许您在地图上暂时不显示形状。要永久删除地图的形状，可调用 Polyline.remove() 方法。
使用 PolylineOptions.visible() 设置折线的可见性。折线添加到地图之后，可以通过 Polyline.isVisible() 或 Polyline.setVisible() 查看或改变可见性。

边框虚线

折线的边框默认是以实线样式来绘制，如果要设置虚线样式，可以使用 PolylineOptions.setDottedLine(true) 设置折线边框样式为虚线。在折线添加到地图之后，可以通过 Polyline.isDottedLine() 查看折线是否是虚线样式。

7.4 多边形

类 Polygon 与 Polyline 比较相似，它们都包括有序序列的一系列坐标。然而，多边形包含有内部区域。

添加多边形

首先创建 PolygonOptions 类，添加一些点指向它。这些点形成多边形的轮廓。然后调用 map.addPolygon(PolygonOptions) 将多边形添加到地图中，返回一个 Polygon 对象。

7.5 设置外观

填充颜色

填充颜色仅适用于多边形和圆，折线没有填充颜色。
填充颜色是 ARGB 格式。使用 PolygonOptions.fillColor() 设置多边形的填充颜色。默认没有填充颜色。多边形添加到地图之后，可以使用 Polygon.getFillColor() 和

Polygon.setFillColor() 方法查看或改变填充颜色。

其他

多边形的边框颜色、边框宽度、可见性、Z 轴指数可参见添加折线的介绍。方法名称略有不同。

7.6 圆

除了通用的 Polygon 类，本 Android SDK 还定义了 Circle 类，简化创建圆的过程。要构建一个圆，您必须指定以下两个属性：

中心点（经纬度）

半径（米）

添加圆

创建一个 CircleOptions 对象，使用 CircleOptions.center(LatLng point) 和 CircleOptions.radius(double radius) 设置中心点和半径，最后调用 map.addCircle(CircleOptions) 将圆添加到地图上。

注意事项：创建圆时，不能指定两个经纬度坐标来绘制圆形。

设置外观

圆的边框颜色、边框宽度、可见性、Z 轴指数可参见“添加折线”的介绍。方法名称略有不同。圆的填充颜色，可参见“添加多边形”的介绍。

7.7 圆层覆盖片(GroundOverlay)

图片覆盖层是将一张图片以合适的大小贴在地图指定的位置上。

图片覆盖层的属性如下：

anchor 图片和指定点的对齐方式，[0,0]是左上角，[1,1]是右下角 。如果不设置，默认为[0.5,0.5]图片的中心点。

bearing 图片覆盖层相对锚点从正北顺时针的旋转角度。

image 此图片覆盖层的图片。

position 根据描点和宽高设置图片层。在显示时，图片会被缩放来适应指定的尺寸。

transparency 图片覆盖层的透明度。默认透明度为 0，不透明。

visible-图片覆盖层是否可见。默认为可见。

zIndex 图片覆盖层的 z 轴指数。

添加图片覆盖层

需要通过设置中心点或者图片区域来确定图片覆盖层的位置，初始化一个 GroundOverlayOptions 对象来设置图片层图片区域、透明度、锚点、要显示的图片，使用 map.addGroundOverlay(GroundOverlayOptions) 将此图片层添加到地图。

```
// 设置当前地图显示为北京市恭王府
map.moveCamera(CameraUpdateFactory.newLatLng(newLatLng(39.936713,116.386475)));
```

//设置图片的显示区域。

LatLngBounds bounds = new LatLngBounds.Builder()

.including(new LatLng(39.935029, 116.384377))

.including(new LatLng(39.939577, 116.388331));

GroundOverlay groundoverlay = map.addGroundOverlay(new GroundOverlayOptions().anchor(0.5f,0.5f).transparency(0.1f).image(BitmapDescriptorFactory.fromResource(R.drawable.groundoverlay)).positionFromBounds(bounds));

7.8 标记(Marker)

添加默认标记

标记显示地图上的单一位置。它可以使用一个标准的图标，也可以由开发者自定义图标。您可以通过 map.addMarker(markerOptions) 方法将一个标记添加到地图上。

