

ArcGIS API for Android 开发教程

Esri（中国）有限公司 吴泳锋 程轩昂

目录

前言	3
1 ArcGIS API for Android 系统要求	4
1.1 Eclipse IDE 要求	4
1.2 支持的 Android SDK 平台	4
1.3 支持的 Server 版本和平台	4
1.4 搭建 ArcGIS API for Android 环境配置	5
1.5 Eclipse 和 Android SDK 关联	5
1.6 配置虚拟设备	7
1.7 安装 ArcGIS API For Android 插件	9
2 新建示例工程	10
2.1 新建一个 HelloWorld 工程	10
2.2 运行这个 ArcGIS Android 程序	11
3 开发手机地图	12
3.1 地图服务	12
3.2 动态操作地图服务	13
3.3 导航与触屏操作	16
3.4 客户端要素图层	20
3.5 通过交互绘制几何对象	21
4 查询和识别	25
4.1 空间查询和属性查询	25
4.2 要素识别	28
5 几何对象操作与地理处理	29
5.1 几何对象的操作	29
5.2 地理处理服务	30
6 要素编辑	32
6.1 Feature Layer	32
6.2 属性编辑	34
6.3 几何编辑	35
7 在线资源	40

前言

Android 是什么？

Android 一词的本义指“机器人”，同时也是 Google 于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，该平台由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放和完整的移动软件。自 2007 年发布以来，Android 系统逐步为移动设备提供商和用户所接受。2010 年底的统计显示，Android 系统已经以 32.9% 的市场占有率雄踞智能移动设备操作系统之首。

ArcGIS For Android 介绍

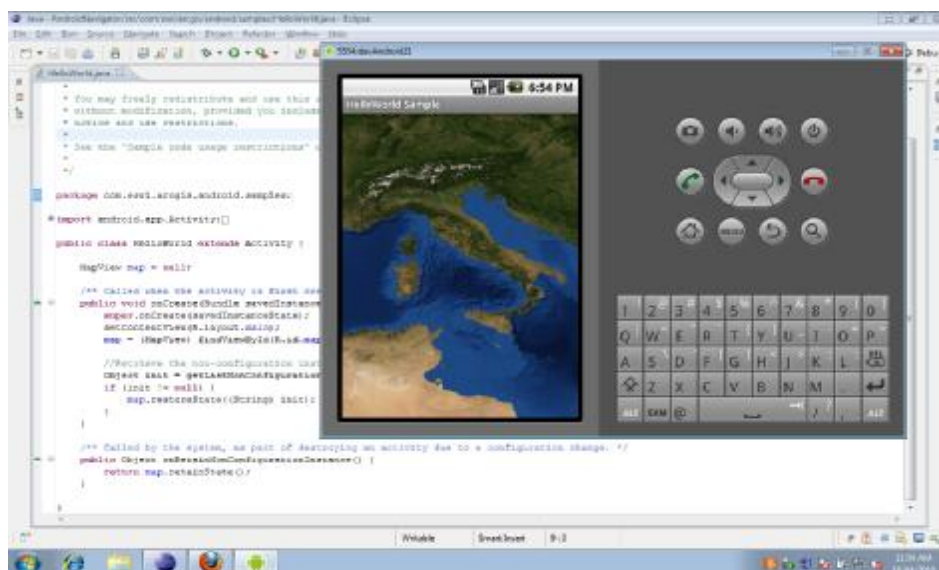
ArcGIS for Android 将 GIS 的适用范围从办公室扩展到移动 Web。发布时，ArcGIS for Android 将包括一个应用程序，您将能够从 Android Market 应用商店下载这款称为 ArcGIS 的应用程序。这个应用程序将类似于已经发布的 ArcGIS for iOS 和 Windows Phone 应用程序。使用该 ArcGIS 应用程序，您能够浏览 ArcGIS.com 或 ArcGIS Server 提供的地图，并且利用程序中提供的工具进行搜索，识别位置和要素，测量线和面，以及编辑。



ArcGIS for Android 还包括一个 API，该 API 的测试版已经于 2011 年 2 月发布。ArcGIS API for Android 使您可使用 Java 构建多种应用程序，这些应用程序可以运用 ArcGIS Server 提供的强大制图、地理编码、地理处理和自定义功能实现复杂的业务功能，并将它们部署到 Android 设备。API 以一个 Eclipse 集成开发环境 (IDE) 开发插件的形式发布，其中提供了丰富的工具、文档和示例，可帮助开发人员使用 ArcGIS API for Android 创建应用程序。

ArcGIS Android API 依赖 ArcGIS Server 的 REST 接口。我们在 API 中加载地图服务，对要素进行添加删除等编辑操作，调用地理处理服务 (Geoprocessing Service, GP 服务) 等等，都可以通过对 ArcGIS Server 提供的 Rest 服务调用来实现。接下来的章节，我们将为您简要介绍在 ArcGIS Android API 中实现地图加载，要素查询和识别，调用 GP 服务，

要素编辑等操作的基本原理和实现方法，并以此来帮助您理解 API 大致的工作原理、开发涉及的主要概念等内容。



1 ArcGIS API for Android 系统要求

1.1 Eclipse IDE 要求

- ┆ The Android Development Tools (ADT) Plug-In for Eclipse 插件
- ┆ The ArcGIS API for Android Eclipse Plug-In for Eclipse 插件(该插件目前只支持 Eclipse3.5(Galileo)和 Eclipse3.6(Helios))
- ┆ ADT 安装之前需要安装 Eclipse Javadevelopment tools (JDT) 插件
- ┆ Java development kit (JDK) 6

1.2 支持的 Android SDK 平台

- ┆ SDK Platform Android 2.1, API 7
- ┆ SDK Platform Android 2.2, API 8

1.3 支持的 Server 版本和平台

- ┆ ArcGIS API for Android 支持 ArcGIS Server 9.3.1 和 10.0 的服务

1.4 搭建 ArcGIS API for Android 环境配置

Eclipse IDE 与 JDK

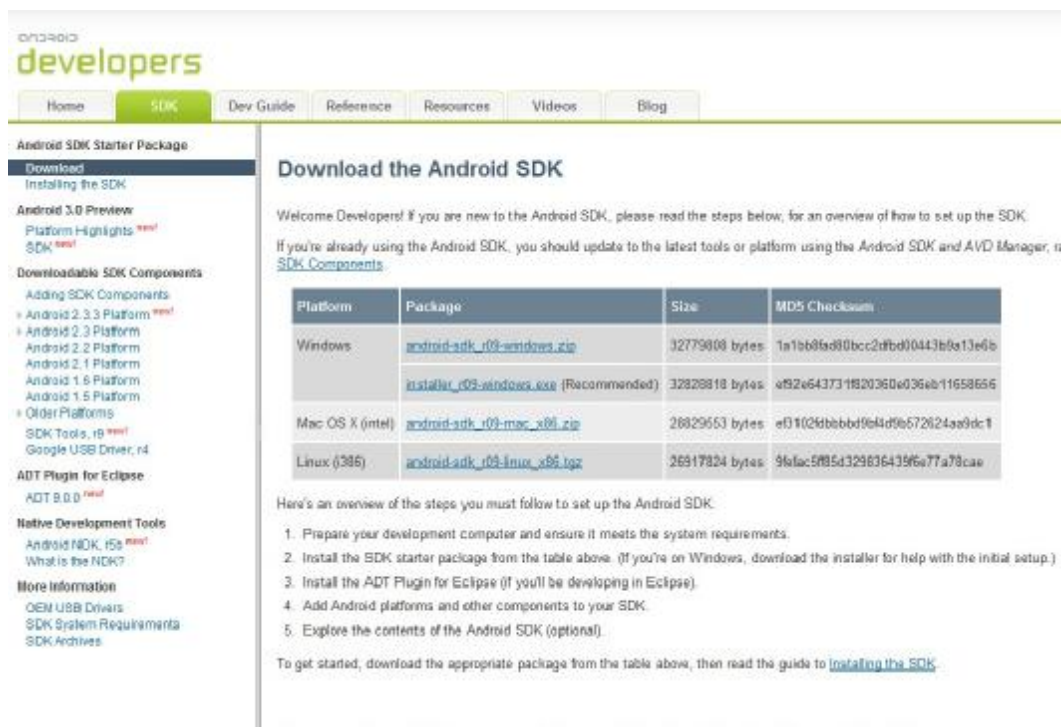
在搭建 ArcGIS API for Android 开发环境之前您需要先安装 Eclipse IDE, 并配置 JDK。相关的文档有很多, 这里就不在重复。需要您注意的是, 目前 ArcGIS API for Android 要求 Eclipse 3.5 或者 3.6 版本, JDK 6。

Android SDK 和 ADT 插件

首先, 可以从一下地址下载 Android SDK:

<http://androidappdocs.appspot.com/sdk/index.html>,

推荐下载 zip 压缩包, 下载完成可以直接解压。



The screenshot shows the 'Download the Android SDK' page from the Android Developers website. It includes a table with download links for Windows, Mac OS X (intel), and Linux (386). The table has columns for Platform, Package, Size, and MD5 Checksum.

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r09-windows.zip	32779008 bytes	1a1bb9fad80bcc2dfbd0443b3e13e6b
	installer_r09-windows.exe (Recommended)	32828818 bytes	e952e643731f820360e036eb11658656
Mac OS X (intel)	android-sdk_r09-mac_x86.zip	26829663 bytes	ef11024dbbbd964d96b72624a9dc1
Linux (386)	android-sdk_r09-linux_x86.tgz	26817824 bytes	9f6ac585d328636439f6e77a78cae

Below the table, there is a list of steps to set up the Android SDK:

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from the table above. (If you're on Windows, download the installer for help with the initial setup.)
3. Install the ADT Plugin for Eclipse (if you'll be developing in Eclipse).
4. Add Android platforms and other components to your SDK.
5. Explore the contents of the Android SDK (optional).

To get started, download the appropriate package from the table above, then read the guide to [installing the SDK](#).

