Configuring and Using Map Views

The MapView class is a View that displays the actual map; it includes several options for deciding how the map is displayed.

By default, the Map View will show the standard street map, as shown in Figure 7-5. In addition, you can choose to display a satellite view, StreetView, and expected traffi c, as shown in the code snippet below:

```
mapView.setSatellite(true);
mapView.setStreetView(true);
mapView.setTraffic(true);
```

You can also query the Map View to fi nd the current and maximum available zoom level, as well as the center point and currently visible longitude and latitude span (in decimal degrees). The latter (shown below) is particularly useful for performing geographically limited Geocoder lookups:

```
GeoPoint center = mapView.getMapCenter();
int latSpan = mapView.getLatitudeSpan();
int longSpan = mapView.getLongitudeSpan();
```

You can also optionally display the standard map zoom controls. The following code snippet shows how to get a reference to the Zoom Control View and pin it to a screen location. The Boolean parameter lets you assign focus to the controls once they're added.

```
int y = 10;
int x = 10;
MapView.LayoutParams lp;

lp = new MapView.LayoutParams(MapView.LayoutParams.WRAP_CONTENT, MapView.LayoutParams.WRAP_CONTENT,
x, y, MapView.LayoutParams.TOP_LEFT);

View zoomControls = mapView.getZoomControls();
mapView.addView(zoomControls, lp);
mapView.displayZoomControls(true);
```

The technique used to pin the zoom controls to the MapView is covered in more detail later in this chapter.

Using the Map Controller

You use the Map Controller to pan and zoom a MapView. You can get a reference to a MapView's controller using getController, as shown in the following code snippet:

```
MapController mapController = myMapView.getController();
```

Map locations in the Android mapping classes are represented by GeoPoint objects, which contain latitude and longitude measured in microdegrees (i.e., degrees multiplied by 1E6 [or 1,000,000]).

Before you can use the latitude and longitude values stored in the Location objects used by the locationbased services, you'll need to convert them to microdegrees and store them as GeoPoints, as shown in the following code snippet:

```
Double lat = 37.422006*1E6;
Double lng = -122.084095*1E6;
GeoPoint point = new GeoPoint(lat.intValue(), lng.intValue());
```

Re-center and zoom the MapView using the setCenter and setZoom methods available on the MapView's MapController, as shown in the snippet below:

```
mapController.setCenter(point);
mapController.setZoom(1);
```

When using setZoom, 1 represents the widest (or furthest away) zoom and 21 the tightest (nearest) view.

The actual zoom level available for a specifi c location depends on the resolution of Google's maps and imagery for that area. You can also use zoomIn and zoomOut to change the zoom level by one step.
The setCenter method will "jump" to a new location; to show a smooth transition, use animateTo as shown in the code below:

```
mapController.animateTo(point);
```

Mapping "Where Am I?"

In the following code example, the "Where Am I?" project is extended again. This time you'll add mapping functionality by transforming it into a Map Activity. As the device location changes, the map will automatically re-center on the new position.

1. Start by adding the uses-permission tag for Internet access to the application manifest. Also import the Android maps library within the application tag.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.paad.whereami">
<application
android:icon="@drawable/icon">
<activity
android:name=".WhereAmI"
android:label="@string/app_name">
<intent-fi lter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<uses-library android:name="com.google.android.maps"/>
</application>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
</manifest>
```

2. Change the inheritance of WhereAmI to descend from MapActivity instead of Activity. You'll also need to include an override for the isRouteDisplayed method. Because this Activity won't show routing directions, you can return false.

```
public class WhereAmI extends MapActivity {
@Override
protected boolean isRouteDisplayed() {
return false;
}
[ ... existing Activity code ... ]
        }
```

3. Modify the main.xml layout resource to include a MapView using the fully qualifi ed class name. Be sure to include an android:apikey attribute within the com.android.MapView node. If you have an Android maps API key, use it here.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fi ll_parent"
android:layout_height="fi ll_parent">
<TextView
android:id="@+id/myLocationText"
android:layout_width="fi ll_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
<com.google.android.maps.MapView
android:id="@+id/myMapView"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:enabled="true"
android:clickable="true"
android:apiKey="myMapKey"
/>
</LinearLayout>
```

4. Running the application now should display the original geolocation text, with a MapView beneath it, as shown in Figure 7-6.

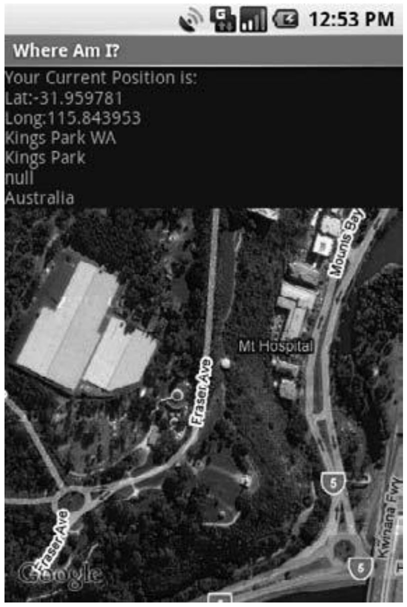
Where Am I?
Your Current Position is:
Lat:-31.959781
Long:115.843953
Kings Park WA
Kings Park
null
Australia

Figure 7-6

5. Confi gure the Map View and store a reference to its MapController as an instance variable. Set up the Map View display options to show the satellite and StreetView and zoom in for a closer look.

```
MapController mapController;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
// Get a reference to the MapView
MapView myMapView = (MapView)findViewById(R.id.myMapView);
// Get the Map View's controller
mapController = myMapView.getController();
// Configure the map display options
myMapView.setSatellite(true);
myMapView.setStreetView(true);
myMapView.displayZoomControls(false);
// Zoom in
mapController.setZoom(17);
LocationManager locationManager;
String context = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(context);
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
String provider = locationManager.getBestProvider(criteria, true);
Location location = locationManager.getLastKnownLocation(provider);
updateWithNewLocation(location);
locationManager.requestLocationUpdates(provider, 2000, 10, locationListener);
}
```

6. The fi nal step is to modify the updateWithNewLocation method to re-center the map to the current location using the Map Controller.

```
private void updateWithNewLocation(Location location) {
String latLongString;
TextView myLocationText;
        myLocationText = (TextView)fi ndViewById(R.id.myLocationText);
        String addressString = “No address found”;
        if (location != null) {
        // Update the map location.
        Double geoLat = location.getLatitude()*1E6;
        Double geoLng = location.getLongitude()*1E6;
        GeoPoint point = new GeoPoint(geoLat.intValue(),
        geoLng.intValue());
        mapController.animateTo(point);
        double lat = location.getLatitude();
        double lng = location.getLongitude();
        latLongString = “Lat:” + lat + “\nLong:” + lng;
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        Geocoder gc = new Geocoder(this, Locale.getDefault());
        try {
        List<Address> addresses = gc.getFromLocation(latitude, longitude, 1);
        StringBuilder sb = new StringBuilder();
        if (addresses.size() > 0) {
        Address address = addresses.get(0);
        for (int i = 0; i < address.getMaxAddressLineIndex(); i++)
        sb.append(address.getAddressLine(i)).append(“\n”);
        sb.append(address.getLocality()).append(“\n”);
        sb.append(address.getPostalCode()).append(“\n”);
        sb.append(address.getCountryName());
        }
        addressString = sb.toString();
        } catch (IOException e) {}
        } else {
        latLongString = “No location found”;
        }
        myLocationText.setText(“Your Current Position is:\n” + latLongString + “\n” + addressString);
                }
```