标记的属性如下：

position(Required) 在地图上标记位置的经纬度值。参数不能为空。

title 当用户点击标记，在信息窗口上显示的字符串。

snippet 附加文本，显示在标题下方。

draggable 如果您允许用户可以自由移动标记，设置为“ true ”。默认情况下为“ false ”。

visible 设置“ false ”，标记不可见。默认情况下为“ true ”。

anchor 图标摆放在地图上的基准点。默认情况下，锚点是从图片下沿的中间处。

perspective 设置 true，标记有近大远小效果。默认情况下为 false。

可以通过 Marker.setRotateAngle() 方法设置标记的旋转角度，从正北开始，逆时针计算。如设置旋转 90 度，

通过 setFlat() 方法设置标志是否贴地显示。默认情况下为“false”，不贴地显示。

自定义标记图标

您如果需要改变标记图像，可以设置自定义的图像，通常被称为图标。自定义图标通常由 BitmapDescriptor 设置。您可以在类 BitmapDescriptorFactory 使用以下其中一种方法定义。

fromAsset(String assetName) 在 assets 目录中使用图像创建自定义标记。

fromBitmap (Bitmap image) 使用位图图像创建自定义标记。

fromFile (String path) 指定路径的文件创建自定义图标。

fromResource (int resourceId) 使用已经存在的资源创建自定义图标。

动画标记

动画标记允许用户在地图上添加一个动态的标记。用户通过设置动画标记的每帧显示的图片和刷新一次动画资源的时间来设置动画效果。在原有默认标记的基础上增加动画标记的属性，如下：

icons 设置标记图标的动画帧列表。

period 设置标记动画刷新一次图片资源的周期。

弧形(Arc)

本 Android SDK 定义了 Arc 类用于创建弧形对象。

添加弧形

创建一个 ArcOptions 对象，使用 ArcOptions.points(LatLng startpoint, LatLng passpoint, LatLng endpoint) 设置弧形的起点、途经点和终点，最后调用 map.addArc(ArcOptions) 将弧形添加到地图上。

注意事项：设置点时，必须按照起点、途经点、终点的顺序。

设置外观

弧形的边框颜色、边框宽度、可见性、Z 轴指数可参见“添加折线”的介绍。

```
// 绘制一个经过北京的弧形
        ArcOptions arcOptions = new ArcOptions().points(
                new LatLng(47.828, 85.621),
                new LatLng(43.828, 87.621),
    New     LatLng(45.808,126.55)).strokeColor(Color.RED);
        map.addArc(arcOptions);
```

7.9 点击标记事件

您可以使用 map.setOnMapMarkerClickListener 监听点击标记的事件。在地图上设置监听器，需调用 map.setOnMapMarkerClickListener(OnMarkerClickListener) 。当用户点击标记，方法 onMapMarkerClick(Marker) 将被调用，标记将会作为参数传递。方法返回布尔型，表示是否执行事件（例如：您想要阻止默认操作）。如果返回 “false”，除您定义的操作之外，默认操作将会被执行。点击标记的默认操作是显示它的信息窗口（如果可用）。

```
map.setOnMapMarkerClickListener(this);// 设置点击 marker 事件监听器
map.setOnMapMarkerDragListener(this);// 设置 marker 可拖拽事件监听器
```

7.10 点击信息窗口事件

您可以使用 map.OnInfoWindowClickListener 去监听点击信息窗口的事件，map.InfoWindowAdapter 显示信息窗口。在地图上设置监听器，需调用 map.setOnInfoWindowClickListener(OnInfoWindowClickListener) 。当用户点击信息窗口，onInfoWindowClick(Marker) 方法将被调用。

说明：当使用接口 map.InfoWindowAdapter（显示信息窗口）的 getInfoWindow(Marker) 方法时，请确认标记（Marker）的 title 或 snippet 已赋值。

示例代码

```
map.setOnMapMarkerClickListener();// 设置点击 marker 事件监听器
map.setOnInfoWindowClickListener(this);// 设置点击 infoWindow 事件监听器
map.setInfoWindowAdapter(this);// 设置自定义 InfoWindow 样式
```

7.11 拖拽标记事件

您可以使用 map.onMapMarkerDragListener 去监听拖动标记的事件。在地图上设置监听器，需调用 map.setOnMapMarkerDragListener(OnMarkerDragListener)。拖拽标记，用户的手指长按住标记在屏幕上滑动到什么地方，标记就会停留在那个地方。当标记被拖动，首先调

用 onMapMarkerClick(Marker)；拖拽过程中，不断调用 onMapMarkerDrag(Marker)；拖拽结束，调用 onMapMarkerDragEnd(Marker)。您可以通过调用 Marker.getPosition() 随时获取标记的位置。