Android SDK 下载页面

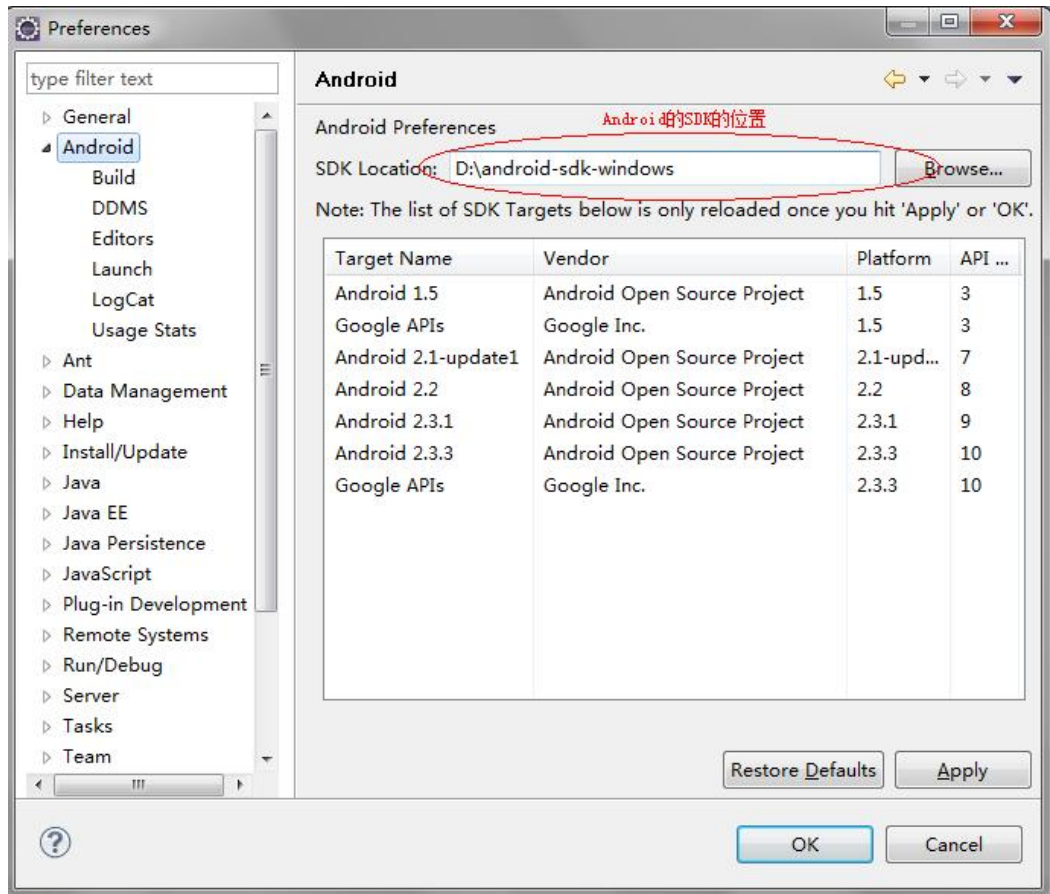
然后, 启动 Eclipse, 打开【Help】->【Install New Software】窗口, 安装 ADT 插件, 插件地址:

<https://dl-ssl.google.com/android/eclipse/>

注: 在国内更新的时候, 可能会出现更新失败, 这时候可以将 https 改成 http

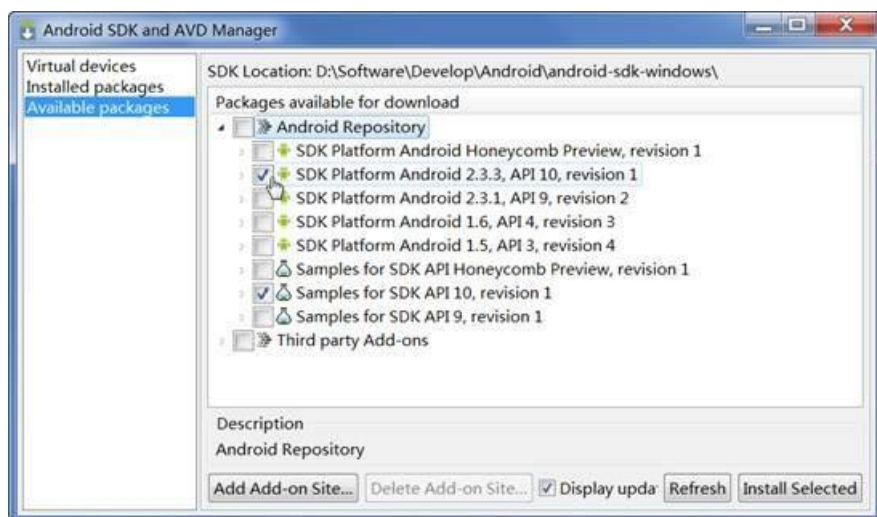
1.5 Eclipse 和 Android SDK 关联

在上一步, 已经将 Eclipse 的 ADT 插件集成, 同时下载了 Android SDK。这时, 要将 Eclipse 和 Android SDK 关联起来。打开【Window】->【References】菜单, 在 Eclipse 弹出的 References 窗体, 找到 Android 选项, 指定 Android SDK 的所在位置。如下图:

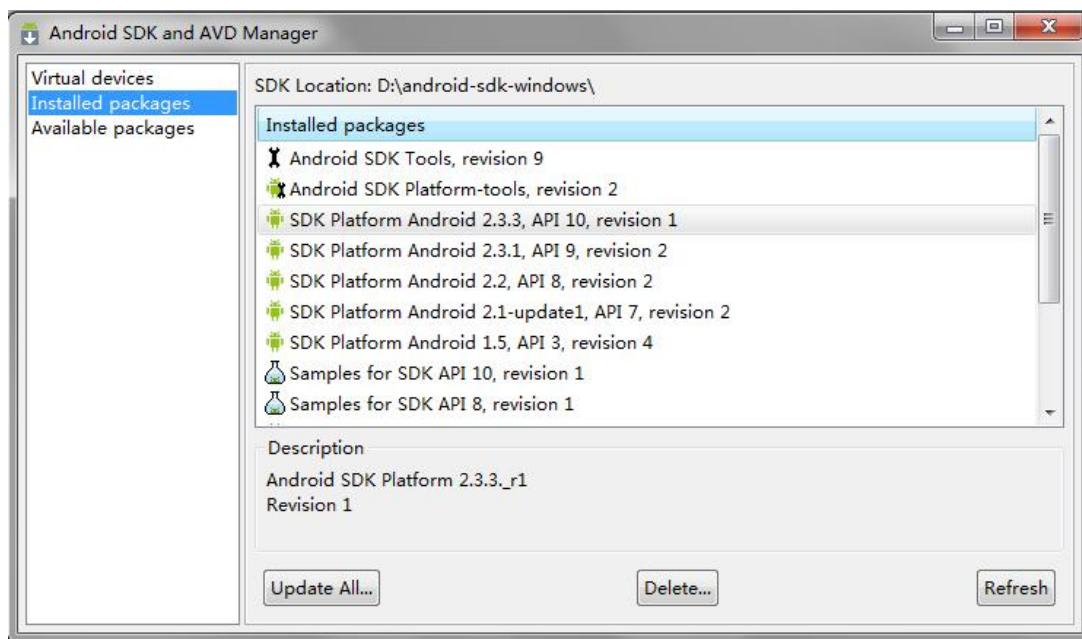


关联 Eclipse 和 Android SDK

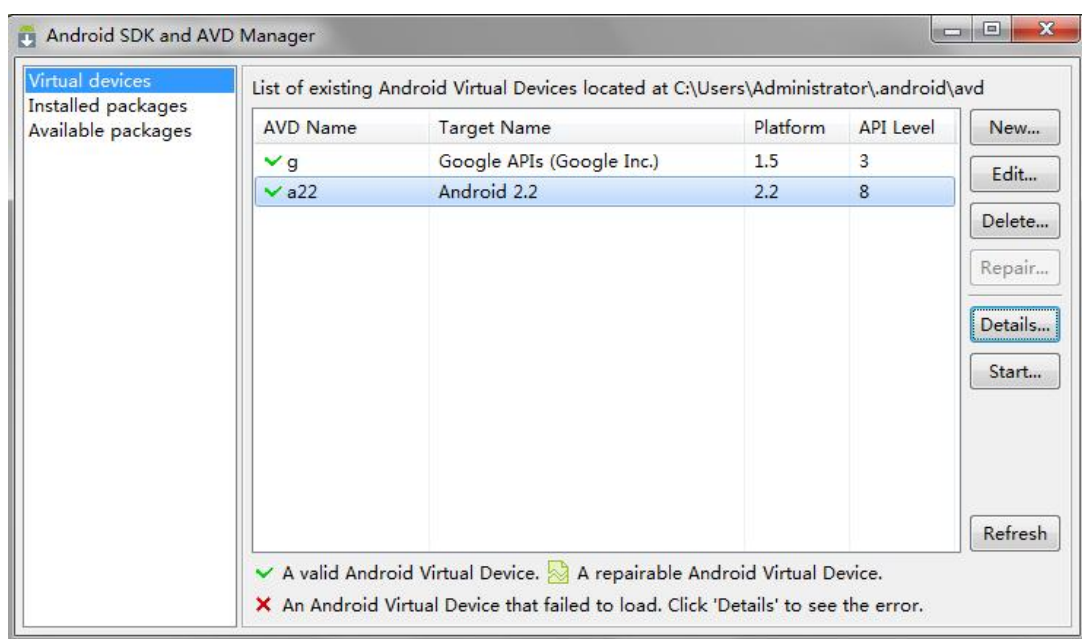
此时，Android 的开发环境配置完成。然后根据需要，更新 Android 的目标平台设备。通过【Window】->【Android SDK and AVD Manager】菜单，下载需要的平台并创建模拟器。建议下载 SDK Platform Android 2.2 API 8 和 SDK Platform Android 2.1-update API 7。



