Handling Map Tap Events

To handle map taps (user clicks), override the onTap event handler within the Overlay extension class. The onTap handler receives two parameters:

❑ A GeoPoint that contains the latitude/longitude of the map location tapped

❑ The MapView that was tapped to trigger this event

When overriding onTap, the method should return true if it has handled a particular tap and false to let another overlay handle it, as shown in the skeleton code below:

```
@Override
public boolean onTap(GeoPoint point, MapView mapView) {
// Perform hit test to see if this overlay is handling the click
if ([... perform hit test ... ]) {
[ ... execute on tap functionality ... ]
return true;
}
// If not handled return false
return false;

}
```

Adding and Removing Overlays

Each MapView contains a list of Overlays currently displayed. You can get a reference to this list by calling getOverlays, as shown in the snippet below:

```
List<Overlay> overlays = mapView.getOverlays();
```

Adding and removing items from the list is thread safe and synchronized, so you can modify and query the list safely. Iterating over the list should still be done within a synchronization block synchronized on the List.

To add an Overlay onto a Map View, create a new instance of the Overlay, and add it to the list, as shown in the snippet below:

```
List<Overlay> overlays = mapView.getOverlays();
MyOverlay myOverlay = new MyOverlay();
overlays.add(myOverlay);
mapView.postInvalidate();
```

The added Overlay will be displayed the next time the Map View is redrawn, so it's usually good practice to call postInvalidate after you modify the list to update the changes on the map display.

Annotating "Where Am I?"

This fi nal modifi cation to "Where Am I?" creates and adds a new Overlay that displays a red circle at the device's current position.

1. Start by creating a new MyPositionOverlay Overlay class in the WhereAmI project. package com.paad.whereami;

```
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.RectF;
import android.location.Location;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;
public class MyPositionOverlay extends Overlay {
```

```
@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
}
@Override
public boolean onTap(GeoPoint point, MapView mapView) {
return false;
}
}
```

2. Create a new instance variable to store the current Location, and add setter and getter methods for it.

```
Location location;
public Location getLocation() {
return location;
}
public void setLocation(Location location) {
this.location = location;
}
```

3. Override the draw method to add a small red circle at the current location.

```
private final int mRadius = 5;
@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
Projection projection = mapView.getProjection();
if (shadow == false) {
// Get the current location
Double latitude = location.getLatitude()*1E6;
Double longitude = location.getLongitude()*1E6;
GeoPoint geoPoint;
geoPoint = new GeoPoint(latitude.intValue(),longitude.intValue());
// Convert the location to screen pixels
Point point = new Point();
projection.toPixels(geoPoint, point);
RectF oval = new RectF(point.x - mRadius, point.y - mRadius,
point.x + mRadius, point.y + mRadius);
// Setup the paint
Paint paint = new Paint();
paint.setARGB(250, 255, 0, 0);
paint.setAntiAlias(true);
paint.setFakeBoldText(true);
Paint backPaint = new Paint();
backPaint.setARGB(175, 50, 50, 50);
backPaint.setAntiAlias(true);
RectF backRect = new RectF(point.x + 2 + mRadius,
point.y - 3*mRadius,
point.x + 65, point.y + mRadius);
// Draw the marker
canvas.drawOval(oval, paint);
canvas.drawRoundRect(backRect, 5, 5, backPaint);
canvas.drawText("Here I Am", point.x + 2*mRadius, point.y, paint);
}
super.draw(canvas, mapView, shadow);
        }
```

4. Now open the WhereAmI Activity class, and add the MyPositionOverlay to the MapView. Start by adding a new instance variable to store the MyPositionOverlay, then override onCreate to create a new instance of the class, and add it to the MapView's Overlay list.

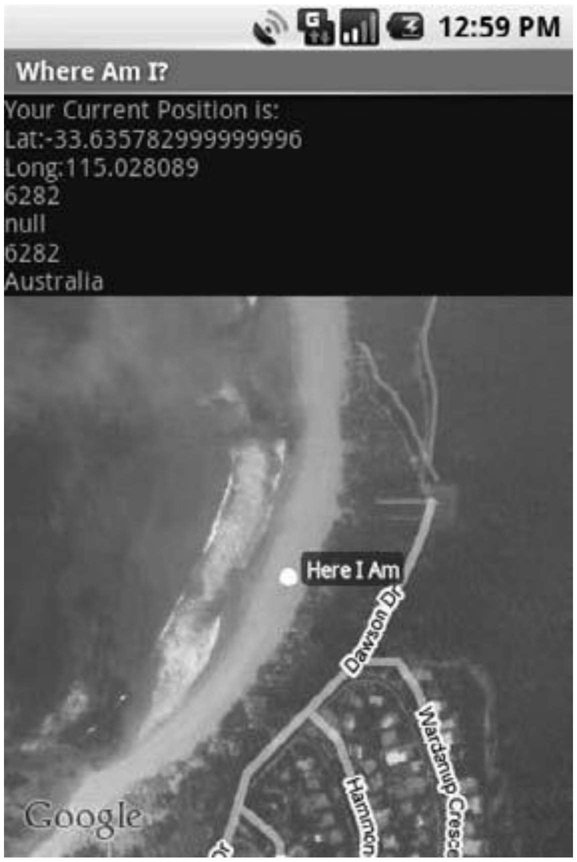
```
MyPositionOverlay positionOverlay;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
MapView myMapView = (MapView)fi ndViewById(R.id.myMapView);
mapController = myMapView.getController();
myMapView.setSatellite(true);
myMapView.setStreetView(true);
myMapView.displayZoomControls(false);
mapController.setZoom(17);
// Add the MyPositionOverlay
positionOverlay = new MyPositionOverlay();
List<Overlay> overlays = myMapView.getOverlays();

overlays.add(positionOverlay);
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);
}
```

5. Finally, update the updateWithNewLocation method to pass the new location to the overlay. 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 my location marker
positionOverlay.setLocation(location);
// 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);
        }
```

When run, your application will display your current device location with a red circle and supporting text, as shown in

Figure 7-7.



```
Your Current Position is:
Lat:-33.635782999999996
Long:115.028089
6282
null
6282
Australia
```

It's worth noting that this is not the preferred technique for displaying your current location on a map. This functionality is implemented natively by Android through the MyLocationOverlay *class. If you want to display and follow your current location, you should consider using this class (as shown in the next section) instead of implementing it manually as shown here.*