Location and Map APIs

Sensor Based Mobile Applications

Patrick Ausderau, Ulla Sederlöf, Jarkko Vuori

Helsinki Metropolia University of Applied Science

2022

Metropolia

Outline

Location

- android.location
- Google Play services location APIs

Lab - part 1

Map

- osmdroid

Lab - part 2

For more info:

`android.location`: `https://developer.android.com/reference/android/location/package-summary`

Google Play services location APIs: `https://developer.android.com/training/location/index.html`

osmdroid API: `https://github.com/osmdroid/osmdroid`

Google Maps API: `https://developers.google.com/maps/documentation/android-sdk/intro`

Location

- Mobile users take their devices with them almost everywhere.
- Adding location awareness to your app offers users a more contextual experience. E.g.
  - propose the closest shop, restaurant,...
  - local weather forecast, timezone,...
  - sport tracking
  - navigation
  - ...

Android Location Strategies

- Global Navigation Satellite System (GNSS) and Satellite-Based Augmentation Systems (SBAS): GPS, GLONASS, Galileo, BeiDou, NavIC, QZSS,...[1]
  - very precise (about 3 to 15 meters, 1 meter with Galileo, L5 band GPS receiver up to 30 centimeters)
  - provides geolocation and time information anywhere on or near the Earth (e.g. flying, middle of ocean,...)
  - works in offline mode (and in airplane mode)
  - only works outdoor or close to window (can even have problems e.g. in a street with big building, in deep forest, high rocks,...)
  - can be turned off by user
  - consumes more battery
  - can be slow
  - takes time to get location when turned on (up to 1 minute)

[1] depending on hardware. If interested to test see e.g. `https://github.com/barbeau/gpstest` or `https://developer.android.com/guide/topics/sensors/gnss`

Android Location Strategies

- Network based: Cell-ID, Wi-Fi,...
  - fast
  - low battery consumption (no extra to normal network tasks)
  - varying in precision (about 10 to 300 meters)
  - is almost always available (except in airplane mode, in the middle of the ocean,...)
  - works inside building

`android.location` V.S. Google Play APIs

- `android.location`
  - native in the Android framework
  - more work e.g. to choose location provider (e.g. one minute old location data from one source can be more precise that the newest data from another source) and power management strategies
- Google Play services location APIs
  - requires proprietary Google Play services client library
  - automates tasks such as location provider choice and power management
  - adds new features such as activity detection
  - recommended (or as stated by Android developer: *"you are strongly encouraged to switch to the Google [Play] Location Services API"*)
  - requires network connection between the user's device and Google service

android.location and Google Play APIs

- Specify App Permissions in your manifest
  - ACCESS_COARSE_LOCATION (network based location)
  - ACCESS_FINE_LOCATION (GPS (include network) based location)
  - If target Android 5.0 (API level 21) or higher, then also

```
<uses-feature
android:name="android.hardware.location.gps" />
<!-- and/or -->
<uses-feature
android:name="android.hardware.location.network"
/>
```

android.location and Google Play APIs

Continued...

- Specify App Permissions in your manifest
  - If target Android 10 (API 29) or higher and want to access location data when the app is running in foreground, you must declare a foreground service type of location

```
<application>
  <!-- activities,... -->
  <service
    android:name="MyNavigationActivity"
    android:foregroundServiceType="location">
    <!-- Any inner elements would go here. -->
  </service>
</application>
```

  - If target Android 10 (API 29) or higher and want to track location when app is in background, then add permission `ACCESS_BACKGROUND_LOCATION`

android.location and Google Play APIs

Continued...

- For Android >= 6 (API 23), you need to request permissions at run time[2] (or catch SecurityException)

```
1  if ((Build.VERSION.SDK_INT >= 23 &&
       ContextCompat.checkSelfPermission(this,
       android.Manifest.permission.ACCESS_FINE_LOCATION) !=
       PackageManager.PERMISSION_GRANTED)) {
2      ActivityCompat.requestPermissions(this,
           arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION),
           0)
3  }
4  // similar for ACCESS_COARSE_LOCATION,
       ACCESS_BACKGROUND_LOCATION,...
```

[2] https://developer.android.com/training/permissions/requesting

`android.location` and Google Play APIs

Continued...

- For Android >= 11 (API level 30), if you request at run time the foreground and the background location permissions at the same time, the system ignores the request and doesn't grant your app either permission. Request the permissions separately when first used in app!

android.location

- Implements the android.location.LocationManager and override the onLocationChanged() to get the current (last known) location of the device

```
1  class LocationActivity : Activity(), LocationListener {
2    //...
3    override fun onLocationChanged(p0: Location) {
4      //new location react...
5      Log.d("GEOLOCATION", "latitude: ${p0.latitude}, longitude:
           ${p0.longitude}, etc: $p0")
6    }
7    // if needed, also override...
8    override fun onProviderEnabled(p0: String) {}
9    override fun onProviderDisabled(p0: String) {}
10   //if needed; but deprecated in API level 29
11   override fun onStatusChanged(p0: String?, p1: Int, p2: Bundle?)
         {}
12 }
```

android.location

Continued...