支持下载的Android 平台

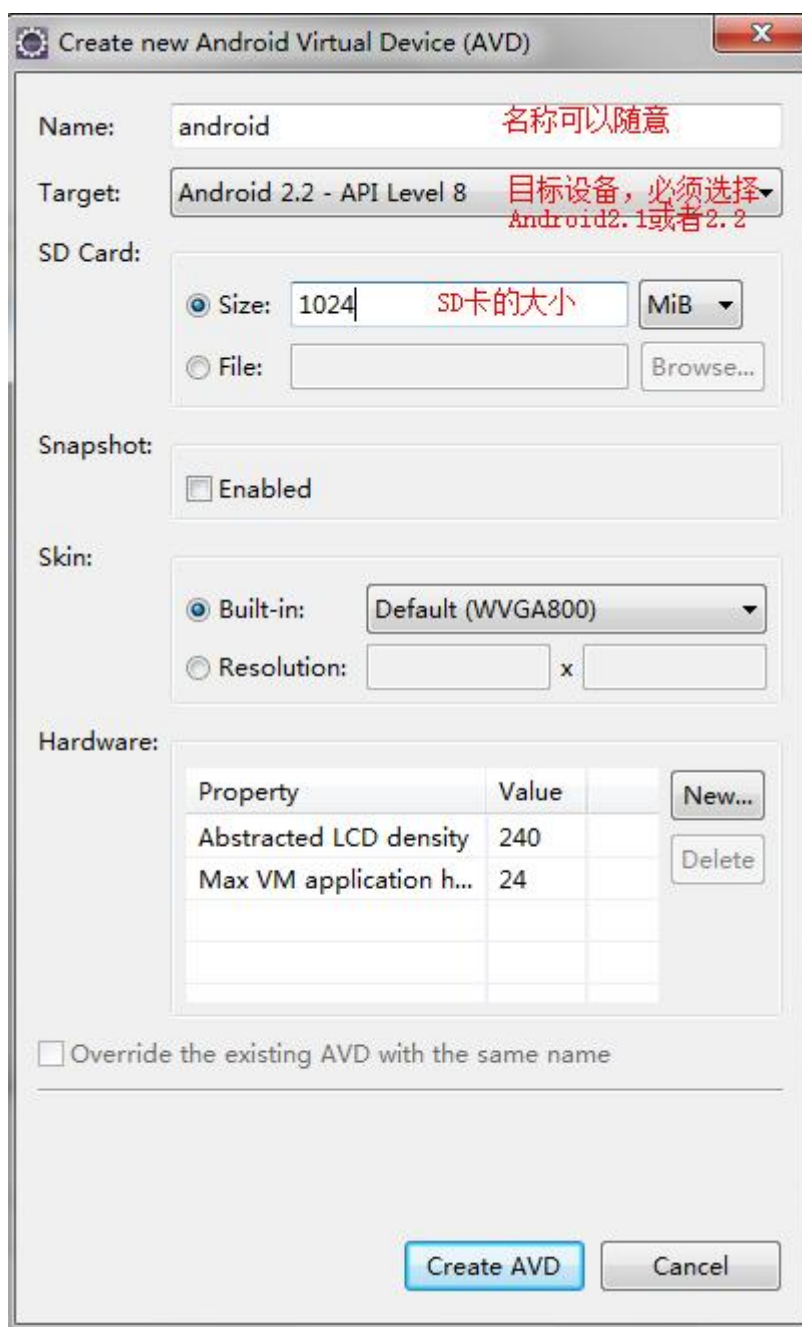


已经安装Android 平台



1.6 配置虚拟设备

下载完成对应 Android 平台，然后创建一个新的虚拟设备。如下图：



创建一个AVD

此时,Eclipse 下 Android 开发插件已经配置完成,我们可以开始配置 ArcGIS API For Android 开发插件。

1.7 安装 ArcGIS API For Android 插件

现在 Android 的开发环境已经具备，后面就需要安装 ArcGIS 开发相关的库和 Eclipse 插件了。在此之前，请确认您已经下载了 Android 2.1 或 2.1 的平台，因为这是 ArcGIS Android API 的系统需求¹。

Esri 提供了一个在线升级地址（<http://downloads.esri.com/software/arcgis/android>）来帮助用户安装 ArcGIS 的开发插件，在 Eclipse 中，还是通过【Help】->【Install New Software...】菜单就可以顺利地安装上这些组件：

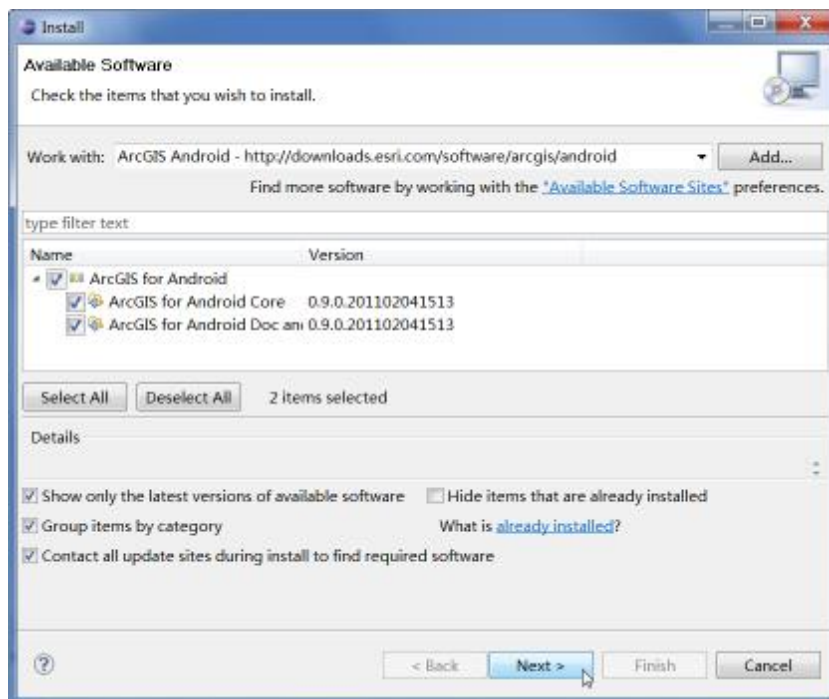


图 1 安装 ArcGIS 的开发插件

详细的安装步骤，请参考《Eclipse 下 ArcGIS for Android 插件安装指南》：

<http://support.esrichina-bj.cn/uploadfile/Mobile%20GIS%20APIs/InsallationStepsforAndroidBeta.pdf>

当 ArcGIS 开发插件安装完成后，在新建工程的选项中就可以看到【ArcGIS Project for Android】和【ArcGIS Samples for Android】的菜单，ArcGIS Android API 的开发环境就顺利配置完成了。

¹http://help.arcgis.com/en/arcgismobile/10.0/apis/android/help/#/System_requirements/01190000004000000/

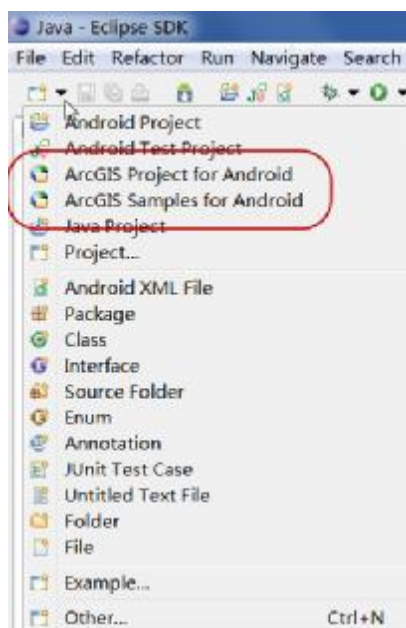


图 2 ArcGIS Android API 开发环境中新增的工程类型

2 新建示例工程

2.1 新建一个 HelloWorld 工程

我们安装的 ArcGIS API for Android Eclipse 开发插件中除了有 SDK，还内含了说明文档以及多个实例工程。刚开始接触 ArcGIS API for Android 时，我们建议您多研究和参考这些示例工程，并且结合我们的帮助文档逐步理解和熟悉使用该 API 进行开发的相关概念。

顺利配置完成 ArcGIS Android API 开发环境的之后，要新建一个示例工程是非常容易的下面我们新建和运行 Hello World 示例工程为例，向您介绍示例工程的使用方法。

在 Eclipse 中点击【File】->【New】->【ArcGIS Samples for Android】菜单，很快我们就能看到一个包含了所有例子的向导。然后我们选择 Map_View -> Hello World。

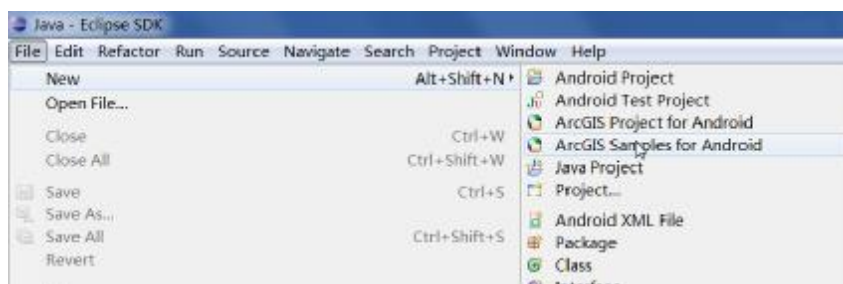


图 3 新建一个例子工程的菜单

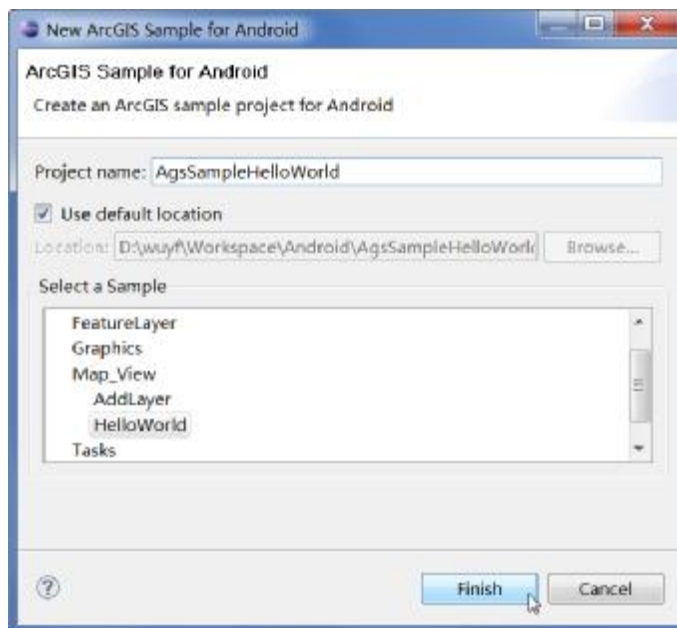


图 4 选择新建“Hello World”例子

新建工程结束后，在当前的工作空间中就会出现以图 4 中所示的工程名字——“AgsSampleHelloWorld”为名称的目录，在这个目录中存放了这个“Hello World”例子所有相关的资源。

