Creating and Using Overlays

Overlays are a way to add annotations and click handling to MapViews. Each Overlay lets you draw 2D primitives including text, lines, images and shapes directly onto a canvas, which is then overlaid onto a Map View.

You can add several Overlays onto a single map. All the Overlays assigned to a Map View are added as layers, with newer layers potentially obscuring older ones. User clicks are passed through the stack until they are either handled by an Overlay or registered as a click on the Map View itself.

Creating New Overlays

Each overlay is a canvas with a transparent background that you can layer on top of a Map View and use to handle map touch events.

To add a new Overlay, create a new class that extends Overlay. Override the draw method to draw the annotations you want to add, and override onTap to react to user clicks (generally when the user taps an annotation added by this overlay).

The following code snippet shows the framework for creating a new Overlay that can draw annotations and handle user clicks:

```
import android.graphics.Canvas;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
public class MyOverlay extends Overlay {
@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
if (shadow == false) {
[ ... Draw annotations on main map layer ... ]
}
else {
[ ... Draw annotations on the shadow layer ... ]
}
}
@Override
public boolean onTap(GeoPoint point, MapView mapView) {
// Return true if screen tap is handled by this overlay
return false;
}
}
```

Introducing Projections

The Canvas used to draw Overlay annotations is a standard Canvas that represents the visible display surface. To add annotations based on physical locations, you need to convert between geographical points and screen coordinates.

The Projection class lets you translate between latitude/longitude coordinates (stored as GeoPoints) and x/y screen pixel coordinates (stored as Points).

A map's Projection may change between subsequent calls to draw, so it's good practice to get a new instance each time. Get a Map View's Projection by calling getProjection, as shown in the snippet below:

```
Projection projection = mapView.getProjection();
```

Use the fromPixel and toPixel methods to translate from GeoPoints to Points and vice versa.

For performance reasons, the toPixel Projection method is best used by passing a Point object to be populated (rather than relying on the return value), as shown below:

```
Point myPoint = new Point();
// To screen coordinates
projection.toPixels(geoPoint, myPoint);
// To GeoPoint location coordinates
projection.fromPixels(myPoint.x, myPoint.y);
```

Drawing on the Overlay Canvas

Canvas drawing for Overlays is handled by overriding the Overlay's draw handler.

The passed-in Canvas is the surface on which you draw your annotations, using the same techniques introduced in Chapter 4 when creating custom User Interfaces for Views. The Canvas object includes the methods for drawing 2D primitives on your map (including lines, text, shapes, ellipses, images, etc.). Use Paint objects to define the style and color.

The following code snippet uses a Projection to draw text and an ellipse at a given location:

```
@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
Projection projection = mapView.getProjection();
Double lat = -31.960906*1E6;
Double lng = 115.844822*1E6;
GeoPoint geoPoint = new GeoPoint(lat.intValue(), lng.intValue());
if (shadow == false) {
Point myPoint = new Point();
projection.toPixels(geoPoint, myPoint);
// Create and setup your paint brush
Paint paint = new Paint();
paint.setARGB(250, 255, 0, 0);
paint.setAntiAlias(true);
paint.setFakeBoldText(true);
// Create the circle
int rad = 5;
RectF oval = new RectF(myPoint.x-rad, myPoint.y-rad, myPoint.x+rad, myPoint.y+rad);
// Draw on the canvas
canvas.drawOval(oval, paint);
canvas.drawText("Red Circle", myPoint.x+rad, myPoint.y, paint);
}
}
```

For more advanced drawing features, check out Chapter 11, where gradients, strokes, and fi lters are introduced that provide powerful tools for drawing attractive and compelling map overlays.