Location & Maps

Programming the Android Platform
Location Services

- Mobile applications can benefit from being location-aware, e.g.,
  - Routing from a current to a desired location
  - Searching for stores near a current location
- Android allows applications to determine & manipulate location
Location

- Represents a position on the Earth
- A Location instance consists of:
  - Latitude, longitude, a UTC timestamp
  - Optionally, altitude, speed, and bearing
LocationProvider

- Represent sources of location data
- Actual data may come from
  - GPS satellites
  - Cell phone towers
  - Internet
- Different LocationProviders will exhibit different tradeoffs between cost, accuracy, availability & timeliness
Location Provider Types

- **Passive**
  - Returns locations generated by other providers
  - Requires `android.permission.ACCESS_FINE_LOCATION`

- **Network**
  - Determines location based on cell tower and WiFi access points
  - Requires either
    - `android.permission.ACCESS_COARSE_LOCATION`
    - `android.permission.ACCESS_FINE_LOCATION`

- **GPS**
  - Determines location using satellites
  - Requires `android.permission.ACCESS_FINE_LOCATION`
LocationManager

- System service for accessing location data
  - `getSystemService(Context.LOCATION_SERVICE)`
- Enables
  - Determining the last known user location
  - Registering for location updates
  - Registering to receive Intents when the device nears a given location
LocationListener

- Defines callback methods that are called when Location or LocationProvider status changes
  - void onLocationChanged(Location location)
  - void onProviderDisabled(String provider)
  - void onProviderEnabled(String provider)
  - void onStatusChanged(String provider, int status, Bundle extras)
Obtaining Location

1. Start listening for updates from location providers
2. Maintain a "current best estimate" of location
3. When estimate is “good enough”, stop listening for location updates
4. Use best location estimate
public class LocationGetLocationActivity extends Activity {
  ...
  public void onCreate(Bundle savedInstanceState) {
    ...
    final LocationManager locationManager = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);
    bestReading = locationManager.
        getLastKnownLocation(LocationManager.GPS_PROVIDER);
  
    locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER, 0, 0, locationListener);
  
    ...
}
final LocationListener locationListener = new LocationListener()
public synchronized void onLocationChanged(Location location)
if (location.getAccuracy() < bestReading.getAccuracy())
bestReading = location;
tv.setText(getDisplayString(location));

....
Executors.newScheduledThreadPool(1).schedule(
    new Runnable() {
        public void run() {
            locationManager.removeUpdates(locationListener);
        }
    }, 10000, TimeUnit.MILLISECONDS);
...

Obtaining Location (cont.)
Determining Best Location

- Several factors to consider
  - Measurement time
  - Accuracy
  - Provider type
Battery Saving Tips

- Location measurement really drains the battery
- To limit battery use
  - Return updates less frequently
  - Restrict the set of Location Providers
    - Use least accurate (cheaper) provider necessary
  - Always check last known measurement
  - Turn off updates in onPause()
Maps

- A visual representation of area
- Today’s examples use Google Maps library
- Not part of standard Android distribution
  - Install SDK add-on
  - build against add-on ("Google APIs (Google Inc.)")
- Permissions
  - `<uses-library
    android:name="com.google.android.android.maps" />
  - `<uses-permission
    android:name="android.permission INTERNET" />

Maps Classes

- MapActivity
- MapView
- GeoPoint
- Overlay
- ItemizedOverlay
MapActivity

- Base class for Activities that display MapViews
- Subclass creates MapView in onCreate()
- Only one MapActivity is allowed per process
MapView

- Extends ViewGroup
- Displays a Map in one of several modes
  - Street View - photographs
  - Satellite View – aerial
  - Traffic View – real time traffic superimposed
- Supports panning and zooming
- Support overlay views
- Requires a Maps API key
  - See http://code.google.com/android/add-ons/google-apis/mapkey.html
public class MapsEarthquakeMapActivity extends MapActivity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    protected boolean isRouteDisplayed() {
        return false;
    }
}
<RelativeLayout ..">

<com.google.android.maps.MapView
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:apiKey="my Maps API key"
/>

</RelativeLayout>
GeoPoint

- Represents a location on Earth
  - latitude and longitude measured in microdegrees
  - 1 microdegree == 1 millionth of a degree
Overlay

- Manages information drawn over a map
  - E.g., points of interest within a given city
- MapView maintains a list of overlays
  - Retrieve via MapView.getOverlays()
ItemizedOverlay

- Subclass of Overlay
- Manages a list of OverlayItems
  - OverlayItems have a particular location
  - Draws a drawable at OverlayItem’s location
- Keeps track of a focused item
Example
public class MapsEarthquakeMapActivity extends MapActivity {
    List<Overlay> mapOverlays;
    Drawable mOverlayDrawable;
    EartQuakeDataOverlay itemizedOverlay;
    MapView mapView = null;
    public void onCreate(Bundle savedInstanceState) {

        new HttpGetTask().execute("http://api.geonames.org/earthquakesJSON?
            north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo");

        mapView.setBuiltInZoomControls(true);
        mapOverlays = mapView.getOverlays();
        mOverlayDrawable = this.getResources() .getDrawable(R.drawable.pushpin_red);
        itemizedOverlay = new EartQuakeDataOverlay(mOverlayDrawable);
    }
}
... // called when HttpResponse.execute() finishes
private void onFinishGetRequest(List<EarthQuakeRec> result) {
    for (EarthQuakeRec rec : result) {
        itemizedOverlay.addOverlay(new OverlayItem(rec.getGeoPoint(),
                    String.valueOf(rec.getMagnitude()),
                    ""));
    }
    mapOverlays.add(itemizedOverlay);
    MapController mc = mapView.getController();
    mc.setCenter(new GeoPoint((int) (14.6041667 * 1E6),
                                (int) (120.9822222 * 1E6)));
}
public class EarthQuakeDataOverlay extends ItemizedOverlay<OverlayItem> {

    ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();

    protected EarthQuakeDataOverlay(Drawable defaultMarker) {
        super(boundCenter(defaultMarker));
    }

    public void addOverlay(OverlayItem overlay) {
        mOverlays.add(overlay);
        populate();
    }

    protected OverlayItem createItem(int i) { return mOverlays.get(i); }

    public int size() { return mOverlays.size(); }
}
Lab Assignment
Source Code Examples

- LocationGetLocation
- MapsMapActivity