2.2 运行这个 ArcGIS Android 程序

现在，让我们运行“AgsSampleHelloWorld”。

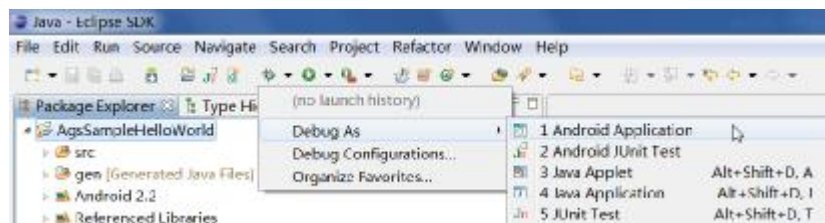


图 5 调试 Android 程序

和一般的 Java 程序类似，这里选择运行类型为 **Android Application**。一个 Android 程序想要运行，还需要一个模拟器或者真实的设备。这里我们以通过 ADT 插件中自带的模拟器为例来进行演示。

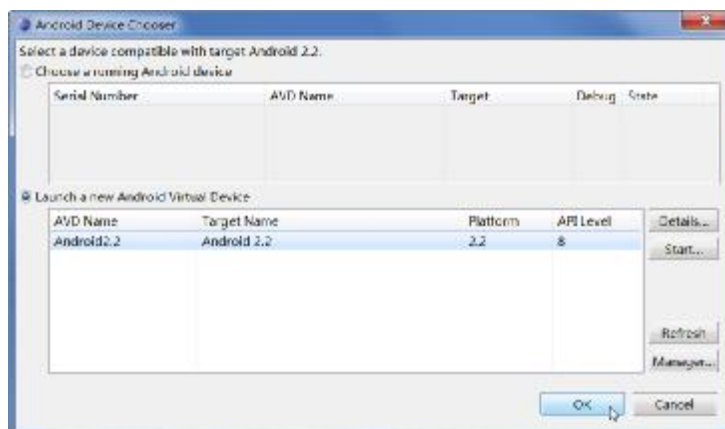


图 6 选择模拟器

第一次启动模拟器会花费比较多的时间，当成功启动后，你可以看到“AggSampleHelloWorld”工程运行的结果。

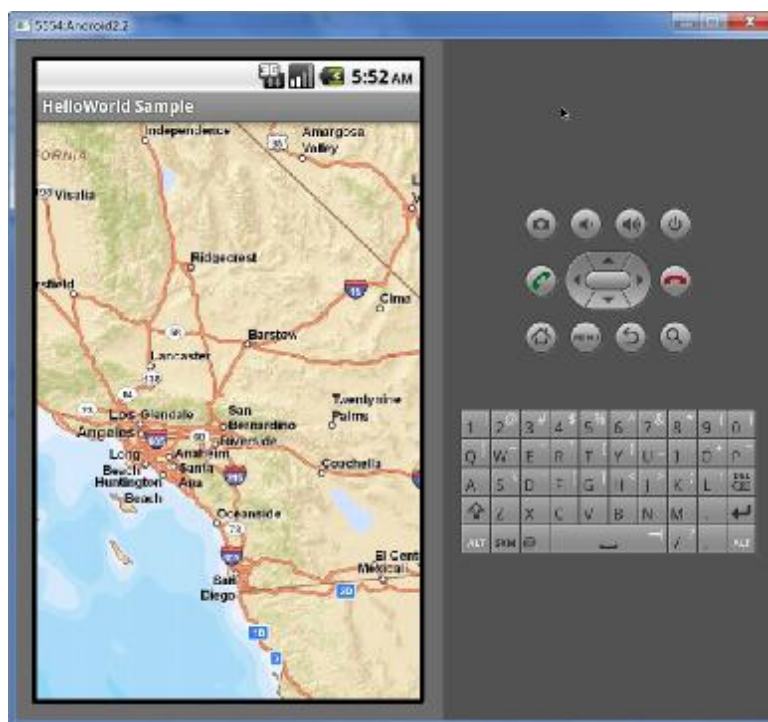


图 7 在 Android 模拟器上运行 Hello World 程序

3 开发手机地图

3.1 地图服务

MapView 类是 ArcGIS API For Android 里最主要的地图组件，它用来显示一组地图服务。MapView 类可以使用触摸进行地图导航，默认的地图导航有，放大，缩小，平移。

MapView 类直接继承于 Android 的 ViewGroup，因此，MapView 类集成了 Android ViewGroup 的所有方法和属性，应用起来非常简单，就像使用 Android 其他视图组件一样。

MapView 至少要有有一个 LayerView。MapView 是所有 LayerView 的父类。

LayerView 类是所有加载到 MapView 上的图层类的基类。在 MVC 架构中，这个类代表着 View，它用来处理这个架构功能相关的图片渲染。可以使用抽象类的任何一个子类。

例如：

- ArcGISTiledMapServiceLayer 用于显示缓存服务，瓦片地图。
- ArcGISDynamicMapServiceLayer 用于显示动态服务。
- ArcGISFeatureLayer 用于显示和操作要素图层。
- GraphicsLayer 用来在地图上显示一些额外的，或者临时的元素。

同时，可以扩展 LayerView 类，定义自己的服务类型。可以直接继承 LayerView，也可以继承 LayerView 的子类 DynamicLayer 和 TiledLayer。LayerView 类继承于 Android 的 View 类，所以它也可以使用 View 的所有属性和方法。

3.2 动态操作地图服务

刚才我们一开始就在布局文件中加入了地图服务，现在我们来尝试在程序运行的时候动态操作地图服务，比如先动态添加一个地图服务。

ArcGIS Android API 中有一个“AddLayer”例子可以用来作为参考，让我们先导入这个工程（位置：Map_View/AddLayer），运行一下这个例子：



图 8 例子 AddLayer 的运行效果

虽然看起来效果和上面的“Hello World”没什么太大的区别，只不过这里是两个地图服务叠加在了一起。但是，如果打开这个工程的源代码，你可以发现，在布局 XML 中只有一个切片地图服务，而在 Activity 运行时加入了另外一个地图服务：

```
public class AddLayer extends Activity {  
    private MapView map = null;  
    String dynamicMapURL = "http://sampleserver1.arcgisonline.com/ArcGIS/rest/  
        services/Specialty/ESRI_StateCityHighway_USA/MapServer";  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        this.map = (MapView) findViewById(R.id.map);  
  
        ArcGISDynamicMapServiceLayer dynamicLayer = new ArcGISDynamicMapServiceLayer(  
            this, this.dynamicMapURL);  
        this.map.addLayer(dynamicLayer);  
    }  
}
```

如果通过一个按钮来添加一个地图服务，那么可以把这个例子修改一下，首先，在 Activity 的布局中添加一个按钮：

```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="fill_parent" android:layout_height="fill_parent">  
  
    <com.esri.android.map.MapView android:id="@+id/map"  
        android:layout_width="fill_parent" android:layout_height="fill_parent"  
        initExtent="-1.3296373526814876E7 3930962.41823043  
        -1.2807176545789773E7 4201243.7502468005">  
        <com.esri.android.map.ags.ArcGISStyledMapServiceLayer  
            android:id="@+id/tiles1"  
            url="http://services.arcgisonline.com/ArcGIS/rest/services  
                /World_Street_Map/MapServer" />  
        </com.esri.android.map.MapView>  
  
    <Button android:id="@+id/buttonAdd"  
        android:layout_width="fill_parent" android:layout_height="wrap_content"  
        android:layout_alignParentBottom="true"  
        android:text="添加服务" />
```



```
</RelativeLayout>
```

当点击这个按钮的时候，我们希望能够在原来的切片地图服务之上再叠加一个动态服务，因此，还要给这个按钮添加一个点击的事件监听，一旦按钮被点击，则再加入一个地图服务。让我们看看修改过的 **Activity**:

```
public class AddLayer extends Activity {  
    String dynamicMapURL = "http://sampleserver1.arcgisonline.com/ArcGIS/rest  
        /services/Specialty/ESRI_StateCityHighway_USA/MapServer";  
    private MapView map = null;  
    private Button buttonAdd = null;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        this.map = (MapView) findViewById(R.id.map);  
        this.buttonAdd = (Button) findViewById(R.id.buttonAdd);  
  
        this.buttonAdd.setOnClickListener(new OnClickListener() {  
            public void onClick(View v) {  
                ArcGISDynamicMapServiceLayer dynamicLayer =  
                    new ArcGISDynamicMapServiceLayer(  
                        AddLayer.this, AddLayer.this.dynamicMapURL);  
                AddLayer.this.map.addLayer(dynamicLayer);  
            }  
        });  
    }  
}
```

这样，修改过的程序运行起来会出现一个“添加服务”的按钮，而点击这个按钮以后，程序就会在原有的切片服务之上再叠加一个动态地图服务，效果就像图 9 这样。



图 9 通过点击按钮来添加服务

再查看一下 MapView 的 API 我们可以知道，通过其它如 `removeLayer`、`reorderLayer` 等方法，我们可以对一个地图控件中的地图服务进行灵活的操作，以实现根据需要任意显示地图服务的目的。

3.3 导航与触屏操作

现在让我们看一下对 MapView 进行操作，第一个问题是地图导航。现在，我们新建一个 ArcGIS Android 工程，添加一个 MapView，并加入一个地图服务。

运行这个程序后，不出意外您马上就能看到这个地图服务。这时，如果我们想要拖动一下地图，用鼠标拖动一下就可以；如果我们想放大地图，在模拟器上双击也就可以实现。如果是在真机上，也可以用两个手指在屏幕上做一个放大和缩小的手势也可以来缩放地图，但是在模拟器上想要模拟这些动作比较困难。所以，这里我们再加上两个按钮，分别是“放大”、“缩小”：

```
<Button android:id="@+id/buttonZoomIn" android:text="放大"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"/>
<Button android:id="@+id/buttonZoomOut" android:text="缩小"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_alignParentRight="true"/>
```

```
public class Navigation extends Activity {
    private MapView map = null;
```

```
private Button buttonZoomIn = null;
private Button buttonZoomOut = null;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    this.map = (MapView) findViewById(R.id.map);
    this.buttonZoomIn = (Button) findViewById(R.id.buttonZoomIn);
    this.buttonZoomOut = (Button) findViewById(R.id.buttonZoomOut);

    this.buttonZoomIn.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Navigation.this.map.zoomin();
        }
    });

    this.buttonZoomOut.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Navigation.this.map.zoomout();
        }
    });
}
```