说明：默认情况下，标记时不可以拖拽的。用户在拖拽前，必须明确设置它的拖拽属性。在添加标记前，您可以通过使用 MarkerOptions.draggable(boolean) 设置；或者标记添加后，使用 Marker.setDraggable(boolean) 进行设置。

8 地图事件

8.1 点击地图

如果您需要响应用户轻点地图上某个点的事件，您需要在地图上调用 map.setOnMapClickListener(OnMapClickListener) 实现。当用户在地图上点击（轻拍）某个地方，您将收到 onMapClick(LatLng) 事件，显示用户点击地图上的位置。使用时必须在主线程中进行调用。

8.2 长按地图

调用 map.setOnMapLongClickListener(OnMapLongClickListener)，您可以监听用户长时间点击地图的事件。这个监听与点击的监听类似，您通过设置 onMapLongClick(LatLng) 可以接到长按事件的通知。使用时必须在主线程中进行调用。

8.3 移动地图

调用 map.setOnCameraChangeListener(OnCameraChangeListener)，您可以监听用户改变可视区域的事件。可视区域改变时回调 onCameraChange(CameraPosition postion) 方法；当改变地图可视区域的操作（如拖动、动画滑动、缩放）完成之后回调 onCameraChangeFinish(CameraPosition postion) 方法。使用时必须在主线程中进行调用。

8.4 触摸地图

如果您需要响应用户触摸地图上的事件，您需要在地图上调用 map.setOnMapTouchListener(OnMapTouchListener) 实现。触摸到地图将收到 onTouch(MotionEvent event) 事件，获取触摸事件发生的位置。使用时必须在主线程中进行调用。

示例代码

```
/**
 * 分别实现 OnMapClickListener, OnMapLongClickListener,
 * OnCameraChangeListener、OnMapTouchListener 四种监听器接口
 */
    /**
     * map 添加事件监听器
     */
     map.setOnMapClickListener(this);// 对 map 添加单击地图事件监听器
     map.setOnMapLongClickListener(this);// 对 map 添加长按地图事件监听器
```

```
map.setOnCameraChangeListener(this);// 对 map 添加移动地图事件监听器
map.setOnMapTouchListener(this);// 对 map 添加触摸地图事件监听器
    /**
* 实现回调方法
*/
//对单击地图事件回调
Public void onMapClick(LatLng point) {}
//对长按地图事件回调
Public void onMapLongClick(LatLng point) {}
//对正在移动地图事件回调
Public void onCameraChange(CameraPosition cameraPosition) {}
//对移动地图结束事件回调
Public void onCameraChangeFinish(CameraPosition cameraPosition) {}
//对触摸地图事件回调
Public void onTouch(MotionEvent event) {}
```

9 地图控件

本 Android SDK 提供一些内置的 UI 控件。您可以通过 map 里的 map.getUiSettings() 方法获得 UiSettings 对象，去控制这些控件的可见性。在类中改变模式可以立即反应在地图上。当地图创建时，可以使用 mapOptions 类来设置这些选项。

9.1 缩放控件

本 Android SDK 提供内置的缩放控制键，显示在地图的右下角。默认情况下是开启的，但是可以通过调用 UiSettings.setZoomControlsEnabled(boolean) 禁用。

```
Case R.id.zoom_toggle:
    mUiSettings.setZoomControlsEnabled(true);
break;
```

定位按钮

默认情况下，定位按钮不显示。您可以调用 UiSettings.setMyLocationButtonEnabled(boolean) 启用该按钮。调用定位 SDK，定位位置。启用定位按钮后，定位按钮将会出现在屏幕的右上角。当用户点击这个按钮，如果用户的位置当前是可知的，可视区域将显示用户的当前位置。设置定位按钮显示后，如果定位层不显示 map.setMyLocationButtonEnabled(false) 时，定位按钮可见但不可点击。

示例代码

```
case R.id.mylocation_toggle:
    map.setLocationSource(true);//设置定位监听
    mUiSettings.setMyLocationButtonEnabled(true);// 是否显示定位按钮
    map.setMyLocationEnabled(true);// 是否可触发定位并显示定位层
```

地图 logo

您可以通过 UiSettings.setLogoPosition(int position) 方法设置 Logo 的位置。Logo 的位置有左下角、底部居中、右下角三种选择。