- Acquire a reference to the system Location Manager and request location updates

```
1  //companion object private lateinit var lm: LocationManager
2  //onCreate...
3  lm = getSystemService(Context.LOCATION_SERVICE) as
       LocationManager
4  //somewhere e.g. in "start tracking" button click listener
5  lm.requestLocationUpdates(
6          LocationManager.GPS_PROVIDER, // or
               LocationManager.NETWORK_PROVIDER
7          60 * 1000,
8          50f,
9          this
10 )
11 //somewhere e.g. in "stop tracking" button click listener
12 lm.removeUpdates(this)
```

- For getting all location changes, use `0` for time and for distance parameters.

android.location.Geocoder[3]

- Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate.
- Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into a (partial) address.
- Not included in the core android framework, test if the phone implements it with `isPresent()` method

[3] https://developer.android.com/reference/android/location/Geocoder

android.location.Geocoder

```
1  private fun getAddress(lat: Double, lng: Double): String {
2      val geocoder = Geocoder(this)
3      var address = ""
4      if (Build.VERSION.SDK_INT >= 33) {
5          geocoder.getFromLocation(lat, lng, 1) {
6              address = it.first().getAddressLine(0)
7          }
8      } else {
9          address = geocoder.getFromLocation(lat, lng, 1)?
               .first()?.getAddressLine(0) ?: ""
10     }
11     return address
12 }
```

Google Play services location APIs

Using the Google Play services location APIs to get the current (last known) location of the device

- Add Google Play Services to your project dependencies[4]
  - in project build gradle check that you have `google()` repository
  - in module build gradle add in dependencies

```
implementation
    'com.google.android.gms:play-services-location:20.0.0'
```

- Specify App Permissions in your manifest
- create an instance of the Fused Location Provider Client

```
private lateinit var fusedLocationClient:
    FusedLocationProviderClient
```

[4] https://developers.google.com/android/guides/setup

Google Play services location APIs

Continued...

- Get the last known location using the listener provided by Google API Client.

```
1  override fun onCreate(savedInstanceState: Bundle?) {
2    // if Android >= 6, check Permissions at runtime!!!
3    fusedLocationClient =
         LocationServices.getFusedLocationProviderClient(this)
4    if (ActivityCompat.checkSelfPermission(this,
         Manifest.permission.ACCESS_FINE_LOCATION) ==
         PackageManager.PERMISSION_GRANTED) {
5      fusedLocationClient.lastLocation.addOnSuccessListener {
6        Log.d("GEOLOCATION","last location latitude:
             ${it?.latitude} and longitude: ${it?.longitude}")
7      }
8    }
9  }
```

Google Play services location APIs

- Track location change using callback listener provided by Google API Client.

```
1  // lateinit fusedLocationClient...
2  private lateinit var locationCallback: LocationCallback
3  //onCreate(), super..., fusedLocationClient = ...
4  locationCallback = object: LocationCallback() {
5    override fun onLocationResult(locationResult: LocationResult?)
         {
6      locationResult ?: return
7      // e.g. loop through all the locations in the track, very
           likely, you want only the latest result
8      for (location in locationResult.locations) {
9        Log.d("GEOLOCATION", "location latitude:
             ${location.latitude} and longitude:
             ${location.longitude}" )
10     }
11   }
12 }
```

Google Play services location APIs

Continued...

- Request Location change to get the new locations

```
1  //somewhere e.g. in "start tracking" button click listener
2  val locationRequest = LocationRequest
3      .create()
4      .setInterval(1000)
5      .setPriority(PRIORITY_HIGH_ACCURACY)
6  //if permissions granted...
7  fusedLocationClient.requestLocationUpdates(locationRequest,
       locationCallback, Looper.getMainLooper())
8
9  //somewhere e.g. in "stop tracking" button click listener
10 fusedLocationClient.removeLocationUpdates(locationCallback)
```

Google Play services location APIs

Google Play services location APIs offers other services such as

- Create and monitor geofences
- Activity Recognition API
- Place API
- etc.
  `https://developers.google.com/location-context/`

Lab_w2_d3_Location (part 1)

- make an app that tracks location of the device. At your preference, use the `android.location` or the Google Play services location APIs and feel free to test network or GPS based
- Do something fun with it. E.g.
  - remember the fastest speed
  - calculate the total walked distance. Hint: `location.distanceTo(prevLocation)`
  - ...
- Test your app on a real device[5].

[5] testing on emulator is possible `https://developer.android.com/studio/run/advanced-emulator-usage#extended` but not optimal.