通过代码可以看到，调用 MapView 的 zoomin 和 zoomout 方法就可以实现缩放。这样在运行这个程序就会是图 10 这样，通过两个按钮和在地图上的拖动就可以任意缩放地图到你期望的位置了。



图 10 放大缩小地图

我们知道在触摸屏上还有其它的操作，比如“长按”。如果我们希望用户可以通过一个“长按”的操作来获取地图上一个点的坐标，然后再进行后续的标注等其它操作，应该怎么做呢？

在 ArcGIS Android API 中提供了一种事件监听类型为 `OnLongPressListener`，`MapView` 通过这个事件监听就可以对用户的长按事件做出反应，现在我们就给这个 `MapView` 加上一个长按监听，并在长按时在一个文本框中显示触摸处的地图坐标：

```
<TextView android:id="@+id/label"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:textColor="#ffffff" android:layout_centerHorizontal="true"
    android:layout_alignParentBottom="true" />
```

```
this.map.setOnLongPressListener(new OnLongPressListener() {
    public void onLongPress(float x, float y) {
        Point pt = Navigation.this.map.toMapPoint(x, y);
        Navigation.this.label.setText("X: " + pt.getX() + " Y: " + pt.getY());
    }
});
```

运行这个程序，然后用鼠标在模拟器上模拟一个长按的操作，您可以看到底部的文本框中显示了当前的坐标值，如图 11：



图 11 长按屏幕后显示坐标值

MapView 总共支持的事件类型如下, 通过这些监听我们可以在用户进行导航和触屏操作的时候进行自定义需要的相应处理方法:

事件监听	触发条件
OnLongPressListener	在 MapView 上长按时
OnMapExtentChangeListener	MapView 的地图范围改变时
OnPanListener	在 MapView 中拖动地图时
OnPinchListener	在 MapView 上两指缩放操作时
OnSingleTapListener	在 MapView 上单击时
OnStatusChangeListener	在 MapView 的状态改变, 比如被创建或初始化时
OnZoomListener	在 MapView 被缩放时
MapOnTouchListener	在 MapView 上按住并移动 (类似绘图操作) 时

4.2 客户端绘制

这部分主要介绍客户端的绘制。

GraphicLayer, 用于客户端绘制, 可以包含一个或者多个 Graphic。同时, map 可以包含多个 GraphicLayer。

Graphic, 用来显示要素, 它有三个属性, 分别是 Geometry, Symbol, Attribute。Geometry, 用来显示要素的地理位置信息。

Symbol，用来显示要素的样式。

Attribute，要素本身的属性信息。

GraphicLayer，还有一个 Renderer 属性。

下面，看一下客户端绘制的效果。ArcGIS API For Android的模板例子里有

一个DrawGraphicElements的例子，效果如下图：

3.4 客户端要素图层

ArcGIS Android 中提供了一个客户端的要素图层 GraphicsLayer，这是用于客户端要素绘制的图层。我们之前的例子基础上尝试在 MapView 中再添加一个 GraphicsLayer。首先在 main.xml 中添加了一个按钮“添加客户端要素”：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Button android:id="@+id/buttonAddGraphic" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="添加客户端要素" />

    <com.esri.android.map.MapView android:id="@+id/map"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
        <com.esri.android.map.ags.ArcGISStyledMapServiceLayer
            url="http://server.arcgisonline.com/ArcGIS/rest/services/
                /World_Imagery/MapServer" />

        <com.esri.android.map.GraphicsLayer android:id="@+id/gLayer" />
    </com.esri.android.map.MapView>
</LinearLayout>
```

我们希望在点击这个按钮的时候，程序可以在地图的中央添加一个客户端要素，也就是在 GraphicsLayer 上添加一个 Graphic。因此，我们在这个 Activity 创建的时候添加这样的一些代码：

```
gLayer = (GraphicsLayer) findViewById(R.id.gLayer);
gLayer.setRenderer(new SimpleRenderer(new SimpleMarkerSymbol(Color.RED,
    20, SimpleMarkerSymbol.STYLE.SQUARE)));

buttonAddGraphic = (Button) findViewById(R.id.buttonAddGraphic);
```



```
buttonAddGraphic.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Graphic g = new Graphic();
        g.setGeometry(AgsGraphicsLayer.this.map.getCenter());
        AgsGraphicsLayer.this.gLayer.addGraphic(g);
        AgsGraphicsLayer.this.gLayer.postInvalidate();
    }
});
```

请注意一下上面高亮的几行代码，上面对 **GraphicsLayer** 的渲染样式进行了定义，表示这个客户端要素图层上需要显示大小为 **20** 像素、方形的点要素；下面的代码通过按钮添加了客户端要素，并通知程序马上更新显示。这些代码运行的效果如下：



图 12 通过按钮添加客户端要素

3.5 通过交互绘制几何对象

客户端要素除了有显示业务数据的功能，同时也负责着和用户的交互。比如用户想要进行一个多边形查询，首先需要在客户端绘制一个多边形，然后再使用这个多边形进行一个空间查询。这样的一个功能我们可以从开发插件中的 **DrawGraphicElements** 例子来学习。下面先看一下这个示例工程运行的效果：

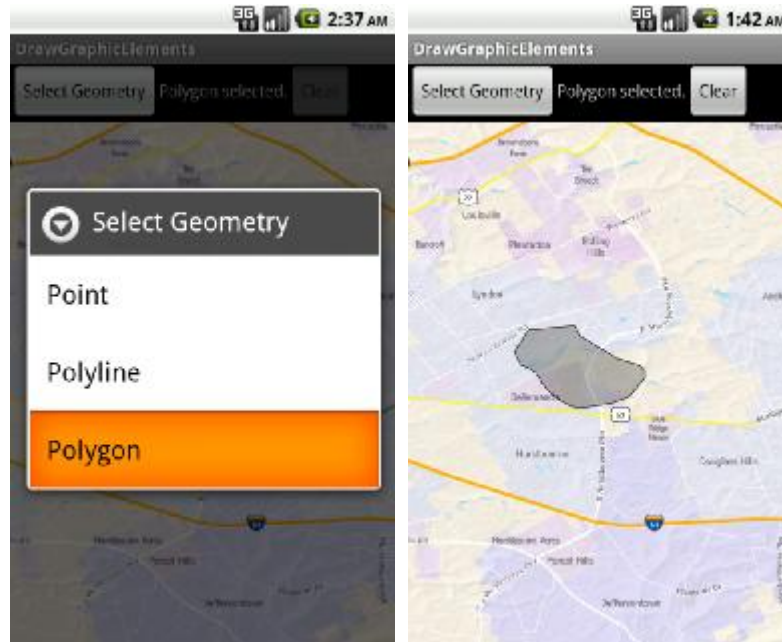


图 13 通过用户交互绘制一个多边形

这个例子的关键在于可以通过在屏幕上进行触屏操作，然后记录触屏的坐标构成一个几何对象，结束后将这个几何对象通过 **Graphic** 的形式绘制到屏幕上。下面让我们看看从点击上面的“**Select Geometry**”按钮之后，一步步都发生了什么。

首先，先会弹出了一个对话框，这在 **Java** 代码中非常简单：

```
geometryButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        showDialog(0);
    }
});
```

showDialog 是 **Activity** 的一个方法，这个方法会显示 **onCreateDialog** 方法定义的对话框，这个方法的内容如下（精简了一些提示信息等代码）：

```
protected Dialog onCreateDialog(int id) {
    return new AlertDialog.Builder(DrawGraphicElements.this)
        .setTitle("Select Geometry")
        .setItems(geometryTypes, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                graphicsLayer.clear();

                String geomType = geometryTypes[which];
                label.setText(geomType + " selected.");
                selectedGeometryIndex = which;

                if (geomType.equalsIgnoreCase("Polygon")) {
```

```
SimpleFillSymbol sfs = new SimpleFillSymbol(Color.GREEN);
sfs.setAlpha(60);
graphicsLayer.setRenderer(new SimpleRenderer(sfs));
myListener.setType("POLYGON");
} else if (geomType.equalsIgnoreCase("Polyline")) {
    graphicsLayer.setRenderer(new SimpleRenderer(new
SimpleLineSymbol(Color.RED, 5));
myListener.setType("POLYLINE");
} else if (geomType.equalsIgnoreCase("Point")) {
    graphicsLayer.setRenderer(new SimpleRenderer(new
SimpleMarkerSymbol(Color.BLUE, 15, STYLE.CIRCLE));
myListener.setType("POINT");
}
}
}).create();
}
```

这个对话框通过 `setItems` 方法显示出了 3 个可选项：“Point”、“Polyline”、“Polygon”，当点击不同的 `Item` 的时候，会触发事件监听，然后根据不同的 `Item` 设置不同的绘制类型（注意高亮的几行代码）。

这里还出现了一个没有出场过的 `myListener` 对象，上面通过给这个对象赋不同值达到改变绘制类型的目的，而具体的实现还需要看这个 `myListener` 的定义：

```
myListener = new MyTouchListener(DrawGraphicsElements.this, mapView);
mapView.setOnTouchListener(myListener);
```

```
class MyTouchListener extends MapOnTouchListener {
    MultiPath poly;
    String type = "";
    Point startPoint = null;

    public MyTouchListener(Context context, MapView view) {
        super(context, view);
    }

    public void setType(String geometryType) {
        this.type = geometryType;
    }

    public String getType() {
        return this.type;
    }
}
```

```
}

@Override
public boolean onSingleTap(MotionEvent e) {
    if (type.length() > 1 && type.equalsIgnoreCase("POINT")) {
        graphicsLayer.clear();
        Graphic graphic = new Graphic();
        graphic.setGeometry(mapView.toMapPoint(new Point(e.getX(), e.getY())));
        graphicsLayer.addGraphic(graphic);
        graphicsLayer.postInvalidate();
        clearButton.setEnabled(true);
        return true;
    }
    return false;
}

@Override
public boolean onDragPointerMove(MotionEvent from, MotionEvent to) {
    if (type.length() > 1 && (type.equalsIgnoreCase("POLYLINE") ||
        type.equalsIgnoreCase("POLYGON"))) {
        Point mapPt = mapView.toMapPoint(to.getX(), to.getY());

        if (startPoint == null) {
            graphicsLayer.clear();

            poly = type.equalsIgnoreCase("POLYLINE") ? new Polyline() : new Polygon();
            startPoint = mapView.toMapPoint(from.getX(), from.getY());
            poly.startPath((float) startPoint.getX(), (float) startPoint.getY());