示例代码

```
if (map != null) {
 if (checkedId == R.id.bottom_left) {

mUiSettings.setLogoPosition(MapOptions.LOGO_POSITION_BOTTOM_LEFT);
 } else if (checkedId == R.id.bottom_center) {

mUiSettings.setLogoPosition(MapOptions.LOGO_POSITION_BOTTOM_CENTER);
 } else if (checkedId == R.id.bottom_right) {
     mUiSettings.setLogoPosition(MapOptions.LOGO_POSITION_BOTTOM_RIGHT);
 }
}
```

9.2 指南针

本 Android SDK 提供指南针功能。当可视区域旋转（方向非 0 度）或倾斜（倾角非 0 度）时，指南针会始终指向地图的正北方向。当用户点击指南针图标时，可视区域将返回默认位置（方向和角度为 0 度）。默认状态下，指南针功能不可用，您可以调用 UiSettings.setCompassEnabled(boolean enabled) 启用指南针功能。

```
caseR.id.compass_toggle:
     mUiSettings.setCompassEnabled(true);
break;
```

9.3 比例尺

本 Android SDK 提供比例尺功能。缩放级别不同，比例尺代表的长度也不同。比例尺代表的长度范围为 5m - 500km，由缩放级别决定。默认状态下，比例尺功能不可用。您可以调用 UiSettings.setScaleControlsEnabled(boolean enabled) 启用此功能。

调用 map.getScalePerPixel() 方法，根据用户当前地图的 zoom 值和地图中心点的坐标，可以获得当前比例尺的数据。

示例代码

```
/**
 * 一像素代表多少米
 */
Case R.id.buttonScale:
     floatscale = map.getScalePerPixel();
     ToastUtil.show(UiSettingsActivity.this, "每像素代表"+ scale + "米");
     break;
/**
 * 设置显示地图的默认比例尺
 */
Case R.id.scale_toggle:
```

```
mUiSettings.setScaleControlsEnabled(true);
break;
```

10 手势控制

10.1 缩放手势

改变可视区域的缩放级别，地图可以响应的手势如下：

双击地图可以使缩放级别增加 1 (放大)

两个手指捏/拉伸

调用类 UiSettings 的 setZoomGesturesEnabled(boolean) 方法您可以禁用缩放手势。这不会影响用户使用地图上的缩放控制按钮。

示例代码

```
Case R.id.zoom_gestures_toggle:
    mUiSettings.setZoomGesturesEnabled(false);
    break;
```

平滑(滑动)手势

用户可以用手指拖动地图四处滚动（平移）或用手指滑动地图（动画效果）。您通过调用类 UiSettings 的 setScrollGesturesEnabled(boolean) 方法可以禁用平移（滑动）手势。

示例代码

```
Case R.id.scroll_toggle:
    mUiSettings.setScrollGesturesEnabled(false);
    break;
```

10.2 旋转手势(3D)

在 3D 地图上，用户可以用两个手指在地图上旋转。您可以通过调用类 UiSettings 的 setRotateGesturesEnabled(boolean) 方法禁用旋转手势。

```
Case R.id.rotate_toggle:
    mUiSettings.setRotateGesturesEnabled(false);
    break;
```

10.3 倾斜手势(3D)

在 3D 地图上，用户可以在地图上放置两个手指，移动它们一起向下或向上去增加或减小倾斜角。您可以通过调用类 UiSettings 的 setTiltGesturesEnabled(boolean) 方法禁用倾斜手势。

```
Case R.id.tilt_toggle:
    mUiSettings.setTiltGesturesEnabled(false);
    break;
```

11 可视区域操作

11.1 可视区域的位置(CameraPosition)

可视区域的位置由以下属性组成：

目的地（target）

缩放级别（zoom）

方向（bearing）

倾斜角度（tilt）

屏幕当前可视区域的位置可以通过 map.getCameraPosition() 方法获取。

目的地（target）

地图的中心位置。位置使用经度和纬度说明。

示例代码

```
LatLng mTarget = map.getCameraPosition().getTarget();
```

缩放级别（zoom）

可视区域的缩放级别决定了地图的比例。地图缩放级别为 4-20 级，缩放级别不必是一个整数。

缩放级别较低时，您可以看到更多地区的地图；缩放级别高时，您可以查看地区更加详细的地图。

下面的图像显示出不同的缩放级别：

```
// 获取当前地图的缩放级别
floatmZoom = map.getCameraPosition().getZoom();
```