Maps APIs

Maps APIs will allow you to add map to your app. They will handle for you

- access to Maps servers and downloading tiles
- map display
- touch gestures on the map
- add markers, polygons and overlays
- change the user's view of a particular map area
- change the zoom level
- ...

osmdroid

- osmdroid is a (almost) full/free replacement for Android's MapView (v1 API) class. It also includes a modular tile provider system with support for numerous online and offline tile sources and overlay support with built-in overlays for plotting icons, tracking location, and drawing shapes.
- uses OpenStreetMap data
  - collaborative project
  - free editable map of the world
  - `https://www.openstreetmap.org/`

osmdroid

Using osmdroid to create an app with a Map

- Add mavenCentral repository to your project `build.gradle`

```
repositories {
    //...
    mavenCentral()
}
```

- Add osmdroid dependencies to your app `build.gradle`

```
dependencies {
    //...
    implementation 'org.osmdroid:osmdroid-android:6.1.14'
    // old android.preference is deprecated, switch to androidx
    implementation 'androidx.preference:preference-ktx:1.2.0'
}
```

osmdroid

Continued...

Using osmdroid to create an app with a Map

- In most cases, you will have to set the following authorizations in your AndroidManifest.xml:
  - `ACCESS_COARSE_LOCATION` (or `ACCESS_FINE_LOCATION`) and eventually `ACCESS_BACKGROUND_LOCATION`
  - `ACCESS_NETWORK_STATE`
  - `INTERNET`
  - `WRITE_EXTERNAL_STORAGE`

osmdroid

Continued...

Using osmdroid to create an app with a Map

- In your main activity set your user agent

```
1  //...
2  import org.osmdroid.config.Configuration
3  import org.osmdroid.tileprovider.tilesource.TileSourceFactory
4  //class ...
5  override fun onCreate(savedInstanceState: Bundle?) {
6    super.onCreate(savedInstanceState)
7    //important! set your user agent to prevent getting banned from
         the osm servers
8    Configuration.getInstance().load(this,
         PreferenceManager.getDefaultSharedPreferences(this))
9    setContent {
10     MainView()
11   }
12 }
```

osmdroid

Continued...

Using osmdroid to create an app with a Map

- Make a compose map[6]

```
@Composable
fun composeMap(): MapView {
    val context = LocalContext.current
    val mapView = remember {
        MapView(context).apply {
            id = R.id.map
        }
    }
    return mapView
}
```

- and create `ids.xml` in `res/values`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="map" type="id" />
</resources>
```

[6] consider cleaner and stop downloading tiles when app move to background https://github.com/osmdroid/osmdroid/issues/1674

osmdroid

Continued...
Using osmdroid to create an app with a Map

- Set the tiles source
  `map.setTileSource(TileSourceFactory.MAPNIK)`
- Then add ability to zoom with 2 fingers (multi-touch)
  `map.setMultiTouchControls(true)`
- Set zoom level with the map controller (e.g. in onCreate. Avoid to change it all the time!)
  `map.controller.setZoom(9.0)`
- Move the center of the map on a default view point with the map controller (e.g. in location change listener)
  `map.controller.setCenter(GeoPoint(60.17, 24.95))`

osmdroid

Continued...

Using osmdroid to create an app with a Map

```
1  @Composable
2  fun ShowMap() {
3      val map = composeMap()
4      // hard coded zoom level and map center only at start
5      var mapInitialized by remember(map) { mutableStateOf(false) }
6      if(!mapInitialized) {
7          map.setTileSource(TileSourceFactory.MAPNIK)
8          map.controller.setZoom(9.0)
9          map.controller.setCenter(GeoPoint(60.17, 24.95))
10         mapInitialized = true
11     }
12     AndroidView({ map })
13 }
```

osmdroid

Continued...

Using osmdroid to create an app with a Map

- Add markers, draw lines,...[7]

```
1  @Composable
2  fun ShowMap(mapViewModel: MapViewModel) {
3    // val map, init...
4    // observer (e.g. update from the location change listener)
5    val address by mapViewModel.mapData.observeAsState()
6    val marker = Marker(map)
7    AndroidView({ map }) {
8      address ?: return@AndroidView
9      it.controller.setCenter(address?.geoPoint)
10     marker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM)
11     marker.position = address?.geoPoint
12     marker.closeInfoWindow()
13     marker.title = address?.address
14     map.overlays.add(marker)
15     map.invalidate()
16   }
17 }
```

[7] https://github.com/osmdroid/osmdroid/wiki/Markers,

osmdroid

Continued...
Using OSMBonusPack[8] that complements osmdroid

- Routes and Directions
- Points of Interests (directory services)
- Marker Clustering
- Support for KML and GeoJSON content
- GroundOverlay
- and more...

[8] https://github.com/MKergall/osmbonuspack

Lab w2_d3_Map (part 2, include location)

- add a map to your Location lab (at your convenience osmdroid or Google Map)
- show the map centered on the current (or last known) location
- Add a pin with extra info (the Street Address and Latitude/Logitude/Altitude)
- Optional: when location change, draw lines between the points on the map
- Optional: give a try to the OSMBonusPack