            Graphic graphic = new Graphic();
            graphic.setGeometry(poly);

            graphicsLayer.addGraphic(graphic);
        }

        poly.lineTo((float) mapPt.getX(), (float) mapPt.getY());

        graphicsLayer.postInvalidate();
        return true;
    }
    return super.onDragPointerMove(from, to);
}
```

```
}

@Override
public boolean onDragPointerUp(MotionEvent from, MotionEvent to) {
    if (type.length() > 1 && (type.equalsIgnoreCase("POLYLINE") ||
                                type.equalsIgnoreCase("POLYGON"))) {
        if (type.equalsIgnoreCase("POLYGON")) {
            poly.lineTo((float) startPoint.getX(), (float) startPoint.getY());
        }
        startPoint = null;

        graphicsLayer.postInvalidate();
        clearButton.setEnabled(true);
        return true;
    }
    return false;
}
}
```

如果选择了“Polygon”，那么显然上面代码中的“type”值就是“Polygon”。这时，如果在屏幕上进行触屏操作，那么就会进入“onDragPointerMove”这个事件。在这个事件中，屏幕首先会捕捉触摸的第一个点，并放入“startPoint”这个对象，同时新建一个 Polygon 对象。再往后只要在屏幕上移动了一点，就会再进入这个事件中调用 lineTo 方法对这个 Polygon 对象进行绘制。最后，用户结束绘制，程序调用“onDragPointerUp”事件监听，这个 Graphic 对象构造完成。

4 查询和识别

仅仅在客户端进行交互是不够的，用户大多数的时候都需要和服务交互，而最基础的需求就是对服务进行查询，涉及到空间查询、属性查询、要素识别等。

4.1 空间查询和属性查询

首先，假设用户在屏幕上绘制一个一个点，然后想用这个点做一个空间查询。在这里，我们把这些工作都放在屏幕长按的事件监听中，当用户在屏幕上长按时，我们认为用户在屏幕上画了一个点，取到这个点以后我们就可以将其作为 Query 对象的几何对象

进行查询。在这里出现的 **Query** 和 **QueryTask** 对象同其他 ArcGIS Server Web API 中的相关概念和使用的方法非常类似。在下面高亮的代码中，我们可以看到对 **Query** 进行一定设置后，再执行 **QueryTask** 的 **execute** 方法，执行成功后就可以得到这次空间查询的结果：**FeatureSet**。

```
this.map.setOnLongPressListener(new OnLongPressListener() {  
    public void onLongPress(float x, float y) {  
        Point pt = AqsQuery.this.map.toMapPoint(x, y);  
  
        Query query = new Query("http://server.arcgisonline.com/ArcGIS/rest/services/  
            /Demographics/USA_1990-2000_Population_Change/MapServer/4");  
        query.setGeometry(pt);  
        query.setReturnGeometry(true);  
        query  
            .setSpatialRelationship(Query.SpatialRelationship.INTERSECTS);  
        QueryTask queryTask = new QueryTask(query);  
  
        try {  
            FeatureSet fs = queryTask.execute();  
  
            SimpleFillSymbol symbol = new SimpleFillSymbol(Color.BLACK);  
            AqsQuery.this.gLayer  
                .setRenderer(new SimpleRenderer(symbol));  
            AqsQuery.this.gLayer.clear();  
            AqsQuery.this.gLayer.addGraphics(fs.getGraphics());  
            AqsQuery.this.gLayer.postInvalidate();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
});
```

FeatureSet 对象中包含了所有结果 **Graphic** 的集合，这个集合甚至可以被直接批量添加到 **GraphicsLayer** 上去。这个例子运行的结果是这样的：

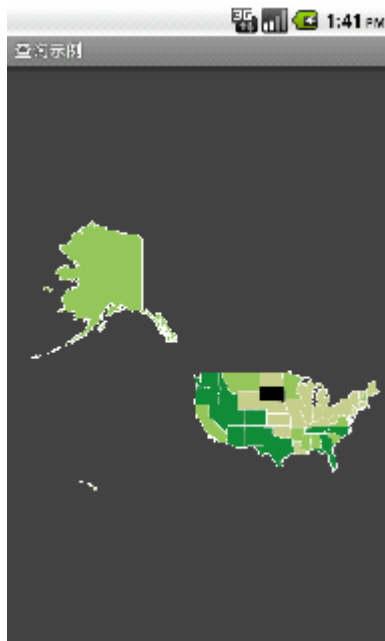


图 14 通过一个点进行空间查询

同样，使用 `QueryTask` 同样可以进行属性查询，比如我们稍微修改一下，只使用一个属性过滤：

```
Query query = new Query(queryUrl);  
query.setWhere("RATE_POP<0.2");  
query.setReturnGeometry(true);  
QueryTask queryTask = new QueryTask(query);
```

这样的查询可能就会有多个结果返回了，同样，把所有的结果绘制到 `GraphicsLayer` 上，得到的结果如下：

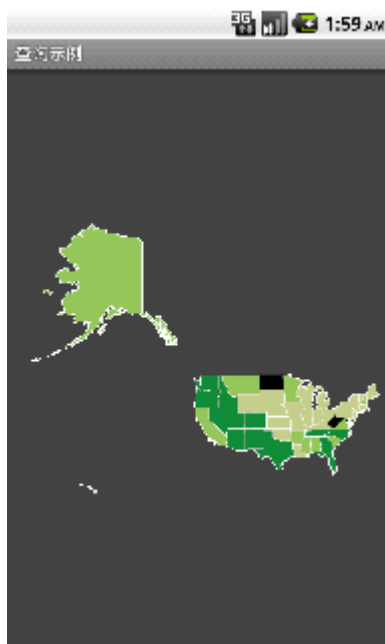


图 15 通过表达式进行属性查询

4.2 要素识别

要素识别就是 ArcGIS 中的 Identify，它和 Query 的区别在于可以执行多个图层的空间过滤，并可以指定一定的容差。要素识别的功能在 ArcGIS Android API 中的“HighlightFeatures”例子中被使用到了，我们先运行一下这个例子：

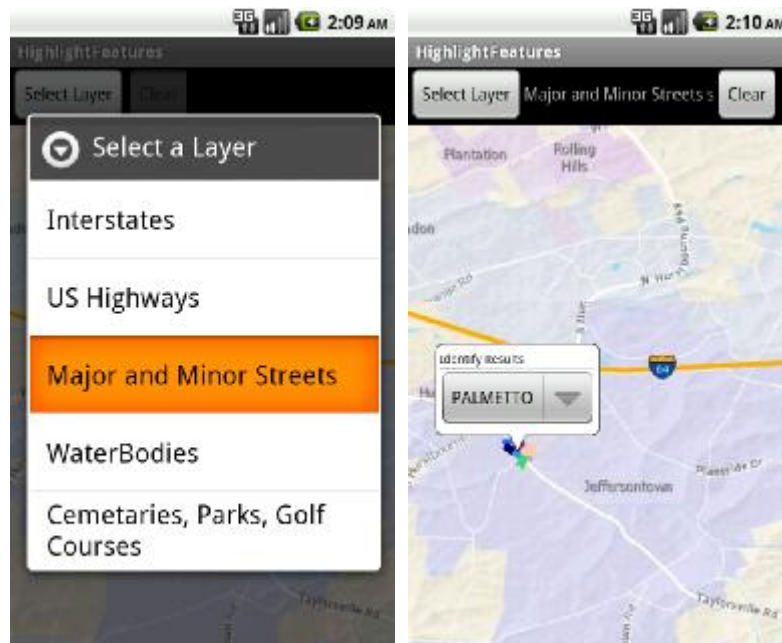


图 16 对地图服务进行要素识别

这样的功能实现上和 QueryTask 非常的类似，只不过调用的对象是 IdentifyTask，让我们来简单看一下代码的实现：

```
IdentifyParameters inputParameters = new IdentifyParameters();
inputParameters.setGeometry(pointClicked);
inputParameters.setUrl("http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/
    /PublicSafety/PublicSafetyBasemap/MapServer/identify");
inputParameters.setLayers(new int[] { layerIndexes[selectedLayerIndex] });
inputParameters.setMapExtent(mapView.getExtent());
inputParameters.setDPI(96);
inputParameters.setMapHeight(mapView.getHeight());
inputParameters.setMapWidth(mapView.getWidth());
inputParameters.setTolerance(10);

final IdentifyTask identifyAction = new IdentifyTask(inputParameters);
IdentifyResult[] results = identifyAction.execute();
```

从代码就可以看出来，Identify 功能的使用和其它所有的 ArcGIS Server API 几乎如出一辙。Identify 需要指定的最重要参数的就是包括的图层和容差，分别使用 setLayers 和 setTolerance 方法实现，这样，ArcGIS Android API 就知道需要

根据容差对查询的几何对象进行缓冲，然后在上述的若干图层中分别进行空间查询，最后把结果汇总后返回到客户端。

5 几何对象操作与地理处理

几何对象（Geometry）的操作和地理处理（GeoProcessing/GP）是在地图基础上实现更复杂 GIS 能力的必要功能，在这一章中让我们来看看在 ArcGIS Android API 中的如何对几何对象进行操作和如何使用 GP 服务。

5.1 几何对象的操作

ArcGIS Android API 对几何对象的操作和其它 ArcGIS 的 Web API 有很大区别，它并不是依赖 ArcGIS Sever 的 Geometry Service，而是在 API 库中本身就包含了对几何对象的定义和处理。如果我们相对一个几何对象进行 buffer 操作，这时不需要再指定一个 Geometry Service 的地址了，而是使用 ArcGIS Android API 提供的 GeometryEngine 类就可以实现这些类似的功能。

比如下面的代码实现了一个 buffer 的功能：

```
map.setOnLongPressListener(new OnLongPressListener() {  
    public void onLongPress(float x, float y) {  
        Point pt = map.toMapPoint(new Point(x, y));  
        Polygon pg = GeometryEngine.buffer(pt, map  
            .getSpatialReference(), 1000000, null); //null 表示采用地图单位  
        Graphic g = new Graphic();  
        g.setGeometry(pg);  
  