方向（bearing）

默认情况下，地图的方向为 0 度，屏幕正上方指向北方。当您逆时针旋转地图时，地图正北方向与屏幕正上方的夹角度数为地图方向，范围是从 0 度到 360 度。例如，一个 90 度的查询结果，在地图上的向上的方向指向正东。

```
// 获取当前地图的旋转角度
floatmBearing = map.getCameraPosition().getBearing();
```

倾斜角度（tilt）

可视区域的位置在地图的中心位置和地球的表面之间的圆弧上，从最低点以度测量（方向直接指向下方的可视区域）。当您改变视角，地图显示的角度来看，远方的建筑变小，附近的建筑则变大，产生立体效果。

地图倾角范围为 0-45 度。

```
// 获取当前地图的倾斜角度
floatmTilt = map.getCameraPosition().getTilt();
```

11.2 移动可视区域

地图 Android SDK 允许您在屏幕上切换显示的地图区域。这是通过改变可视区域的位置实现的。改变可视区域的位置，需要您明确想要把可视区域移动到哪里。您可使用 CameraUpdateFactory 创建不同类型的 CameraUpdate。以下的选项可供使用。

仅更改视图中心点

有三个简便的方法可以改变位置：

CameraUpdateFactory.changeLatLng(LatLng)，仅改变可视区域中心点的坐标，其他属性不变。

CameraUpdateFactory.newLatLng(LatLng)，可以改变可视区域的经纬度，缩放级别默认为固定值 4，保留其他属性。

仅更改旋转角度

使用 CameraUpdateFactory.changeBearing(float bearing) 方法可以改变可视区域的方向并且保留其他属性。返回一个 CameraUpdate 对象，使用 map.moveCamera(CameraUpdate update) 方法更新可视区域显示在地图上。

仅更改倾斜角度

使用 CameraUpdateFactory.changeTilt(float tilt) 方法可以改变可视区域的倾斜角度并且保留其他属性。返回一个 CameraUpdate 对象，使用 map.moveCamera(CameraUpdate update) 方法更新可视区域显示在地图上。

更改可视区域

为了改变可视区域的位置更加灵活，可以使用 CameraUpdateFactory.newCameraPosition(CameraPosition) 移动可视区域到指定位置。CameraPosition 可以用以下方法直接获取：使用 new CameraPosition() 或使用 new CameraPosition.Builder()。

返回一个 CameraUpdate 对象，使用 map.moveCamera(CameraUpdate update) 方法更新可视区域显示在地图上。

12 离线地图(3D)

使用 3D 地图应用时，可以使用离线地图来降低流量消耗，提高访问速度。

12.1 开始下载

目前您可以根据城市编码和城市名称两种方式下载该城市的离线地图。使用方法为

示例代码:

```
//按照 citycode 下载
OfflineMapManager.downloadByCityCode(String citycode)
//按照 cityname 下载
OfflineMapManager.downloadByCityName(String cityname)
```

12.2 暂停下载

离线地图下载过程中，可以通过代码暂停地图的下载。

```
OfflineMapManager manager=new OfflineMapManager(this, null);
        manager.pause();
```

停止下载

```
OfflineMapManager manager=new OfflineMapManager(this, null);
        manager.stop();
```

12.3 更改存储目录

默认离线地图数据下载到手机存储卡的“storage/sdcard/tminavi/mapdata”目录下，您也可以参照示例代码自定义路径。

示例代码

```
OfflineMapManager manager=new OfflineMapManager(this, null);
        String path="";
        manager.setOfflineMapSavepath(path);
```

12.4 获取城市列表

使用 OfflineMapManager.getOfflineMapCityList() 方法您可以查看离线地图可下载城市的列表。

12.5 获取全国数据

OfflineMapProvince 类可以用来获取全国包括各省、市离线地图的数据。

12.6 获取已下载城市列表

使用 OfflineMapManager.getDownloadOfflineMapCityList() 方法您可以查看已下载的城市列表。

12.7 获取正在等待下载城市列表

使用 OfflineMapManager.getDownloadingCityList() 方法您可以查看正在或等待下载的城市列表。

12.8 检查更新

示例代码

```
//通过 updateOfflineCityByName 方法判断该地图包是否存在更新
update = manager.updateOfflineCityByName(city);
```

12.9 删除离线地图包

```
manager.remove(cityname);
```