        AgsGeometryEngine.this.gLayer.addGraphic(g);  
        AgsGeometryEngine.this.gLayer.postInvalidate();  
    }  
});
```

这样，当我在地图上长按时，程序会捕捉到地图上的坐标，然后通过 GeometryEngine 的静态方法 buffer 对这个点进行一个缓冲操作，最后把结果绘制到 GraphicsLayer 上去，这个程序执行的效果是这个样子的：



图 17 使用 GeometryEngine 对点进行缓冲的结果

5.2 地理处理服务

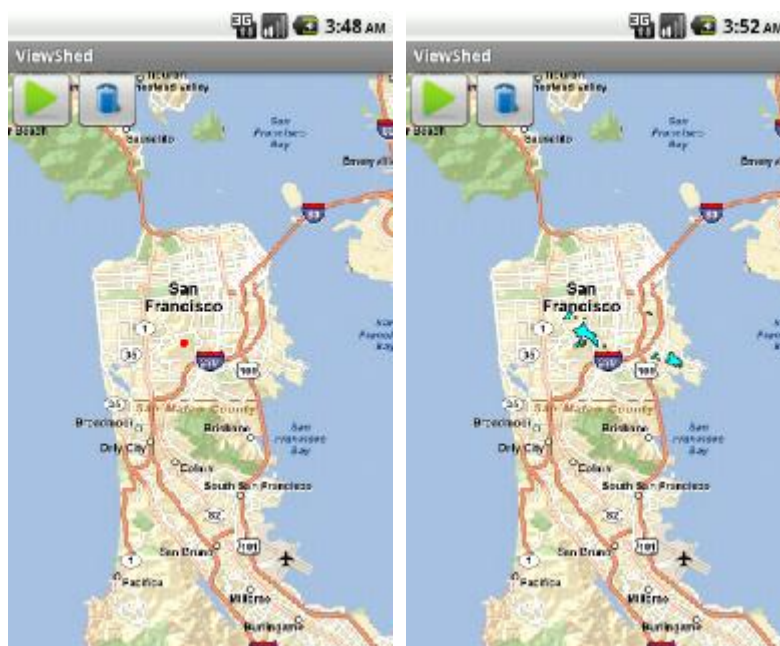


图 18 调用 GP 服务的 ViewShed 例子

ArcGIS Android API 中 ViewShed 示例程序演示了 GP 服务的调用。这个例子让用户在屏幕上画一个点，再通过左上角那个执行的按钮发送一个调用 GP 服务的请求，当这个 GP 服务被调用成功后，返回给客户端的是一些多边形，这些多边形表示的是如果有人站在刚才用户画点的位置所能看到的区域。

现在，如果你打开这个例子的源代码，你应该可以很容易地从左上角那个“go”按钮的事件监听中找到这个 GP 服务的调用方法：

```
GPFeatureRecordSetLayer gpf = new GPFeatureRecordSetLayer(
    "Input_Observation_Point");
gpf.setSpatialReference(map.getSpatialReference());
gpf.setGeometryType("esriGeometryPoint");
Graphic f = new Graphic();
f.setGeometry(mappoint);
gpf.addGraphic(f);

GPLinearUnit gpl = new GPLinearUnit("Viewshed_Distance");
gpl.setUnits("esriMeters");
gpl.setDistance(8046.72);

params = new ArrayList<GPParameter>();
params.add(gpf);
params.add(gpl);

try {
    dialog = ProgressDialog.show(Viewshed.this, "",
        "Loading. Please wait...", true, true);
    new ViewshedQuery().execute(params);
    cancelViewShed = new Timer();
    cancelViewShed.schedule(new TimerTask() {
        @Override
        public void run() {
            uiHandler.sendMessage(CANCEL_LOADING_WINDOW);
        }
    }, 60000);
} catch (Exception e) {
    e.printStackTrace();
}
```

在上面的代码中，主要做了两件事情：第一件事情是准备 GP 服务执行所需要的参数 `ArrayList<GPParameter>`；另一件事情就是调用 GP 服务。在这个例子里，对 GP 服务的调用还进行了一个封装，而事实上核心的代码可以从下面代码中的高亮部分看到（代码略有修改）：

```
class ViewshedQuery extends
    AsyncTask<ArrayList<GPParameter>, Void, GPParameter[]> {
    GPParameter[] outParams = null;

    @Override
```

```
protected void onPostExecute(GPParameter[] result) {  
    ...  
}  
  
@Override  
protected GPParameter[] doInBackground(  
    ArrayList<GPParameter>... params1) {  
  
    gp = new GeoProcessor("http://sampleserver1.arcgisonline.com/ArcGIS  
        /rest/services/Elevation/ESRI_Elevation_World/GPServer/Viewshed");  
    gp.setOutputSR(map.getSpatialReference().getID());  
    try {  
        GPResultResource gpr = gp.execute(params1[0]);  
        outParams = gpr.getOutputParameters();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
  
    return outParams;  
}  
}
```

很显然，这个 GP 服务是一个同步执行的服务，因此在这里调用的是 GeoProcessor 对象的 execute 方法。如果是一个异步执行的 GP 服务，需要执行 GeoProcessor 的 submitJob 方法。

6 要素编辑

从 ArcGIS 10 开始，地图服务有了一个新的 Capability——“Feature Access”，所有的要素编辑都是通过这个接口实现的。同时，在各种客户端中出现了一种新的图层名为“Feature Layer”，它可以对应到一个地图服务的某个要素图层，因此，要进行要素编辑，让我们先熟悉一下 ArcGIS Android API 中的 Feature Layer。

6.1 Feature Layer

ArcGIS Android API 提供了一个选择要素的示例程序：SelectFeatures。这个例子主要演示了 Feature Layer 的用法，运行这个例子的效果大概是这个样子：

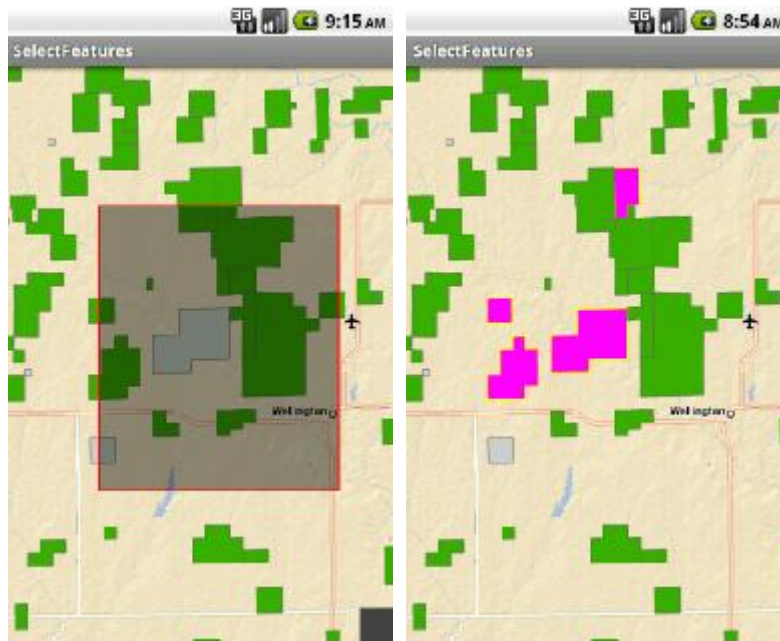


图 19 使用 Feature Layer 选择要素

当用户画定了一个范围，程序就会选中 **Feature Layer** 在这个范围中符合条件的要素，并把这些要素用粉红色高亮（其它地方显示其它颜色的要素是因为下面还叠加的一个动态地图服务，它们并不是在 **Feature Layer** 上显示的）。

从这个例子的源代码中，我们可以很容易找到最关键的几行代码。在这几行代码中，程序定义了一个 **Query** 对象用以设置过滤条件，然后简单地通过调用 **ArcGISFeatureLayer** 对象的 **selectFeature** 方法就实现了对要素的选择。

```
fLayer.clearSelection();
Query q = new Query();
q.setWhere("PROD_GAS=' Yes' ");
q.setReturnGeometry(true);
q.setInSpatialReference(map.getSpatialReference());
q.setGeometry(g.getGeometry());
q.setSpatialRelationship(SpatialRelationship.INTERSECTS);

fLayer.selectFeatures(q, SELECTION_METHOD.NEW, callback);
```

而在本质上，这里调用的 **selectFeature** 方法背后访问的是 **ArcGISFeatureLayer** 所对应的要素图层，通过 **Query** 对象限制的条件进行查询操作，再把查询的结果 (**FeatureSet**) 返回到客户端，并添加到 **ArcGISFeatureLayer** 上去。

ArcGISFeatureLayer 本质上是一个和一个服务器端的要素图层遥相呼应的 **GraphicsLayer**:

```
Options o = new Options();
o.mode = MODE.ONDEMAND;
o.outFields = new String[] { "FIELD_KID", "APPROXACRE", "FIELD_NAME" };
```

```
fLayer = new ArcGISFeatureLayer(this,
    "http://sampleserver3.arcgisonline.com/ArcGIS/rest
    /services/Petroleum/KSPetro/MapServer/1", o);
```

需要您注意的是，这里 `ArcGISFeatureLayer` 对应的 URL 地址是 `MapServer` 中的一个图层，因此在这里这个 `Feature Layer` 虽然可以选择，但是做不了编辑。要想进行编辑，`ArcGISFeatureLayer` 加载的 URL 地址就必须是一个 `FeatureServer` 中的图层。

6.2 属性编辑

`ArcGIS Android API` 中 `AttributeEditor` 示例程序演示了属性编辑的方法。当您点击一个要素，程序就会弹出一个编辑要素属性的对话框供您对现有属性值进行修改：

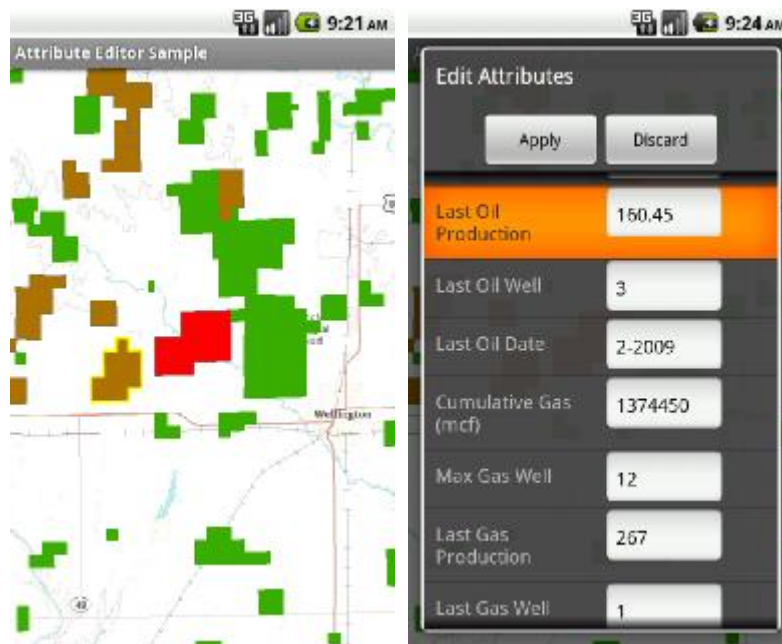


图 20 编辑要素的属性

点击一个要素，从服务查询到该要素的属性信息在前面的内容中已经介绍过了，如何将查询到的属性显示在一个对话框中也是比较容易的（例子中将如何根据属性值来构建对话框这个过程进行了封装），我们需要关心的是一旦修改了某个属性值，然后点击对话框中的“**Apply**”按钮后，程序如何把更改保存到服务器。和其他 `ArcGIS` 的 `Web API` 类似，提交属性编辑的方法逻辑上就是新建一个 `Graphic` 对象，并将修改过的属性值赋予这个 `Graphic`（别忘了 `ObjectID`，这是标识编辑哪个要素的必需属性），然后调用 `ArcGISFeatureLayer` 的 `applyEdits` 方法进行提交。`Feature Layer` 已经封装了 `Feature Server` 服务中的 `applyEdits` 接口，保存编辑到服务器只需要简单的调用这个方法就可以了。

```
Graphic newGraphic = new Graphic();
...//一系列赋值
```

```
newGraphic.setAttributeValue(featureLayer.getObjectIdField(),
    listAdapter.featureSet.getGraphics()[0]
    .getAttributeValue(featureLayer.getObjectIdField()));
featureLayer.applyEdits(null, null, new Graphic[] { newGraphic },
    createEditCallbackListener(updateMapLayer));
```

6.3 几何编辑

几何编辑从原理上和属性编辑是一样的，这里就如何在客户端更改要素的几何属性，并通过 **Feature Layer** 保存到服务器做新建一个示例程序。

要进行几何编辑分为两个步骤，其一：在客户端修改几何对象的节点坐标；其二：将更新过的几何对象提交到服务器。

让我们首先考虑第一步，如何修改一个客户端的几何对象。我们先新建一个工程名为“**AgsEditFeatureLayer**”，在这个工程默认的 **Activity** 中加入 **MapView** 和必要的图层：

```
<com.esri.android.map.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    initExtent="-10868502.895856911 4470034.144641369
                -10837928.084542884 4492965.25312689">

    <com.esri.android.map.ags.ArcGIS tiledMapServiceLayer
        url="http://services.arcgisonline.com/ArcGIS/rest
            /services/World_Topo_Map/MapServer" />

    <com.esri.android.map.ags.ArcGISDynamicMapServiceLayer
        url="http://sampleserver3.arcgisonline.com/ArcGIS/rest
            /services/Petroleum/KSFIELDS/MapServer" />

    <com.esri.android.map.ags.ArcGISFeatureLayer android:id="@+id/fLayer"
        url="http://sampleserver3.arcgisonline.com/ArcGIS/rest
            /services/Petroleum/KSFIELDS/FeatureServer/0"
        mode="selection" />

    <com.esri.android.map.GraphicsLayer android:id="@+id/gLayer"/>
</com.esri.android.map.MapView>
```

我们希望用户可以在屏幕上通过长按操作来选择一个需要编辑的几何对象。因此，需要在长按的事件中进行实现“选择要素”这个操作：

```
public class AgsEditFeatureLayer extends Activity {
    MapView map = null;
    ArcGISFeatureLayer fLayer = null;
    GraphicsLayer gLayer = null;
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    map = (MapView) findViewById(R.id.map);
    fLayer = (ArcGISFeatureLayer) findViewById(R.id.fLayer);
    gLayer = (GraphicsLayer) findViewById(R.id.gLayer);

    SimpleFillSymbol fSymbol = new SimpleFillSymbol(Color.YELLOW);
    fSymbol.setOutline(new SimpleLineSymbol(Color.RED, 2));
    fLayer.setSelectionSymbol(fSymbol);

    SimpleFillSymbol gSymbol = new SimpleFillSymbol(Color.BLACK);
    gSymbol.setOutline(new SimpleLineSymbol(Color.RED, 2));
    gLayer.setRenderer(new SimpleRenderer(gSymbol));

    map.setOnLongPressListener(new OnLongPressListener() {
        public void onLongPress(float x, float y) {
            Point pt = map.toMapPoint(new Point(x, y));
            Query q = new Query();
            q.setReturnGeometry(true);
            q.setGeometry(pt);
            q.setSpatialReference(map.getSpatialReference());
            q.setSpatialRelationship(SpatialRelationship.INTERSECTS);

            fLayer.selectFeatures(q, SelectionMethod.NEW, null);
        }
    });
}
```

当执行了 `selectFeatures` 这个方法之后，实际上程序向 ArcGIS Server 发送了一个空间查询，得到结果后将查询的结果绘制到 `fLayer` 这个 `ArcGISFeatureLayer` 上，实际上也就是在这个 `GraphicsLayer` 上添加了一个 `Graphic`（图 19 中的 ）。

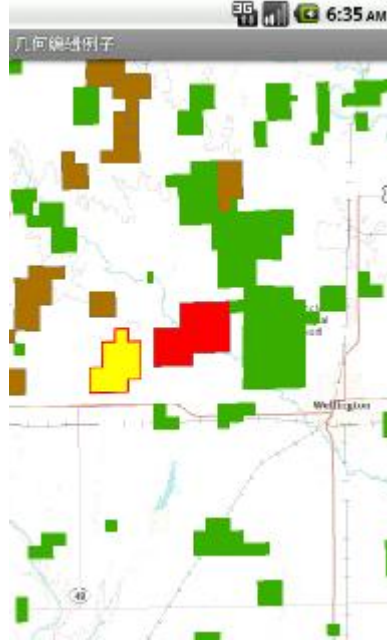


图 21 在 Feature Layer 上选择一个要素

下一步,我们需要做的是可以让用户通过触屏操作来改变这个 **Graphic** 的几何形状。我们可以把这个过程简化一些,比如只允许用户改变已经存在的节点。改变时的几何形状我们绘制在另外一个 **GraphicsLayer** 上,名为 **gLayer**。所以上面的代码我们先稍作修改,在选中要素的同时把这个要素复制到 **gLayer** 上,你可以把这段修改过的选择要素代码和上面高亮的没有修改的代码进行比较:

```
fLayer.selectFeatures(q, SELECTION_METHOD.NEW, new CallbackListener<FeatureSet>() {
    public void onCallback(FeatureSet fs) {
        gLayer.clear();
        gLayer.addGraphics(fs.getGraphics());
        gLayer.postInvalidate();
    }

    public void onError(Throwable t) {
        t.printStackTrace();
    }
});
```

为了对几何对象进行编辑,我们继承 **MapOnTouchListener** 定义了一个新的监听,并在这个监听中封装了编辑几何对象的逻辑,然后将这个监听绑定到 **MapView** 对象上:

```
map.setOnTouchListener(new MyMapOnTouchListener(this, map, fLayer, gLayer));
```

```
public class MyMapOnTouchListener extends MapOnTouchListener {
    private Context context = null;
    private MapView map = null;
```

```
private ArcGISFeatureLayer fLayer = null;
private GraphicsLayer gLayer = null;

private Graphic g = null;
private int editIndex = -1;

public MyMapOnTouchListener(Context context, MapView map,
    ArcGISFeatureLayer fLayer, GraphicsLayer gLayer) {
    super(context, map);

    this.context = context;
    this.map = map;
    this.fLayer = fLayer;
    this.gLayer = gLayer;
}

@Override
public boolean onDragPointerMove(MotionEvent from, MotionEvent to) {
    Graphic[] gs = gLayer.getGraphics();
    if(gs==null || gs.length==0){
        return false;
    }

    Graphic g = gs[0];
    Polygon pg = (Polygon)g.getGeometry();

    if (editIndex<0) { //此时需要获取编辑的节点序号
        Point ptClick = map.toMapPoint(from.getX(), from.getY());
        Proximity2DResult pr = GeometryEngine.getNearestVertex(pg, ptClick);
        editIndex = pr.getVertexIndex();
    }

    if(g!=null && editIndex>=0){
        Point ptTo = map.toMapPoint(to.getX(), to.getY());
        pg.setPoint(editIndex, ptTo); //改变指定节点的坐标
    }

    gLayer.postInvalidate();
    return true;
}
```



```

    }

    @Override
    public boolean onDragPointerUp(MotionEvent from, MotionEvent to) {
        editIndex = -1;
        gLayer.postInvalidate();
        return true;
    }
}

```

这样，当用户在屏幕上拖动的时候，刚才被选中的多边形就可以被编辑。比如在图 22 中我们就编辑了 3 个节点的位置。



图 22 通过屏幕事件编辑几何形状

这个时候在客户端的工作就完成了，现在我们需要进行第二步，就是将在客户端所做的编辑提交到 ArcGIS Server 上去。这里，我们加入一个按钮，当点击这个按钮的时候程序将执行下面的提交操作：

```

buttonApply.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        fLayer.applyEdits(null, null, gLayer.getGraphics(),
            new CallbackListener<FeatureEditResult[][]>() {
                public void onCallback(FeatureEditResult[][] results) {
                    System.out.println("Updated");
                }

                public void onError(Throwable t) {

```

```
t.printStackTrace();
    }
});
}
});
```

如果您留意一下 Eclipse 中 LogCat 中的日志，就会看到这样的记录，表示编辑提交成功：

```
02-21 14:18:02.358: DEBUG/ArcGIS.ThreadPool (736): Updated Features: 1
```

当然，您也可以在 `applyEdits` 方法的 `CallbackListener` 中加入更进一步的处理。只要提交成功，再刷新一下动态底图，就能看到更新后的结果：



图 23 几何编辑后的动态服务效果

7 在线资源

ArcGIS for Android (Resource Center) :

<http://help.arcgis.com/en/arcgismobile/10.0/apis/android/help/>

ArcGIS Android API 参考:

<http://help.arcgis.com/en/arcgismobile/10.0/apis/android/api/index.html>